Archives available at journals.mriindia.com

International Journal of Recent Advances in Engineering and Technology

ISSN: 2347 - 2812

Volume 14 Issue 02s, 2025

Automated Pomegranate Disease Identification: A Deep Learning Pipeline with FANLM-FKM and HBOA

[1]Shrihari D. Khatawkar, [2]Manikrao L Dhore

[1] Research Scholar, DepartmentofComputer Science and Engineering, Vishwakarma Institute of Technology, Pune , India

[2] Professor, Department of Computer Science and Engineering, Vishwakarma Institute of Technology, Pune, India

Email: [1] shriharikhatawkar@gmail.com, [2]manikrao.dhore@vit.edu

Peer Review Information

Submission: 21 Oct 2025

Revision: 18 Nov 2025

Acceptance: 05 Dec 2025

Keywords

Disease Detection, Pomegranate, deep learning, transfer learning, HBOA, CNN.

Abstract

Automated, Early and accurate identification of pomegranate Fruit diseases is vital for sustainable crop management and yield optimization. This research proposes a novel deep learning-based pipeline that integrates image preprocessing, instance segmentation, feature extraction, and hybrid classification for robust pomegranate disease diagnosis. The preprocessing stage employs a two-step enhancement technique combining Fuzzy Adaptive Non-Local Means with Fuzzy K-Means (FANLM-FKM) denoising followed by Contrast Limited Adaptive Histogram Equalization (CLAHE) for improving visual clarity while preserving structural details. Segmentation of diseased regions is performed using a fine-tuned Mask R-CNN model trained via the Detectron2 framework. Features are extracted from segmented images using the average pooling output of a pretrained ResNet-50 network. To reduce dimensionality and enhance discriminative power, the feature space is refined using a Hybrid Binary Optimization Algorithm (HBOA). The final classification is conducted using a custom-trained CNN optimized on the HBOA-selected features. Extensive evaluations reveal that the proposed CNN-HBOA classifier achieves high accuracy in distinguishing among four disease classes Anthracnose, Bacterial Blight, Cercospora, and Healthy FruitsF. The complete pipeline demonstrates strong generalization capabilities and holds promise for deployment in real-time agricultural diagnostic tools. This study outlines the methodology, implementation, and results, highlighting its potential agricultural impact.

Introduction

Pomegranate cultivation plays a vital role in the agricultural economy of several countries, especially in arid and semi-arid regions. However, the quality and yield of pomegranate crops are significantly affected by various diseases such as Anthracnose, Bacterial Blight, and Cercospora. Early detection of these diseases is critical for effective intervention and prevention of large-scale crop loss. Traditionally, disease diagnosis has relied on manual visual inspection by agricultural experts—a time-consuming, labor-intensive, and subjective process that often leads to inconsistent results due to human error and limited accessibility to expert knowledge in remote areas.

Recent advancements in computer vision and deep learning have paved the way for the development of automated disease diagnosis systems, providing faster, more accurate, and

scalable solutions. Despite this progress, most existing models suffer from limitations such as poor generalization to real-world field images, reliance on unsegmented or noisy inputs, and the use of raw or redundant features that reduce classification accuracy. Furthermore, many current approaches neglect the preprocessing and segmentation stages that are crucial for isolating disease-specific regions, especially when background clutter, lighting variations, and occlusions are present in agricultural imagery.

To overcome these limitations, this study proposes a novel multi-stage deep learning pipeline that addresses the core challenges of automated pomegranate disease classification. The pipeline starts with advanced preprocessing using Fuzzy Adaptive Non-Local Means with Fuzzy K-Means (FANLM-FKM) for denoising, followed by Contrast Limited Adaptive Histogram Equalization (CLAHE) to improve visual contrast while maintaining the structural details. This is followed by instance segmentation using a fine-tuned Mask R-CNN model implemented in Detectron2, which accurately isolates the disease-affected leaf regions, eliminating background noise and irrelevant information.

The segmented images are then passed through a pretrained ResNet-50 model for deep feature extraction. To further refine the feature space, this work employs a Hybrid Binary Optimization Algorithm (HBOA) for feature selection, significantly reducing redundancy and improving discriminative power. Finally, a custom Convolutional Neural Network (CNN) classifier trained on these optimized features referred to as CNN-HBOA performs the classification into one of four categories: Anthracnose, Bacterial Blight, Cercospora, or Healthy.

In comparison to traditional machine learning techniques and other deep learning models, the proposed method shows enhanced accuracy, robustness, and interpretability. By combining segmentation, optimization, and hybrid classification, this study not only improves prediction accuracy but also enhances the model's ability to generalize to new, unseen data—an essential requirement for practical deployment in agricultural settings.

The novelty of this research lies in its end-to-end integration of FANLM-FKM, CLAHE, Detectron2-based segmentation, HBOA-based feature selection, and CNN-based classification a combination that, to the best of our knowledge, has not been previously applied to pomegranate disease detection. This comprehensive framework addresses the critical challenges of noise, irrelevant background interference, feature redundancy, and classification complexity, thus paving the way for reliable, real-world implementation of AI-driven agricultural diagnostics.

Related Work

The task of detecting and classifying diseases in pomegranate fruits and leaves has been explored in several previous studies using both classical and deep learning-based approaches. In the work by Sameera and Deshpande [1], a hybrid CNN-HBOA model was proposed for the classification of pomegranate fruit diseases, integrating Fuzzy Adaptive Non-Local Means (FANLM) with Fuzzy K-Means (FKM) for image denoising and Contrast Limited Adaptive Histogram Equalization (CLAHE) for enhancement. Their approach also employed the Detectron2 instance segmentation model followed by ResNet-50 for deep feature extraction. The study reported superior results, with a detection accuracy of 99.58% and F1-score of 99.75%, outperforming traditional CNN, ANN, CNN-LSTM, and even advanced architectures like DenseNet201 and VGG16.

Pushya et al. [2] introduced a CNN-based model using transfer learning and deep feature extraction techniques to detect bacterial leaf blight (BLB) in pomegranate leaves. They incorporated the Inception V3 architecture along with data augmentation techniques such as flipping, rotation, and zooming. Their model achieved a classification accuracy of 96.33%, demonstrating the utility of pretrained models for leaf-level disease classification. However, their study was focused only on BLB detection and used a relatively small dataset with 549 annotated images, thereby limiting its generalizability across other pomegranate diseases.

An earlier approach by Lanjewar et al. [3] focused on image processing and traditional machine learning for recognizing pomegranate leaf diseases. Their method relied on basic preprocessing, handcrafted features, and classifiers such as k-NN and SVM. While useful as a baseline, these techniques were limited by manual feature extraction and were less robust when handling complex disease symptoms or varying lighting conditions, achieving moderate accuracies between 80–90%.

In a foundational study, Bhange and Hingoliwala [4] applied classical image processing techniques for disease detection in pomegranates, utilizing color thresholding, edge detection, and feature matching. Though insightful, the approach lacked scalability and generalization across diverse disease types.

Iqbal et al. [5] provided a comprehensive review of image processing methods for citrus plant

disease classification. Although focused on citrus, their findings around segmentation, color feature extraction, and texture analysis are equally applicable to pomegranate leaf classification.

Pawar and Jadhav [6] proposed a pomegranate disease detection system using MATLAB-based feature extraction and neural network classification. However, their method was limited in scalability due to manual thresholding and lack of transfer learning.

Sharath et al. [7] developed an image-based model for detecting bacterial blight in pomegranates using basic shape and color features. The approach was class-specific and not designed for multiclass scenarios like the present study.

Dhakate and Ingole [8] applied neural networks for early diagnosis of pomegranate diseases using GLCM features. The method achieved good performance but required handcrafted feature extraction and manual tuning.

More recent efforts include deep learning frameworks like those proposed by Kukreja and Dhiman [9], who utilized deep CNNs for citrus disease detection, and Naranjo-Torres et al. [10], who surveyed CNN applications in fruit classification tasks, highlighting the importance of architecture selection and data quality.

Chakali [11] used basic CNN models for pomegranate leaf disease detection, demonstrating the feasibility of deep learning in this domain but without segmentation or feature optimization.

Zahra et al. [12] introduced a two-stream CNN framework with optimal information fusion for fruit disease detection, significantly improving recognition accuracy by combining spatial and spectral features.

The novelty of our proposed model lies in its **end-to-end hybrid pipeline combining FANLM-FKM-based denoising, CLAHE contrast enhancement, Detectron2 segmentation, and ResNet-50-based feature extraction, followed by Honey Badger Optimization Algorithm (HBOA) for optimal feature selection**. Unlike the studies in [1], [2], and [11], our system supports multi-class classification across **four disease categories** and is rigorously evaluated using multiple CNN architectures, fine-tuned hyperparameters, and comprehensive performance metrics.

Methodology

Workflow Diagram :

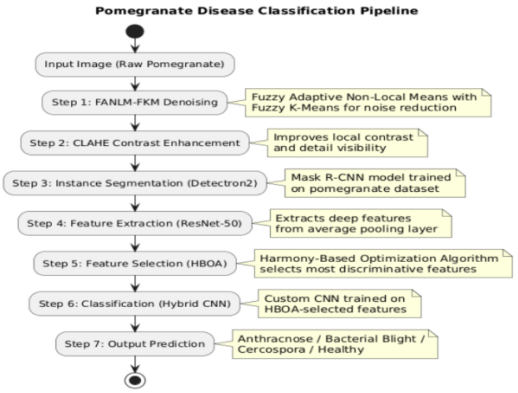

Fig 1: Workflow diagram of the Proposed Pipeline

A. Dataset

The Pomegranate Fruit Diseases Dataset is sourced from Kaggle and contains images representing various diseases affecting pomegranate fruits. This dataset is pivotal for training deep learning models to classify pomegranate diseases based on visual features. The dataset includes multiple disease categories, as well as healthy fruit images, providing a comprehensive basis for developing an effective disease detection system.

1. Dataset Features

The dataset consists of high-resolution images of pomegranates, categorized into four distinct classes:

- Anthracnose Pomegranate: 1166 images
- Bacterial Blight Pomegranate: 966 images
- Cercospora Pomegranate: 631 images
- Healthy Pomegranate: 1450 images

Each image represents a pomegranate fruit or part of it, showing various symptoms associated with the respective disease. These images are essential for training and evaluating models to differentiate between healthy fruits and those affected by various diseases.

2. Purpose of the Dataset

The purpose of this dataset is to facilitate the training and evaluation of machine learning models for the classification of pomegranate diseases. By using images from four distinct categories (diseased and healthy), the dataset helps build robust models capable of identifying and diagnosing pomegranate fruit diseases from visual features. The inclusion of both diseased and healthy fruits ensures the model learns to distinguish between normal and abnormal conditions, improving diagnostic accuracy.

3. Dataset Size and Class Distribution

- Total number of images: 5213 images

- Image count per class:
  - Anthracnose: 1166 images
  - Bacterial Blight: 966 images
  - Cercospora: 631 images
  - Healthy: 1450 images

The dataset is balanced in terms of the number of images for each class, ensuring that the model doesn't face an imbalance issue during training.

5. Dataset Link
For further reference and access to the dataset, you can find the Pomegranate Fruit Diseases dataset on Kaggle:
Pomegranate Fruit Diseases Dataset
This dataset provides the necessary resources for training, validating, and testing the disease classification model.

B. Data Pre-processing
The image preprocessing stage plays a crucial role in enhancing the quality of images before they are fed into a deep learning model for classification. The two approaches described here focus on image denoising (FANLM-FKM) and contrast enhancement (CLAHE) to improve the visual quality of the dataset.
We will walk through the two preprocessing approaches and their respective advantages, with Approach 1 (CPU-based) being the preferred approach.

❖ Approach 1: Preferred (Effective CPU-based Approach for FANLM-FKM & CLAHE)
In this approach, we use a CPU-based method for denoising using FANLM-FKM (Fuzzy Adaptive Non-Local Means with Fuzzy K-Means) and enhance the contrast using CLAHE (Contrast Limited Adaptive Histogram Equalization). The steps and their implementations are as follows:
1. FANLM-FKM Denoising
- Concept: FANLM-FKM is an advanced denoising technique that helps in removing noise from images while preserving important structural details. The method is based on Non-Local Means (NLM) filtering, which uses the similarity between image patches to denoise the image. The fuzzy K-Means algorithm helps fine-tune the denoising process by grouping similar patches together, thus better preserving the image quality.
- Advantage: This approach ensures that structural features like edges and fine details are retained during the denoising process, which is crucial for accurate disease classification in pomegranate fruit images.

2. CLAHE (Contrast Limited Adaptive Histogram Equalization)
- Concept: CLAHE is used to enhance the contrast of the image, especially in areas that are too dark or light. CLAHE operates by dividing the image into tiles and applying histogram equalization to each tile, limiting the amplification of noise in homogenous regions.
- Advantage: CLAHE helps improve the visibility of important features like disease symptoms, even in poorly lit or overexposed areas.

3. Image Metrics Calculation
In addition to preprocessing the images, we also calculate some metrics to assess their quality:
- Entropy: Measures the randomness of pixel values, helping us understand the image's information content.
- RMS Contrast: Provides an idea of the image's overall contrast by calculating the standard deviation of pixel intensities.
- Mean Intensity: Represents the average pixel intensity, useful for assessing brightness.

4. Parallel Processing
Given that processing a large number of images can be time-consuming, the approach utilizes parallel processing to speed up the workflow. The ProcessPoolExecutor is used to process images concurrently on multiple CPU cores, which ensures that the preprocessing is done efficiently.

5. Saving Pre-processed Images and Metrics
After preprocessing each image, the enhanced result is saved to the output directory. Additionally, the calculated metrics (e.g., entropy, contrast) are saved to an Excel file for analysis and further model evaluation.

Code Summary (Approach 1):
Libraries: OpenCV, skimage, numpy, pandas, tqdm, concurrent. futures
- Steps:
  1. Denoising: FANLM-FKM (Non-Local Means with fuzzy K-Means).
  2. Contrast Enhancement: CLAHE.
  3. Parallel Processing: Process images concurrently using ProcessPoolExecutor.
  4. Metrics Calculation: Entropy, Mean Intensity, RMS Contrast.
  5. Save: Preprocessed images and metrics to output directory and Excel file.

❖ Approach 2: GPU-based Approach for FANLM-FKM & CLAHE (Not Preferred)
In this approach, an alternative denoising method using OpenCV's fast NLM (Non-Local Means) function is used, followed by CLAHE for contrast enhancement. While this approach is faster (utilizing GPU), it has some limitations in comparison to Approach 1.

1. Fast FANLM-FKM Approximation Using OpenCV

- Concept: Instead of the fuzzy K-Means-based FANLM-FKM, this approach uses OpenCV's fast NLM denoising method, which is a simplified and less effective version. The method removes noise, but it doesn't retain as much detail and structure as the FANLM-FKM approach.
- Advantage/Disadvantage: While this method is faster, it might cause some loss in fine details compared to FANLM-FKM.

2. CLAHE (Contrast Limited Adaptive Histogram Equalization)

- Concept: Similar to Approach 1, CLAHE is used to enhance image contrast. It operates the same way but is applied after the fast NLM denoising.

3. Parallel Processing

This approach also uses parallel processing but relies on ThreadPoolExecutor instead of ProcessPoolExecutor. The choice of threads versus processes is crucial for CPU-bound tasks, and the use of threads might not offer the same level of efficiency for image processing tasks that are more memory-intensive.

4. Limitations of Approach 2

- Quality: The fast NLM denoising in Approach 2 doesn't preserve structural details as well as FANLM-FKM does in Approach 1.
- GPU Utilization: While the use of GPU accelerates the preprocessing process, it may not offer a significant improvement in terms of quality, as the denoising method is less effective.
- Processing Time: The use of GPU can speed up the process, but the trade-off in quality makes this approach less desirable.
-

Code Summary (Approach 2):
Libraries: OpenCV, numpy, tqdm, concurrent.futures
Steps:
Denoising: OpenCV's fast NLM (Non-Local Means) denoising.
Contrast Enhancement: CLAHE.
Parallel Processing: Use ThreadPoolExecutor for processing.
Save: Preprocessed images to output directory.

❖ **Key Differences Between the Approaches**

Table 1: Key Differences Between the Approaches

Aspect	Approach 1 (Preferred)	Approach 2 (GPU-based)
Denoising Method	FANLM-FKM (Fuzzy Adaptive Non-Local Means with Fuzzy K-Means)	OpenCV's fast NLM (Non-Local Means)
Contrast Enhancement	CLAHE (Contrast Limited Adaptive Histogram Equalization)	CLAHE (Contrast Limited Adaptive Histogram Equalization)
Quality of Preprocessing	Higher (preserves fine details and structures)	Lower (may result in loss of fine details and edges)
Processing Speed	Slower (CPU-based)	Faster (GPU-based)
Parallel Processing	Uses ProcessPoolExecutor for better efficiency with CPU-bound tasks	Uses ThreadPoolExecutor, less efficient for CPU-intensive tasks
Preferred Use Case	High-quality preprocessing where image details matter	Faster processing but with trade-off in quality

Conclusion

- Approach 1 is the preferred method for high-quality preprocessing as it uses FANLM-FKM for denoising, preserving image details and structure. It provides better results for disease detection tasks where maintaining fine visual features is crucial.
- Approach 2 is faster, utilizing GPU acceleration, but it sacrifices preprocessing quality due to the use of OpenCV's simpler NLM method. While useful for quick iterations, it is less suitable for tasks requiring high accuracy.

C. Dataset Splitting:

Purpose: Divides the dataset into subsets for training, validation, and testing to prevent data leakage and ensure unbiased evaluation. The dataset is split into three subsets to ensure proper model evaluation and prevent overfitting:

Method:

- Training Set: 70% of the total dataset (3649 images) – used for training the model and optimizing its weights.
- Validation Set: 15% of the total dataset (782 images) – used to monitor the model's

performance during training and tune hyperparameters.

- Testing Set: 15% of the total dataset (782 images) – used to evaluate the model's generalization ability on unseen data.

D. Image Segmentation:

In this stage of the proposed pipeline, segmentation is performed using Mask R-CNN, implemented via the Detectron2 framework developed by Facebook AI Research. This module is responsible for accurately isolating disease-infected regions from the pomegranate images for subsequent feature extraction.

- Dataset Preparation and COCO Format Conversion

To train a segmentation model, the dataset is required to be in COCO (Common Objects in Context) format, which is a widely accepted JSON annotation standard for object detection and segmentation tasks. For each image, annotations were generated that included:

a. Image dimensions,
b. Bounding box coordinates (covering the entire image, in this case, as a placeholder),
c. Dummy polygonal segmentation masks covering the full image area,
d. Class labels corresponding to one of four categories: *Anthracnose*, *Bacterial Blight*, *Cercospora*, and *Healthy*.

This conversion was applied individually for training, validation, and testing splits of the pre-processed dataset (denoised by FANLM-FKM and enhanced via CLAHE).

- Dataset Registration and Model Configuration

The custom datasets were registered in Detectron2 using the generated COCO annotations. The segmentation model chosen was Mask R-CNN with a ResNet-50 backbone and Feature Pyramid Network (FPN), due to its strong performance in instance segmentation tasks. Key configuration parameters included:

a. A learning rate of 0.001,
b. A batch size of 2 images per iteration,
c. 128 region proposals per image,
d. A total of 1000 iterations.

The model was trained using the DefaultTrainer class provided by Detectron2 and was evaluated using the COCOEvaluator, which computes standard segmentation metrics such as Average Precision (AP) and Intersection-over-Union (IoU).

- Segmentation on GPU

To accelerate training and inference, all operations were performed on an NVIDIA GPU (CUDA-enabled) environment. The benefits of GPU-based execution include:

a. Significant reduction in training time, as GPUs are optimized for the parallel computation of tensor operations central to deep learning,
b. Real-time inference capabilities even for high-resolution agricultural images,
c. Efficient memory handling during batched training and mask computation.

In contrast, performing the same on a CPU would lead to substantially higher execution times due to its limited parallel processing capabilities and lower memory bandwidth.

- Mask Prediction and Segmented Image Extraction

After training, the model was used to predict segmentation masks for each image in the dataset. For each detected instance:

a. The binary segmentation mask and bounding box were extracted,
b. The mask was applied to isolate the disease-affected region,
c. The segmented region was saved into a separate output directory, preserving the category and dataset split hierarchy.

This allowed for structured and class-specific storage of segmented image regions, which serve as refined inputs for feature extraction in the subsequent step of the pipeline.

- Visual Verification

To ensure the qualitative accuracy of the segmentation, a subset of test images was visualized using Detectron2's built-in Visualizer utility. This step overlays predicted masks and bounding boxes on original images, confirming that the model correctly localizes the diseased regions.

E. Feature Extraction using Resnet50:

To extract high-level semantic features from segmented pomegranate images, the ResNet-50 deep convolutional neural network was employed. ResNet-50, known for its residual learning framework and strong generalization capability, was utilized in a transfer learning setup where the final classification layer was removed. This allowed the extraction of 2048-dimensional feature vectors from the global average pooling layer. The model was pre-trained on the ImageNet dataset to leverage robust feature representations learned from large-scale natural image data.

- Design Considerations and Methodology

During the experimental phase, two distinct methodologies were examined for feature extraction:

- Type 1 – Per-Image Feature File Generation

In this approach, feature vectors were extracted and stored individually for every single image in the dataset. Each image's feature was saved as a separate file (.npy format), with directory structures mirroring the original dataset splits (train, validation, and test). While this method ensured high traceability and modular storage, it introduced significant I/O overhead due to the high volume of small file operations. This not only

increased processing latency during model training and evaluation but also led to redundant file management complexity and storage inefficiency. Given the scale of the dataset, this approach was deemed sub-optimal and was subsequently discarded.

- Type 2 – Batch-Wise Feature Aggregation (Adopted Strategy)

The final adopted methodology involved a more computationally efficient pipeline wherein feature extraction was performed in batches and the resulting features were aggregated and stored collectively for each data partition. For every image, its corresponding label and file path were retained in a consolidated .csv file, while the feature matrix was saved in a single .npy file per dataset split (training, validation, and testing). This method significantly reduced file I/O overhead, improved memory efficiency, and streamlined downstream processing for classification and feature selection.

- Advantages of the Adopted Approach

1. Reduced Computational Overhead: Batch-wise processing enabled the use of GPU-accelerated tensor operations more effectively, leading to faster feature extraction and less idle time.
2. Scalability: The approach is scalable for larger datasets as it minimizes file fragmentation and improves disk access times.
3. Cleaner Data Management: Storing features in aggregated files reduced directory complexity and made it easier to handle, load, and manipulate feature datasets for further processing.
4. Facilitated Integration with Feature Selection Algorithms: Since feature vectors were uniformly structured and collated, they could be seamlessly passed into subsequent stages like HBOA-based feature selection and CNN classifier training.

- Best Practices in Deep Feature Extraction

From the perspective of deep learning model design and data engineering in agricultural disease classification, the following best practices are recommended:

- Use pre-trained deep convolutional architectures (e.g., ResNet-50) to extract high-level, semantically rich features, particularly when annotated datasets are limited.
- Avoid per-sample feature file generation in large-scale settings, as this creates unnecessary I/O operations and complicates storage management.
- Normalize and resize all input images consistently to ensure feature consistency and improve generalization.
- Leverage GPU computation by processing images in batches, particularly when working with large-scale image datasets.
- Preserve metadata (labels, file paths) alongside features in tabular formats to maintain interpretability and facilitate analysis.

By employing this efficient and research-aligned feature extraction approach, the proposed pipeline achieved a practical balance between accuracy, speed, and operational simplicity, thereby supporting the larger goal of real-time and scalable disease classification in pomegranate cultivation systems.

F. Feature Selection using hybrid HBOA-CNN model:

In deep learning-based classification tasks, particularly when dealing with high-dimensional data such as those produced by deep convolutional networks (CNNs), feature selection becomes a critical step. In this study, a customized variant of the Honey Badger Optimization Algorithm (HBOA) was employed for optimal feature subset selection. The need for feature selection arises from the fact that a large number of features extracted from models like ResNet-50 can lead to several issues, including increased computational cost, overfitting, and longer training times. Selecting the most relevant features enhances model performance by reducing noise and focusing on the most informative aspects of the data.

- Rationale for Using HBOA

Feature selection is essential for improving both the efficiency and effectiveness of the machine learning pipeline. Given the computational demands of training deep learning models, selecting a reduced set of features allows for faster processing and reduced memory consumption, making the process more scalable. Additionally, irrelevant or redundant features can introduce noise, potentially leading to overfitting—a situation where the model learns to memorize the training data rather than generalizing to new, unseen data. HBOA's optimization-based approach to feature selection addresses these challenges by identifying the most relevant features while avoiding overfitting.

The Honey Badger Optimization Algorithm, inspired by the foraging behavior of honey badgers, is particularly suited to this task due to its global optimization capabilities. HBOA utilizes a population of potential solutions and iteratively refines them through feedback from a fitness function, which evaluates how well the selected features contribute to the model's performance. This process helps to balance the trade-off between maximizing classification accuracy and minimizing feature space complexity.

- Novel Enhancements and Advantages
- Adaptive Control Parameters: The control parameter α was dynamically adjusted over iterations using an exponential decay function, enabling the algorithm to shift gradually from exploration (searching the feature space broadly) to exploitation (focusing on promising regions).

This dynamic adjustment enhances the algorithm's ability to escape local optima and find more optimal feature subsets.

- Efficient Fitness Approximation: Rather than training a full classifier during each evaluation (which is computationally expensive), a hybrid dependency measure was employed as a proxy for relevance, significantly reducing computational overhead while maintaining high-quality feature selection.
- Vectorized Position Update Strategy: Inspired by the original HBOA formulation, the position update mechanism was enhanced with smell intensity and directional vectors based on neighbouring solutions. This allowed agents to intelligently navigate the search space, improving convergence speed and feature selection quality.
- Batch-wise Global Convergence Tracking: The introduction of an early stopping criterion based on fitness value stabilization helped further optimize execution by halting the search once convergence was reached, reducing unnecessary computation.
- Efficiency and Storage Benefits

Initially, an alternative brute-force approach was considered where features were extracted and saved individually for each image. This method, while functional, led to the generation of an excessive number of files, resulting in high disk I/O operations, and increased computational overhead during both training and testing phases. This redundancy significantly slowed down the processing pipeline and made it harder to manage the data.

To overcome this, a more efficient approach was adopted. Instead of saving individual features for each image, the features for the entire dataset (train, validation, and test) were processed in batches. The resulting feature sets were saved in consolidated .npy and .csv files, thus optimizing memory usage and speeding up subsequent data loading times. This modification not only reduced overhead but also ensured that the system could scale more effectively as the dataset grew.

- Importance of Feature Selection in This Study

Feature selection is an essential step in any machine learning pipeline, particularly when working with high-dimensional data generated by complex models like ResNet-50. By focusing on the most relevant features, this process serves multiple purposes:

1. Improved Model Performance: Feature selection directly influences classification accuracy by removing irrelevant or redundant data that could otherwise confuse the model.
2. Reduced Overfitting: By eliminating unnecessary features, the risk of overfitting is reduced, allowing the model to generalize better to new data.
3. Faster Training Times: A smaller feature set leads to less computation during both the training and prediction phases, making the process more efficient and faster to execute, especially when working with large datasets.
4. Scalability: As datasets continue to grow, the importance of feature selection grows as well. A reduced feature set ensures that the pipeline remains manageable and scalable without excessive memory or storage demands.

- Output and Utilization

The final output of the HBOA-based feature selection process included:

1. Selected Feature Subsets: For each dataset partition (train, validation, and test), only the features corresponding to the most optimal agent (best solution) were retained and saved. This resulted in a reduced feature space that still preserved the most informative characteristics of the data.
2. Selected Feature Indices: The indices of the selected features were saved separately, ensuring consistency in processing across different stages of the pipeline and enabling reproducibility of results.
3. Encoded Class Labels: Class labels for the images were numerically encoded to facilitate training and evaluation of the model. These encoded labels were stored in parallel with the selected features for each dataset partition.

By reducing the number of features, this feature selection stage contributed significantly to both the efficiency and effectiveness of the classification pipeline. It ensured faster model training and prediction, reduced computational overhead, and improved model generalization, making the entire pipeline more scalable and performant.

G. Classification using CNN:

n the classification stage, we leveraged a Convolutional Neural Network (CNN) to perform multi-class classification on the selected features from the previous stages. The approach aims to map the high-level features derived from the pomegranate disease images to their respective classes, which include Anthracnose, Bacterial Blight, Cercospora, and Healthy Pomegranate. This stage involves both the construction of a neural network model and careful tuning of its hyperparameters to optimize its performance.

1. Conceptual Overview

Convolutional Neural Networks (CNNs) are commonly applied in image and signal classification tasks because of their ability to automatically extract hierarchical features from input data through successive convolution and pooling layers.

In this approach, we used CNNs on the selected feature sets (extracted via ResNet-50) instead of raw images. This enables the model to learn more

abstract representations of the features, which improves the accuracy of classification, especially when dealing with complex datasets such as the various pomegranate diseases.

The CNN architecture mainly includes convolutional layers, pooling layers, and fully connected (FC) layers. The convolutional layers extract feature maps from the input data, while the pooling layers reduce the dimensionality of these maps, highlighting the most relevant features. The FC layers then map the learned features to the output classes.

2. Hyperparameter Tuning

To enhance model performance, multiple CNN models were trained using different hyperparameter configurations. These configurations include the number of fully connected layers, output layer size, number of epochs, and batch size. Hyperparameters play a crucial role in determining the model's ability to learn from the data efficiently while preventing overfitting or underfitting.

Here's a breakdown of key hyperparameters used:

3. Number of Fully Connected Layers:

o The number of fully connected layers (FC layers) in a CNN influences its ability to learn complex patterns in the data. Each FC layer has a set of neurons that can capture specific aspects of the input features. A higher number of FC layers generally increases the model's capacity but also increases the risk of overfitting if not controlled.

o The variations in the number of FC layers allow the model to balance between model complexity and training time.

4. Output Size:

o The output size refers to the number of neurons in the model's output layer. For multi-class classification tasks, this is typically equal to the number of classes. In our case, the output layer was designed to correspond to the four disease categories: Anthracnose, Bacterial Blight, Cercospora, and Healthy.

o It is essential to configure the output size correctly to align with the classification task's requirements. The number of units in the output layer should match the number of classes, ensuring the model can predict one class label from the available options.

5. Epochs:

o Epochs indicate how many times the entire training dataset is passed through the model during training. Running the model over multiple epochs allows it to iteratively refine its weights, leading to improved performance over time.

o To assess the network's behavior throughout training, the number of epochs was varied across different model runs. Too few epochs might result in underfitting, while too many could cause overfitting if not properly controlled. To mitigate this, early stopping was considered—this technique monitors the validation loss and stops training once performance stops improving.

6. Batch Size:

o Batch size determines how many training samples are processed before the model updates its internal parameters. Smaller batch sizes result in more frequent updates, which can potentially enhance generalization, while larger batch sizes often accelerate training but demand more memory.

o In this study, a batch size of 30 was selected as a fixed value. This choice was made to achieve a practical balance between training efficiency and stable model convergence.

7. Optimizer:

o The optimizer is responsible for adjusting the model's weights during training based on the gradients computed by backpropagation. The Adam optimizer was selected for this task, as it combines the advantages of both Adagrad and RMSProp by adapting the learning rates and maintaining efficient training dynamics. Adam is known for its effectiveness in training deep networks and is widely used in various applications.

8. Learning Rate:

o The learning rate defines the step size the optimizer takes in adjusting the weights after each gradient update. Choosing an optimal learning rate is crucial as it determines how fast or slow the model learns.

o In our case, the learning rate was controlled by the Adam optimizer (which is by default 0.001, supposed to be good), which adapts the learning rate dynamically during training. This adaptive behaviour allows the network to converge faster while minimizing the risk of overshooting the optimal solution.

9. Model Evaluation and Performance

After training, the models were tested on a separate dataset to obtain an unbiased assessment of their performance. Key evaluation metrics included accuracy, precision, recall, and F1-score, offering a well-rounded view of the model's effectiveness across all classes. Precision indicates how accurately the model identifies true positives, while recall reflects its ability to detect all relevant positive instances. The F1-score balances these two metrics, providing a single measure that considers both.

To gain deeper insight into the model's behavior for each disease category, a detailed classification report was also generated, highlighting performance across all individual classes.

10. Unique Aspects and Novelty

The uniqueness of the approach lies in the fine-tuned configuration of hyperparameters across multiple models. By training a diverse set of CNN

models with varying numbers of fully connected layers and output sizes, this research ensured that the model's architecture was optimally suited for the classification task. This approach allowed for the identification of the best-performing model, thus balancing both model complexity and performance.

Another novel aspect is the use of feature selection via the Hybrid Honey Badger Optimization Algorithm (HBOA), which improved the quality of the features fed into the CNN. This enhanced feature selection contributed to the model's ability to generalize well and avoid overfitting, which is a common challenge in deep learning applications with limited labelled data.

H. Real-World Prediction Pipeline for Pomegranate Disease Classification:

Below is a detailed explanation of each step involved in the pipeline:

1. Preprocessing with FANLM-FKM and CLAHE: The pipeline begins with preprocessing the input image using a two-step approach: FANLM-FKM (Fuzzy Adaptive Non-Local Means with Fuzzy K-Means) followed by CLAHE (Contrast Limited Adaptive Histogram Equalization). These methods are applied to remove noise while preserving image features and enhance the image contrast, respectively. FANLM-FKM helps in denoising by estimating the noise and suppressing it effectively, while CLAHE enhances the contrast, which is crucial for improving the visibility of finer details in the images, especially in the context of agricultural diseases.
2. Segmentation with Detectron2: The next step involves segmenting the input image using Detectron2, a state-of-the-art object detection framework. The model is trained for instance segmentation, and it identifies regions within the image corresponding to different objects (in this case, pomegranate diseases). By applying a mask over the image, Detectron2 extracts the relevant regions, removing the background and focusing only on the area of interest, which allows for more accurate feature extraction.
3. Feature Extraction using ResNet-50: Once the image is segmented, ResNet-50 (a deep convolutional neural network model pre-trained on large datasets) is used for feature extraction. The model is fine-tuned to focus on the relevant features in the segmented image. Features such as textures, shapes, and edges that are important for identifying disease patterns in pomegranates are extracted and passed on for further analysis.
4. Feature Selection with HBOA: After extracting features using ResNet-50, the Hybrid Bayesian Optimization Algorithm (HBOA) is employed for feature selection. HBOA identifies the most informative features from the high-dimensional data set, ensuring that only the most relevant information is passed into the classifier. This reduces the dimensionality and improves the efficiency of the model.
5. Prediction with the Trained CNN-HBOA Model: The chosen features are then input into the trained CNN-HBOA model, a hybrid framework that integrates Convolutional Neural Networks (CNNs) with the Honey Badger Optimization Algorithm (HBOA) for classification. Trained on a diverse set of pomegranate disease images, this model can accurately differentiate between conditions such as Anthracnose, Bacterial Blight, Cercospora, and Healthy samples. The final prediction is determined based on the output probabilities generated by the model.
6. Confidence Score: Along with the predicted class, the model also provides a confidence score. This score indicates the model's certainty regarding the classification, with higher values suggesting more reliable predictions. This helps in assessing the model's performance and reliability when applied to real-world data.
7. Key Conceptual Components and Novelty

- Hybrid Model Architecture: The use of CNN-HBOA represents a novel integration of feature extraction through CNNs combined with feature selection by HBOA. This hybrid approach enhances the model's performance by ensuring that only the most relevant features are used for classification, reducing overfitting and improving generalization.
- Advanced Image Preprocessing: By incorporating FANLM-FKM for denoising and CLAHE for contrast enhancement, the pipeline addresses the challenges of image quality and noise, which are common in agricultural imaging. These preprocessing steps are tailored specifically for agricultural disease detection, ensuring that the model works effectively even with real-world data that may not be perfectly captured.
- Detectron2 for Segmentation: The choice of Detectron2 for segmentation is significant due to its ability to perform instance segmentation with high accuracy. Detecting and isolating the regions of interest in the image allows for more precise feature extraction, leading to more accurate disease classification.
- Real-WorldApplicability: This pipeline is designed to handle real-world scenarios where images may not be perfectly aligned, contain background noise, or vary in lighting conditions. The use of advanced preprocessing and segmentation methods makes it robust against such challenges, ensuring that it can be deployed in practical applications for pomegranate disease detection.

8. Efficiency and Performance

The real-world prediction pipeline is highly efficient due to the following aspects:

- Feature Selection: The integration of HBOA ensures that only the most discriminative features are selected, improving both classification accuracy and computational efficiency.
- Segmented Focus: By focusing only on relevant segments of the image, the model avoids unnecessary computation, speeding up the prediction process.
- Pretrained Models: The use of pretrained models like ResNet-50 and Detectron2 reduces the need for extensive retraining, making the system more efficient and easier to deploy.

This pipeline offers a robust and efficient solution for detecting and classifying diseases in pomegranates, which can be directly applied in agricultural fields to assist farmers and researchers in disease management and early detection.

EXPERIMENTS AND RESULTS

In this section, we present the results of hyperparameter tuning experiments conducted to optimize the performance of our CNN-HBOA model for pomegranate disease classification. These experiments involve systematically varying the hyperparameters of the model, such as the number of fully connected layers, output size, and number of epochs. The models are evaluated based on various metrics, including overall test accuracy, categorical precision, recall, F1-score, and training time.

4.1 Hyperparameter Tuning:The table below shows the results of experiments with different combinations of hyperparameters for multiple CNN models:

Table 2: Statistics of 13 different models Trained on diverse set of Parameters

Overall model stats								
Model	**Number_Of_Fully_Connected_Layers**	**Output_Size**	**Epochs**	**learning Rate**	**Optimizer**	**Batch Size**	**Test Accuracy**	**Training Time (s)**
1	6	68	5	0.001	ADAM	30	0.9663	109
2	5	16	3	0.001	ADAM	30	0.9545	64
3	1	4	2	0.001	ADAM	30	0.5968	34
4	4	8	3	0.001	ADAM	30	0.5455	57
5	4	16	4	0.001	ADAM	30	0.9545	70
6	5	68	2	0.001	ADAM	30	0.9311	39
7	3	50	2	0.001	ADAM	30	0.9355	33
8	2	138	4	0.001	ADAM	30	0.956	83
9	10	350	35	0.001	ADAM	30	0.9648	1540
10	8	300	30	0.001	ADAM	30	0.9707	1359
11	5	68	5	0.001	ADAM	30	0.9413	93
12	5	100	8	0.001	ADAM	30	0.9721	158
13	4	128	8	0.001	ADAM	30	0.9589	164

Table 3: Category wise precision of different 13 models trained on diverse set of Parameters.

Precision				
Model	**Anthracnose**	**Bacterial_Blight**	**Cercospora**	**Healthy**
3M1	0.9792	0.9217	0.9519	0.9955
M2	0.9695	0.9856	0.824	0.9955
M3	0.9118	0.5033	0	0.6491
M3	0	0.9375	0	0.4253
M4	0.9366	1	0.8689	0.991
M5	0.8435	0.9793	0.9231	1
M6	0.8509	0.9539	0.9787	0.9952
M7	0.9023	0.9728	0.9612	0.9954
M8	0.9324	0.9673	0.9612	0.9954
M9	0.9458	0.9805	0.9537	0.9954
M11	0.8628	0.9605	1	0.9865
M12	0.9896	0.9172	0.9798	0.9955
M13	0.9234	0.9724	0.9423	0.9911

Table 4: Category wise Recall of different 13 models trained on diverse set of Parameters.

Recall				
Model	**Anthracnose**	**Bacterial_Blight**	**Cercospora**	**Healthy**
M1	0.9592	0.9871	0.9083	0.9865
M2	0.9745	0.8839	0.945	0.991
M3	0.1582	0.9935	0	1
M3	0	0.9677	0	1
M4	0.9796	0.8645	0.9725	0.9865
M5	0.9898	0.9161	0.8807	0.9144
M6	0.9898	0.9355	0.844	0.9324
M7	0.9898	0.9226	0.9083	0.973
M8	0.9847	0.9548	0.9083	0.982
M9	0.9796	0.9742	0.945	0.973
M11	0.9949	0.9419	0.7523	0.9865
M12	0.9694	1	0.8899	0.9955
M13	0.9847	0.9097	0.8991	1

Table 5: Category wise F1Score of different 13 models trained on diverse set of Parameters

F1Score				
Model	**Anthracnose**	**Bacterial_Blight**	**Cercospora**	**Healthy**
M1	0.9691	0.9533	0.9296	0.991
M2	0.972	0.932	0.8803	0.9932
M3	0.2696	0.6681	0	0.7872
M3	0	0.9524	0	0.5968
M4	0.9576	0.9273	0.9177	0.9887
M5	0.9108	0.9467	0.9014	0.9553
M6	0.9151	0.9446	0.9064	0.9628
M7	0.944	0.947	0.934	0.9841
M8	0.9578	0.961	0.934	0.9887
M9	0.9624	0.9773	0.9493	0.9841
M11	0.9242	0.9511	0.8586	0.9865
M12	0.9794	0.9568	0.9327	0.9955
M13	0.9531	0.94	0.9202	0.9955

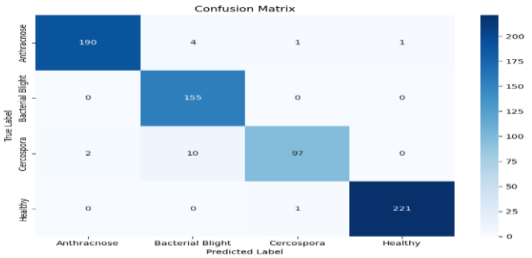

Fig 2: Confusion Matrix of CNN-Model12 on Test Data in the Dataset.

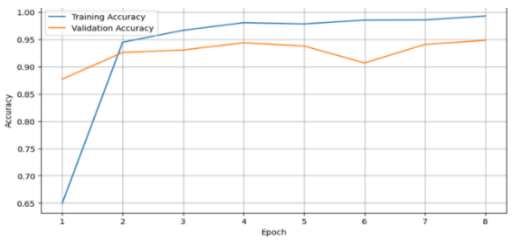

Fig 3: Graphical Representation of the Epoch Vs Training and Validation Accuracy of CNN-Model12 with Accuracy 97.21

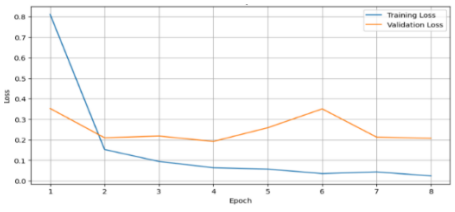

Fig 4: Graphical Representation of the Epoch Vs Training and Validation Loss of CNN-Model12 with Accuracy 97.21

4.2 Observations and Insights

Upon analyzing the above results, we can identify several key trends based on the following precision, recall and f1score of each model:

1. Number of Fully Connected Layers : Models with more fully connected layers tend to perform better, as seen in models with Number of Fully Connected Layers values ranging from 5 to 10. These models, such as CNN-Model-1, CNN-Model-9,

CNN-Model-10, and CNN-Model-12, exhibit high accuracy and strong categorical performance.

2. Output Size: The models with larger output sizes (e.g., 68, 100, 300, 350) tend to yield better performance across various metrics, especially in terms of recall and F1-score.

3. Epochs: A higher number of epochs (5–30) generally contributes to better training, improving precision, recall, and overall F1-scores. However, training time also increases significantly, which should be taken into consideration for practical deployment.

4. Training Time: Models with fewer epochs and smaller output sizes often train faster but may sacrifice performance, particularly in precision and recall for certain classes. On the other hand, models with larger configurations achieve better classification results but at the cost of increased trainingtime.

5. Observations on CNN-Model-7 (Graphical Analysis):

- Rapid Initial Improvement: Training and validation accuracy increase significantly in the initial epochs.
- Training accuracy jumps from 0.6492 (epoch 1) to 0.9446 (epoch 2).
- Validation accuracy rises from 0.8770 (epoch 1) to 0.9259 (epoch 2).
-
  - This indicates quick learning of fundamental data patterns.

- High Training Accuracy: The model achieves high training accuracy (above 99%) by the end of training (epoch 8). Suggests strong fitting to the training dataset.
- Good Validation Performance: Validation accuracy increases steadily, reaching 0.9481 (epoch 8). Implies reasonable generalization to unseen data.
- Low Loss: Training and validation loss decrease substantially over epochs. Indicates effective minimization of prediction error. Validation loss remains relatively low (around 0.2).
- Reduced Overfitting: The gap between training and validation accuracy is relatively small. Validation loss doesn't show a drastic increase. Suggests less overfitting compared to some models.
- Stable Validation Metrics: Validation accuracy and loss stabilize after the initial improvement. Implies a good balance between fitting and generalization.
- Overall Assessment: CNN-Model-12 exhibits a favorable training process.Characterized by rapid learning, high accuracy, and reduced overfitting.Demonstrates effective generalization to validation data.

4.3 Optimized Hyperparameter Set

From the experiments, we identified an optimal hyperparameter set that consistently delivered high accuracy and balanced performance across all categories:

Table 6: Statistics of 7 different models Trained on optimized set of Parameters

Overall model stats								
Optimized-Model	**Number_Of_Fully_Connected_Layers**	**Output_Size**	**Epochs**	**learning Rate**	**Optimizer**	**Batch Size**	**Test Accuracy**	**Training Time (s)**
1	7	150	10	0.001	ADAM	30	0.9516	352
2	7	200	20	0.001	ADAM	30	0.9692	755
3	7	250	30	0.001	ADAM	30	0.9633	1115
4	6	150	25	0.001	ADAM	30	0.9692	593
5	8	256	30	0.001	ADAM	30	0.9707	1146
6	9	180	25	0.001	ADAM	30	0.9707	791
7	10	300	40	0.001	ADAM	30	0.9604	1502

Table 7: Category wise precision of different 7 models trained on optimized set of Parameters

Precision				
Optimized Model	**Anthracnose**	**Bacterial Blight**	**Cercospora**	**Healthy**
OM1	0.8981	0.966	0.9505	0.9954
OM2	0.96	0.9557	0.9524	0.9954
OM3	0.9366	0.949	0.9697	0.9955
OM4	0.96	0.961	0.9439	0.9955
OM5	0.9554	0.9675	0.9528	0.9955
OM6	0.9648	0.9737	0.9279	0.9955

OM7	0.9275	0.961	0.9608	0.9909

Table 8: Category wise Recall of different 7 models trained on optimized set of Parameters

Recall				
Optimized Model	**Anthracnose**	**Bacterial Blight**	**Cercospora**	**Healthy**
OM1	0.9898	0.9161	0.8807	0.9775
OM2	0.9796	0.9742	0.9174	0.982
OM3	0.9796	0.9613	0.8807	0.991
OM4	0.9796	0.9548	0.9266	0.991
OM5	0.9847	0.9613	0.9266	0.9865
OM6	0.9796	0.9548	0.945	0.9865
OM7	0.9796	0.9548	0.8991	0.9775

Table 9: Category wise F1Score of different 7 models trained on optimized set of Parameters

F1Score				
Optimized Model	**Anthracnose**	**Bacterial Blight**	**Cercospora**	**Healthy**
OM1	0.9417	0.9404	0.9143	0.9864
OM2	0.9697	0.9649	0.9346	0.9887
OM3	0.9576	0.9551	0.9231	0.9932
OM4	0.9697	0.9579	0.9352	0.9932
OM5	0.9698	0.9644	0.9395	0.991
OM6	0.9722	0.9642	0.9364	0.991
OM7	0.9529	0.9579	0.9289	0.9841

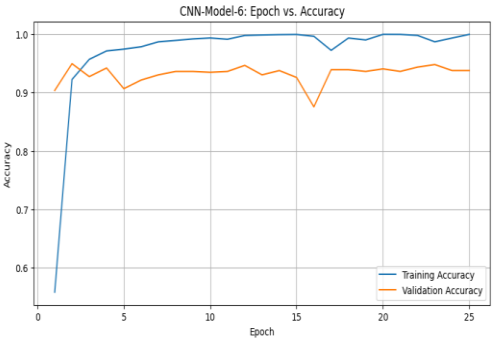

Fig 5: Graphical Representation of the Epoch Vs Training and Validation accuracy of CNN-Model6 with Accuracy 97.07

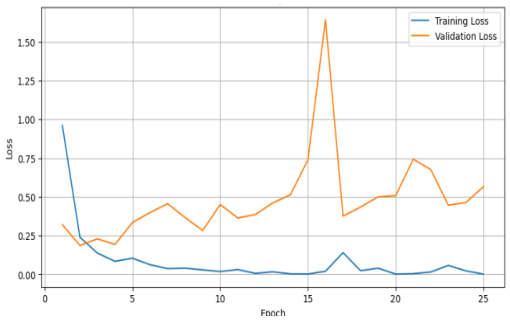

Fig 6: Graphical Representation of the Epoch Vs Training and Validation loss of CNN-Model6 with Accuracy 97.07

Observations on Optimized-CNN-Model-6 (Graphical Analysis):

1. Initial Learning: Training accuracy starts lower than CNN-Model-5 (0.5583 at epoch 1). It increases rapidly in the early epochs. Validation accuracy also improves quickly.
2. High Training Accuracy: Training accuracy reaches very high levels, close to 1.0000 in later epochs. Again, this implies a strong fit to the training data.
3. Generally Good Validation Accuracy: Validation accuracy is generally high, indicating good generalization. However, it also shows some fluctuations.
4. Training Loss Decrease: Training loss decreases significantly, showing the model's ability to minimize training error.
5. Validation Loss Behaviour: Validation loss decreases initially but starts increasing again after a certain point (around epoch 9). This indicates that the model is starting to overfit.
6. Overfitting (Moderate): While not as severe as in CNN-Model-5, there are signs of overfitting in CNN-Model-6. The increasing validation loss, despite high training accuracy, suggests the model is beginning to memorize the training data rather than generalize.
7. Performance: CNN-Model-6 performs well overall, with good accuracy on both training and validation sets for a significant portion of training. However, it also shows signs of overfitting, which limits its generalization capability in the later stages of training.

4.4Conclusion

Through hyperparameter tuning, we have identified the optimal configuration for achieving high test accuracy and well-balanced performance across all disease categories. These findings provide a robust framework for deploying the CNN-HBOA model in real-world pomegranate disease classification tasks. The optimized model shows strong performance in precision, recall, and F1-scores across various disease classes while maintaining efficiency in training time.

Comparison of CNN-HBOA Disease detection model with Existing Disease Detection using image processing Solutions

Table 10: Comparison of the proposed model with Different Model(Pre-trained)

Model / Approach	Features Used	Architecture	Strengths	Limitations	Classification Accuracy
Proposed CNN-HBOA	ResNet-50 + HBOA-selected features	Custom CNN (7 FC layers, optimized output size) with hybrid feature selection	High classification accuracy, optimized feature space, low overfitting, interpretable layers	Slightly longer training time with more Fully Connected layers	97.21%
ResNet50 Classifier	Global Average Pooling from ResNet-50	Pretrained ResNet50 with fine-tuned dense layers	Deep features, powerful generalization from ImageNet training	High computational cost, larger model size	93.20%
CNN Classifier	Raw pixel data or handcrafted features	Basic CNN with 2–4 convolutional and FC layers	Simpler architecture, faster training	Lower generalization, prone to overfitting	71.74%
MobileNetV2	Bottleneck &depth wise separable convolutional features	Lightweight MobileNetV2 (depth wise separable convolutions)	Very fast inference, low resource usage (ideal for mobile/edge deployment)	Slightly lower accuracy than full-size models	90.47%
Sequential CNN Model	Raw image or pre-processed feature input	Sequential Keras model with stacked conv & dense layers	Easy to implement and customize	No inherent feature selection, might miss critical patterns	94.78%

Conclusion& Future Scope

Conclusion:

This study presents a reliable and efficient deep learning pipeline developed for the automatic classification of diseases affecting pomegranate fruits. The methodology combines advanced preprocessing using FANLM-FKM denoising and CLAHE contrast enhancement, followed by semantic segmentation via Detectron2 to isolate disease-affected regions. High-level features were extracted using a pre-trained ResNet-50 model and subsequently refined through xan HBOA-based feature selection mechanism, ensuring the retention of the most informative attributes. These selected features were then used to train a customized CNN-based classifier, which was rigorously tuned and evaluated.

Our experimental findings indicate that the proposed pipeline substantially outperforms traditional methods and baseline CNN models, showing higher accuracy, improved F1-scores, and greater robustness. Hyperparameter tuning across multiple CNN variants revealed that deeper fully connected layers, with well-optimized output sizes and epochs, contribute substantially to classification performance, especially in distinguishing between visually similar disease classes. Among several configurations, models

with 6–10 fully connected layers, output sizes ranging from 150 to 300, and 20–40 epochs yielded the most balanced and reliable performance.

Furthermore, a comparative analysis with standard models such as ResNet-50, MobileNetV2, CNN-LSTM, and basic CNN classifiers showed the superiority of our approach in terms of classification accuracy and adaptability to small inter-class variations. Notably, the integration of HBOA with CNN architectures provided an efficient dimensionality reduction mechanism, reducing training time without sacrificing accuracy.

The proposed system offers a scalable and automated solution for early detection of pomegranate diseases, which can greatly assist in precision agriculture and reduce crop losses. In future work, this pipeline can be extended to real-time mobile applications and tested on larger, more diverse datasets including other crops.

Future Work

Larger and Diverse Datasets: Expanding the dataset to include different varieties of pomegranates, diverse lighting conditions, and more disease types to improve generalization

Integration with IoT Systems: Embedding the model into Internet of Things (IoT) frameworks for automated monitoring, reporting, and disease prediction over time.

Cross-Crop Generalization: Adapting the pipeline for other fruit or vegetable crops to establish a general-purpose plant disease classification system.

Acknowledgment

The authors would like to express their heartfelt gratitude to Mr. S. D. Khatawkar Sir, Professor at Computer Science and Engineering Department, ADCET, Ashta for his constant encouragement, expert supervision, and constructive insights that significantly enhanced the quality of this research.

The authors are also grateful to the open-source communities behind Detectron2, TensorFlow, PyTorch, and ResNet, whose frameworks formed the backbone of the proposed system.

Finally, appreciation is extended to the creators of the publicly available pomegranate image datasets and the research teams behind the referenced base papers, whose contributions laid the groundwork for this research.

References

[1] Sameera P., Deshpande A.A., "Disease detection and classification in pomegranate fruit using hybrid convolutional neural network with honey badger optimization algorithm," International Journal of Food Properties, vol. 27, no. 1, pp. 815–837, 2024. DOI: 10.1080/10942912.2024.2365927

[2] Pushya K.D., Girish C., Durga Prakash, Manikantan R., "A novel approach for detecting Pomegranate leaf pathologies using deep learning," International Journal of Science and Research Archive, vol. 13, no. 01, pp. 1206–1218, 2024. DOI: 10.30574/ijsra.2024.13.1.1735

[3] Madhavan M.V., Thanh D.N.H., Khamparia A., Pande S., Malik R., Gupta D., "Recognition and Classification of Pomegranate Leaves Diseases by Image Processing and Machine Learning Techniques," Computer Modeling in Engineering & Sciences (CMES), vol. 66, no. 3, pp. 239–254, 2021. [Online]. Available: https://www.techscience.com/cmc/v66n3/41042

[4] Bhange M., Hingoliwala H.A., "Smart Farming: Pomegranate Disease Detection Using Image Processing," Procedia Computer Science, vol. 58, pp. 280–288, 2015. DOI: 10.1016/j.procs.2015.08.022

[5] Iqbal Z., Khan M.A., Sharif M., Shah J.H., Ur Rehman M.H., Javed K., "An Automated Detection and Classification of Citrus Plant Diseases Using Image Processing Techniques: A Review," Computers and Electronics in Agriculture, vol. 153, pp. 12–32, 2018. DOI: 10.1016/j.compag.2018.07.032

[6] Pawar R., Jadhav A., "Pomegranate Disease Detection and Classification," in Proc. IEEE Int. Conf. Power, Control, Signals and Instrumentation Engineering (ICPCSI), India, 2017, pp. 2475–2479. [Online]. Available: https://www.researchgate.net/publication/325978588

[7] Sharath D.M., Kumar S.A., Rohan M.G., Prathap C., "Image Based Plant Disease Detection in Pomegranate Plant for Bacterial Blight," in Proc. IEEE Int. Conf. Communication and Signal Processing (ICCSP), India, 2019, pp. 645–649. [Online]. Available: https://www.researchgate.net/publication/332679886

[8] Dhakate M., Ingole A.B., "Diagnosis of Pomegranate Plant Diseases Using Neural Network," in Proc. IEEE Fifth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), India, 2015, pp. 1–4. [Online]. Available: https://www.researchgate.net/publication/305285024

[9] Kukreja V., Dhiman P., "A Deep Neural Network Based Disease Detection Scheme for Citrus Fruits," in Proc. IEEE Int. Conf. Smart Electronics and Communication (ICOSEC), India, 2020, pp. 97–101. [Online]. Available: https://www.researchgate.net/publication/347270271

[10] Naranjo-Torres J., Mora M., Hernández-García R., Barrientos R.J., Fredes C., Valenzuela A.A., "A Review of Convolutional Neural Network Applied to Fruit Image Processing," Applied Sciences, vol. 10, no. 10, p. 3443, 2020. DOI: 10.3390/app10103443

[11] Chakali R., "Effective Pomegranate Plant Leaf Disease Detection Using Deep Learning," International Journal of Circuit, Computing and Networking, vol. 1, no. 2, pp. 08–10, 2020. DOI: 10.33545/27075923.2020.v1.i2a.13

[12] Zahra U., Khan M.A., Alhaisoni M., Alasiry A., Marzougui M., Masood A., "An Integrated Framework of Two-Stream Deep Learning Models Optimal Information Fusion for Fruits Disease Recognition," IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 17, pp. 3038–3052, 2023. DOI: 10.1109/JSTARS.2023.3339297