Discover Artificial Intelligence

RESEARCH Open Access

Deep learning based plant health disease detection in tomatoes using inception v4 convolutional neural network and YOLO V8

B. Sowmya[1*] and S. Guruprasad[2]

*Correspondence:
B. Sowmya
sowmyabnaik@gmail.com
[1]Department of Computer Science and Engineering, Sri Siddhartha Academy of Higher Education, Tumkur 572107, India
[2]Department of Biomedical Engineering, Sri Siddhartha Academy of Higher Education, Tumkur 572107, India

Abstract
Tomato plant diseases significantly impact agricultural productivity, posing a challenge for sustainable crop management. This study proposes a deep learning-based framework for accurate and automated detection of tomato plant diseases using the Inception v4 convolutional neural network (CNN) and YOLOv8 (You Only Look Once, version 8) object detection model. A curated dataset of tomato plant images, encompassing various diseases such as bacterial spot, early blight, late blight, and leaf mold, alongside healthy samples, was developed for training and evaluation. The Inception v4 CNN is employed for feature extraction and classification, while YOLOv8 is utilized for real-time disease detection and localization. Experimental results demonstrate that the combined use of Inception v4 and YOLOv8 achieves a classification accuracy of 96% and a mean Average Precision (mAP@0.5) of 86% for leaf disease detection with precision and recall improving by 5.3% and 4.8%, respectively, compared to existing methods. The proposed model highlights the potential of deep learning techniques to enhance early disease diagnosis, enabling farmers to take timely and effective measures to mitigate crop losses.

Keywords Tomato plant diseases, Inception v4 CNN, YOLOv8, Feature extraction, Plant health monitoring, Agricultural productivity, Real-time localization

1 Introduction

Recent developments in deep learning and computer vision have significantly contributed to advancements in precision agriculture, especially in the field of automated detection of plant diseases. Tomato plants, which hold considerable economic and nutritional value worldwide, are highly vulnerable to various pathogens that can severely impact yield and crop quality. Conventional methods for identifying plant diseases often rely on manual inspection by agricultural experts, which can be time-intensive, inconsistent, and unsuitable for large-scale applications. As a result, researchers have turned to deep learning models, particularly convolutional neural networks (CNNs) and object detection frameworks like YOLO (You Only Look Once), to improve the speed, accuracy, and scalability of disease identification in tomato crops [1, 2].

Among CNN architectures, Inception v4 stands out for its ability to extract detailed and multi-scale features from complex plant imagery. Its design enables the network to capture subtle disease symptoms by utilizing deeper and wider convolutional layers. Techniques such as knowledge-pretrained models and adaptive convolution have also been incorporated to enhance performance across varied environmental conditions [3, 4]. Moreover, lightweight adaptations of CNN models have been proposed to support real-time disease detection on low-resource devices, making it feasible for use directly in the field [5, 6].

Simultaneously, object detection algorithms, particularly those in the YOLO family, have proven effective for pinpointing infected areas on plant leaves. YOLOv8, the latest version, offers superior detection speed and accuracy compared to earlier variants. Custom implementations like YOLOv8-ACCW have achieved promising results in tasks involving grape leaf disease detection, indicating its strong potential for adaptation to tomato disease detection as well [7]. These models are capable of identifying multiple diseased areas within a single leaf image, which is essential for quantifying infection levels and spatial patterns [2, 7, 8].

Researchers have also investigated the use of hybrid deep learning models that integrate CNNs with transformer-based attention modules and multi-scale feature fusion strategies [9–11]. These architectures combine the localized texture analysis strength of CNNs with the long-range dependency modeling of transformers, making them more effective for disease recognition under challenging conditions such as variable lighting and background noise. Additional advancements, such as the integration of capsule networks and residual learning, have further improved accuracy by preserving spatial integrity and enhancing feature representation [12, 13].

To address the challenges posed by limited labelled data and data sensitivity, methods like zero-shot learning and federated learning have been introduced [6, 14]. These approaches enhance model adaptability to unseen disease types and ensure data privacy during training. Parallel efforts to minimize computational load without compromising accuracy have led to the design of compact yet powerful CNN architectures, which support rapid disease detection on mobile and edge devices, a necessity in rural and resource-constrained settings [15, 16].

The proposed focuses on an integrated deep learning approach combining the Inception v4 CNN model for precise classification with YOLOv8 for real-time localization of tomato leaf diseases. The synergy between these two models is designed to provide a high-performing and scalable solution for monitoring tomato crop health. This framework not only facilitates accurate identification but also spatial mapping of diseases, enabling better crop management decisions in the context of modern, technology-driven agriculture [1, 2, 4, 7, 17].

This paper is organized as follows: Sect. 2 reviews related works on plant disease detection using deep learning. Section 3 details the proposed methodology, including dataset preparation, model architecture, and training process. Section 4 presents the experimental results and evaluation metrics, followed by a discussion in Sect. 4. Finally, Sect. 5 concludes the study and outlines future research directions.

2 Related works

The increasing impact of plant diseases on global crop yields has led to the rapid development of AI-powered solutions aimed at ensuring food security and sustainable agriculture. A promising approach is the APDDCM-SHODL system [1], which integrates IoT with deep learning using DenseNet-201 for feature extraction, Smart Hybrid Optimization (SHO) for hyperparameter tuning, and Recurrent Spiking Neural Networks (RSNN) for disease classification. Real-time health monitoring through sensor networks empowers farmers with actionable insights, although scalability in remote areas remains a challenge.

To improve detection accuracy for economically vital crops like tomatoes, an enhanced YOLOv7 framework [2] was introduced with architectural upgrades including focus mechanisms and network module adjustments, achieving 98.8% accuracy. Similarly, a lightweight model for tomato disease detection using AKConv and the EIoU loss function [3] outperformed larger models such as YOLOv9 and Faster R-CNN, offering better real-time applicability with fewer parameters. For grapes, the improved YOLOv8-ACCW [7] model employed coordinate attention mechanisms to enhance small-lesion detection while reducing computational costs.

Beyond object detection models, transformer-based architectures are pushing the boundaries of crop disease classification. The Convolution Self-Guided Transformer (CSGT) [9] improves recognition accuracy through fine-tuning and convolutional modules, while the HOWSVD-TEDA method [4] combines Higher Order Whitened Singular Value Decomposition and Tensor Exponential Discriminant Analysis to extract multi-dimensional features, although real-time efficiency remains a limitation.

Fusion-based models are also gaining momentum. The Dual-Track Swin Transformer and DAMFN architecture [10] leverages global–local feature synergy for accurate detection. Meanwhile, MobileNetV3 outperformed conventional CNNs [5] in controlled environments but struggled in real-world settings, prompting the development of the FL-ToLeD lightweight model [6], which achieved 99.04% accuracy with minimal resources.

Efficiency-focused models like EGWT [11] demonstrated robust performance (99.8% accuracy) with low computational demands, and SE-SK-CapResNet [12] fused Capsule Networks with ResNet to enhance rotational invariance in leaf spot detection. In data-scarce scenarios, zero-shot transfer learning [14] significantly improved disease classification by leveraging knowledge from related domains.

Several other innovative frameworks have emerged: a lightweight 2D CNN [15] for mobile deployment on tomato and cotton leaves; PCA DeepNet [13] for aiding Indian farmers with real-time insights; and compact CNNs designed for drone-based monitoring [18]. Attention-enhanced models using adversarial loss [16, 19] further optimize performance for mobile and UAV deployment, focusing on few-shot learning and early disease detection.

Multi-model comparisons, such as those in [20], revealed that EfficientNetV2_m outperformed ResNet, Inception, and DenseNet variants in apple leaf disease classification. Swinv2-Base achieved 100% accuracy on grape datasets [21], and Res2Next50 reached 99.85% accuracy on tomato leaves [22], highlighting the dominance of advanced CNN and Transformer architectures in fine-grained detection tasks.

For sugarcane, EfficientNet-b6 and InceptionV4 [23] performed best on an 11-class dataset, while Vision Transformers like DeiT3-Small [24] offered promising results (93.79% accuracy), reinforcing their role in high-volume crops. A systematic review [25] of 160 studies underscored the critical role of deep learning in segmentation, classification, and detection, pointing out persistent challenges in model generalization and dataset diversity.

Hybrid approaches continue to refine feature selection. Combining MobileNetV2 and ResNet50V2 with the Gravitational Search Algorithm [26] resulted in 99.2% classification accuracy while reducing feature size by over 50%. Similarly, CNNs enhanced with MultiRes blocks, attention modules, and Dilated Spatial Pyramid Pooling [27] achieved 99.35% accuracy, demonstrating the potential of multi-scale abstraction for fine-grained plant disease classification.

In maize disease detection, MSCPNet [28] integrated a truncated MobileNetV2 with a Multi-Scale Convolutional PoolFormer block to reach 97.44% accuracy on maize and 99.32% on tomato leaf datasets. Its efficiency and real-time suitability make it ideal for deployment in agriculture. A related lightweight depthwise CNN, augmented with SE blocks and improved residual links [29], reached 98% accuracy and 98.2% F1 score, showing strong adaptability even in unstructured real-world settings.

3 Proposed methodology

The detailed discussion of the methodology showed in Fig. 1 is

1. Image Acquisition of Tomato Leaves
   - The process begins with collecting images of tomato leaves using cameras or other imaging devices.
   - These images capture healthy and diseased leaves, forming the initial dataset.
2. Creating the Database
   - The acquired images are stored in a structured database.

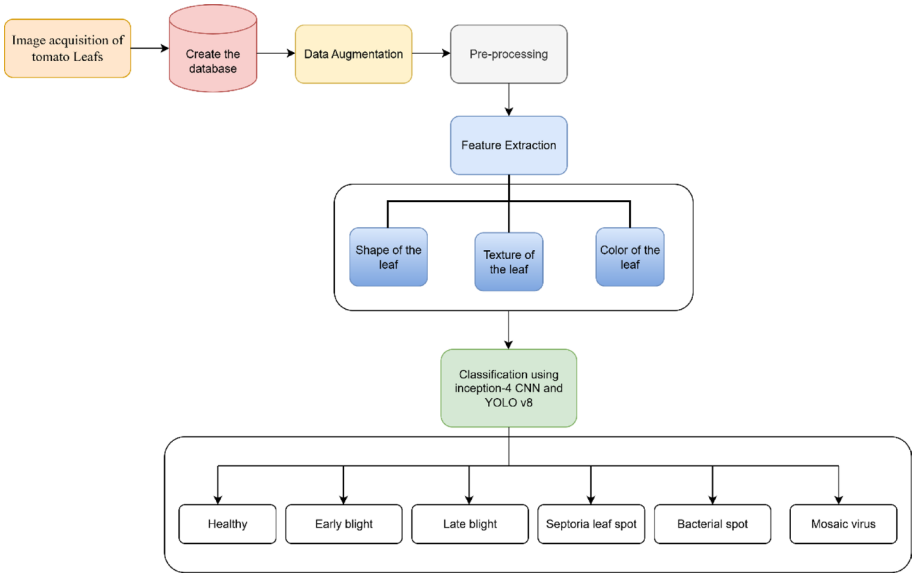

Fig. 1 Overall architecture of the proposed work

- The database includes labeled images for different classes (e.g., healthy, early blight, late blight, bacterial spot, and Septoria leaf spot).
- This step ensures that the dataset is well-organized for further processing.

3. Data Augmentation
   - Since deep learning models require large amounts of data, augmentation techniques are applied to expand the dataset.
   - Augmentation techniques may include:
     - Rotation
     - Flipping
     - Cropping
     - Brightness adjustments
     - Noise addition
   - These transformations improve the model's generalization and robustness.
4. Pre-Processing
   - Image pre-processing is performed to enhance quality and remove noise.
   - Pre-processing steps may include:
     - Resizing images to a fixed dimension suitable for deep learning models.
     - Normalization to scale pixel values between 0 and 1.
     - Contrast enhancement to highlight disease features.
5. Feature Extraction
   - Extracting relevant features from tomato leaves is crucial for accurate classification.
   - Features are categorized into:
     - Shape of the leaf—structural patterns and contours.
     - Texture of the leaf—surface irregularities and disease-induced patterns.
6. Classification Using Inception v4 CNN and YOLO v8
   - The extracted features are used for classification through Deep Learning models:
     - Inception v4 CNN (Convolutional Neural Network):
       - A deep learning architecture optimized for image classification.
       - Efficient in learning complex features from images.
     - YOLO v8 (You Only Look Once version 8):
       - A real-time object detection model.
       - Detects multiple disease symptoms in a single image with high speed and accuracy.
7. Disease Classification
   - The deep learning models classify tomato leaves into one of five categories:
     - Healthy—No signs of disease.

- Early Blight—Circular dark spots with a yellow halo.
- Late Blight—Irregular grayish lesions leading to leaf decay.
- Septoria Leaf Spot—Small brown spots with dark edges.
- Bacterial Spot—Watery lesions that turn black and necrotic.

The Inception-v4 CNN is a convolutional neural network architecture developed by Google as part of their Inception family. It's known for its highly optimized architecture that combines depth, width, and multiple filter sizes in parallel for efficient feature extraction.

1. Basic Convolutional Layer (Conv Layer)
   The Eq. (1), explains Each convolutional layer performs:

$$\mathrm{Y_{i,j,k}} = \mathrm{f}\left(\sum_{\mathrm{m=1}}^{\mathrm{M}}\sum_{\mathrm{u=1}}^{\mathrm{U}}\sum_{\mathrm{v=1}}^{\mathrm{V}}\omega_{\mathrm{v,u,m,k}}\cdot \mathrm{X_{i+u,j+v,m}} + \mathrm{b_k}\right) \tag{1}$$

   where X—input tensor, ω—convolutional filters, b_k—bias of kth filter, f—ReLU activation function, Y—Output feature map

2. Inception module
   The Inception block uses parallel convolutional paths, each with different kernel sizes (1×1, 3×3, 5×5, and pooling). This allows the model to capture multi-scale spatial features.
   Inceptions block mathematical formulation:
   If the input is X, then the output is:

$$\mathrm{Y} = \mathrm{Concat}\left(\mathrm{f_{1\times1}}\left(\mathrm{X}\right), \mathrm{f_{3\times3}}\left(\mathrm{X}\right), \mathrm{f_{5\times5}}\left(\mathrm{X}\right), \mathrm{f_{pool}}\left(\mathrm{X}\right)\right) \tag{2}$$

   Each $\mathrm{f_{k\times k}}\left(\mathrm{X}\right)$ represents a separate convolutional branch.

3. Dimensionality Reduction using 1×1 Convolutions
   1×1 convolutions reduce computation:

$$\mathrm{Y_{i,j,k}} = \mathrm{f}\left(\sum_{\mathrm{m=1}}^{\mathrm{M}}\omega_{\mathrm{1,1,m,k}}\cdot \mathrm{X_{i,j,m}} + \mathrm{b_k}\right) \tag{3}$$

   This is equivalent to a linear projection from input depth M to output depth K, reducing the number of parameters.

4. Auxiliary Classifier (used during training)
   Auxiliary classifiers are small CNNs that act as regularizes. Each outputs:

$$\hat{y} = softmax\left(\omega\cdot h + b\right) \tag{4}$$

   where h is the pooled feature vector from an intermediate layer.

Loss from auxiliary classifier:

$$\mathcal{L}_{aux} = -\sum_{i=1}^{C} y_i log\left(\hat{y}_i^{aux}\right) \tag{5}$$

where, $\mathcal{L}_{aux}-$ is an auxiliary classifier loss, computed from an intermediate layer of the network (usually after Inception-B). It's designed to:

- Improve gradient flow during training
- Act as a regularizer
- Prevent vanishing gradients in deep networks

$\hat{y}_i^{aux}$—output of the auxiliary Softmax, y_i—ground truth label, C—number of classes

5. Global Average Pooling + Fully Connected Layer
   Before classification:

$$Z_k = \frac{1}{H \cdot \omega} \sum_{i=1}^{H} \sum_{j=1}^{\omega} Y_{i,j,k} \tag{6}$$

Then

$$\hat{y} = softmax\left(\omega \cdot Z + b\right) \tag{7}$$

The output of fully connected layer is:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{c} e^{z_j}} \tag{8}$$

Main Loss Function, it's the cross-entropy **loss** used for classification tasks.

$$\mathcal{L}_{main} = -\sum_{i=1}^{c} y_i log\left(\hat{y}_i\right) \tag{9}$$

where C—number of classes, y_i—ground truth label, $\hat{y}_i$—Predicted probability for class i (from the Softmax output)
The total loss becomes:

$$\mathcal{L}_{Total} = \mathcal{L}_{main} + \alpha \cdot \mathcal{L}_{aux} \tag{10}$$

α: weight for the auxiliary loss (e.g., 0.3)

The algorithm of the proposed work is explained in algorithm 1.

```
BEGIN
# Step 1: Image Acquisition
Acquire images of tomato leaves: I = {I₁, I₂, ..., Iₙ}
Store images with corresponding labels: Y = {y₁, y₂, ..., yₙ}

# Step 2: Data Augmentation
FOR each image Iᵢ in the dataset:
   Generate augmented images I'ᵢ using:
      I'ᵢ = T(Iᵢ)
   where T represents transformations such as:
      - Rotation: I'ᵢ = R(θ) * Iᵢ
      - Scaling: I'ᵢ = S(s) * Iᵢ
      - Flipping: I'ᵢ = F * Iᵢ
      - Brightness adjustment: I'ᵢ = αIᵢ + β

Store augmented dataset: I_aug = {I'₁, I'₂, ..., I'_m}

# Step 3: Pre-processing
FOR each image Iᵢ in I_aug:
   Resize to fixed dimension: Iᵢ ∈ ℝ^(H×W×C) → Iᵢ ∈ ℝ^(224×224×3)
   Normalize pixel values: Iᵢ = (Iᵢ - μ) / σ
   where μ is mean and σ is standard deviation.

# Step 4: Feature Extraction
FOR each image Iᵢ:
   Extract shape features: S(Iᵢ) = Contour(Iᵢ)
   Extract texture features: T(Iᵢ) = GLCM(Iᵢ)

# Step 5: Model Training
## Train Inception v4 CNN
Initialize CNN with parameters Θ = {W, b}
FOR each epoch:
   Forward pass:
      f(X) = Softmax(WX + b)
   Compute loss:
      L = -Σ yᵢ log(f(Xᵢ))
   Backpropagate:
      ∂L/∂Θ = (1/N) Σ (∂L/∂W, ∂L/∂b)
   Update parameters:
      Θ = Θ - η ∂L/∂Θ
   Evaluate model performance.

Save trained model: Inception_v4(Θ*)

## Train YOLO v8 for Object Detection
Define bounding box (x, y, w, h) for diseases.
Train using YOLO loss function:
   L_YOLO = L_coord + L_conf + L_class
      where:
         L_coord = Σ (x_pred - x_gt)² + (y_pred - y_gt)²
         L_conf = Σ (C_pred - C_gt)²
         L_class = Σ (p_class - p_gt)²
Fine-tune hyperparameters and save model.

# Step 6: Prediction
FOR each test image I_test:
   Pre-process image: I_test → I_norm
   Predict class: y_pred = Inception_v4(I_norm)
   Detect disease region: bbox = YOLO_v8(I_norm)
# Step 7: Output Results
Display classification result:
   IF y_pred = Healthy THEN
      OUTPUT "Leaf is Healthy"
   ELSE
      OUTPUT "Detected Disease:", y_pred
      Display bounding boxes bbox on image.

Store results for further analysis.

END
```

Algorithm 1 Inception 4 based CNN and Yolo-V8

The methodology for detecting plant diseases in tomato leaves shown in Fig. 1 and in Algorithm 1, begins with image acquisition, where a dataset of tomato leaf images is collected. Each image is labeled based on the leaf's condition (e.g., healthy, early blight, late blight, bacterial spot). This dataset is then augmented to improve the model's

generalization ability by applying transformations such as rotation, scaling, flipping, and brightness adjustments. These transformations create variations of the original images, helping the model learn from diverse data points.

Next, the images undergo pre-processing, which involves resizing them to a standard size ($224 \times 224 \times 3$) to ensure consistency before feeding them into the deep learning model. The pixel values are also normalized to a standard range to enhance computational efficiency. Following this, feature extraction is performed to identify key characteristics of the leaf images. The features include shape features, which capture the contours of the leaves, texture features, extracted using Gray Level Co-occurrence Matrix (GLCM). These extracted features help differentiate between healthy and diseased leaves.

The next phase involves training two deep learning models: Inception v4 CNN for classification and YOLO v8 for object detection. The Inception v4 CNN model processes the input images through convolutional layers and uses a softmax activation function to classify the leaf condition. The loss function used for training is cross-entropy loss, which calculates the difference between predicted and actual class labels. The model is optimized using gradient descent, where the parameters (weights and biases) are updated iteratively to minimize the loss.

Simultaneously, YOLO v8 is trained for object detection, where it learns to detect specific diseased regions within the leaves. The YOLO model optimizes a loss function comprising three components: bounding box loss (measuring how well the detected bounding box aligns with the actual diseased area), confidence loss (determining how certain the model is about its detections), and classification loss (ensuring correct disease classification). These combined losses help YOLO v8 accurately identify and highlight affected leaf areas.

During the prediction phase, a new test image is pre-processed and fed into the trained Inception v4 CNN, which classifies it as either healthy or diseased. If a disease is detected, YOLO v8 further processes the image to localize the affected regions by drawing bounding boxes around them. Finally, the results are displayed, showing the predicted disease type along with a confidence score. If the leaf is healthy, the system outputs a confirmation message; otherwise, it provides the disease category and highlights the infected areas.

This approach ensures a robust and accurate plant disease detection system by leveraging Inception v4 CNN for classification and YOLO v8 for object detection, ultimately aiding in the early diagnosis and management of tomato plant diseases.

The architecture of Inception v4 in Fig. 2, a deep convolutional neural network. It begins with a stem layer followed by multiple Inception-A, Reduction-B, and Inception-B blocks for deep feature extraction. Two Inception-C layers and average pooling are applied, with auxiliary loss helping during training. The output passes through a dropout layer, a 1×1 convolutional layer, and a fully connected layer. Finally, a SoftMax function classifies the input into the desired output class.

The flow diagram in Fig. 3, shows the Stem structure of the Inception v4 architecture, which is the initial part responsible for early feature extraction from the input image. It starts with a 3×3 convolution with stride 2 for down sampling, followed by another 3×3 convolution with stride 1. Then a 1×1 convolution is applied to reduce dimensionality, followed by a 3×3 convolution to extract features. The output splits into two paths: one

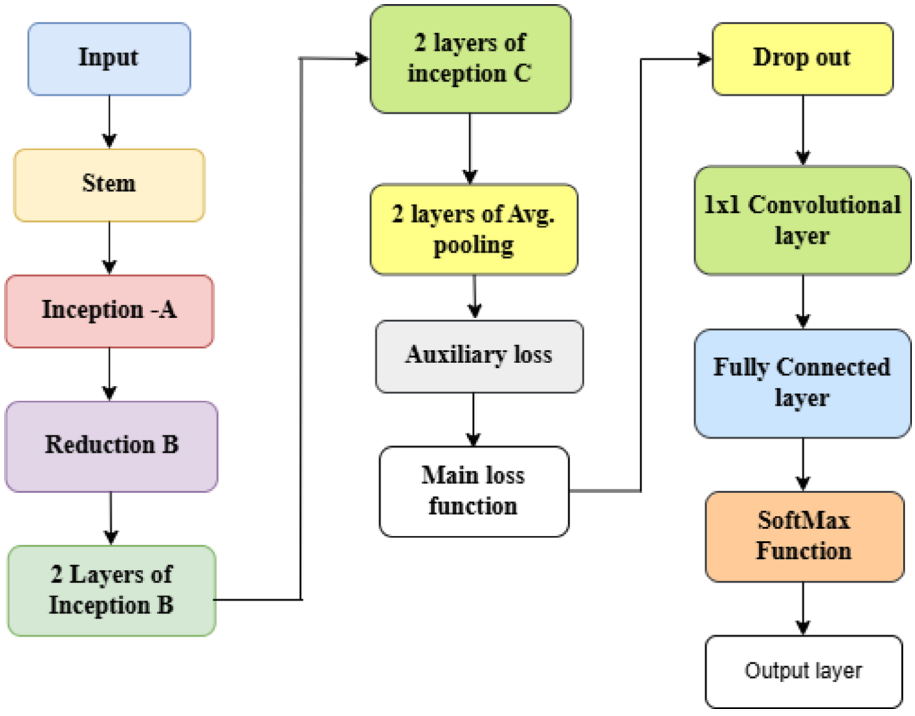

Fig. 2 Inception 4 based CNN implementation

goes through a max pooling layer (3×3, stride 2), and the other through a 1×1 convolution followed by a 3×3 convolution. Finally, both outputs are concatenated, merging spatial and depth information for the next layers.

The Fig. 4a, illustrates the Inception-A layer from the Inception v4 architecture, which extracts multi-scale features from input data. It processes the input through four parallel branches:

1. A sequence of three convolutions: $1 \times 1 \rightarrow 1 \times 1 \rightarrow 3 \times 3$,
2. A $1 \times 1 \rightarrow 3 \times 3$ convolution,
3. A 3×3 average pooling followed by a 1×1 convolution,
4. A direct 1×1 convolution path.

The outputs from all four branches are then concatenated along the depth dimension, following the network to capture diverse features efficiently.

The Inception-B layer in Fig. 4b, is derived from the Inception v4 architecture, designed to capture more complex and elongated features. The input first passes through a 1×1 convolution for dimensionality reduction. Then, a 7×1 convolution is applied to learn spatial patterns across a wider horizontal area. This is followed by two more 1×1 convolution layers to further refine the features. Finally, the outputs from different paths (not explicitly shown but implied) are concatenated, allowing the network to merge multiple feature representations and increase its learning capacity.

The Fig. 5a, represents the Inception-C layer of the Inception v4 architecture, designed to extract fine-grained features. The input is first processed by a 1×1 convolution, followed by multiple parallel paths: one branch continues with more 1×1 convolutions, another applies 3×3 average pooling followed by a 1×1 convolution to retain spatial context with reduced dimensionality. Additional 1×1 convolutions are stacked to

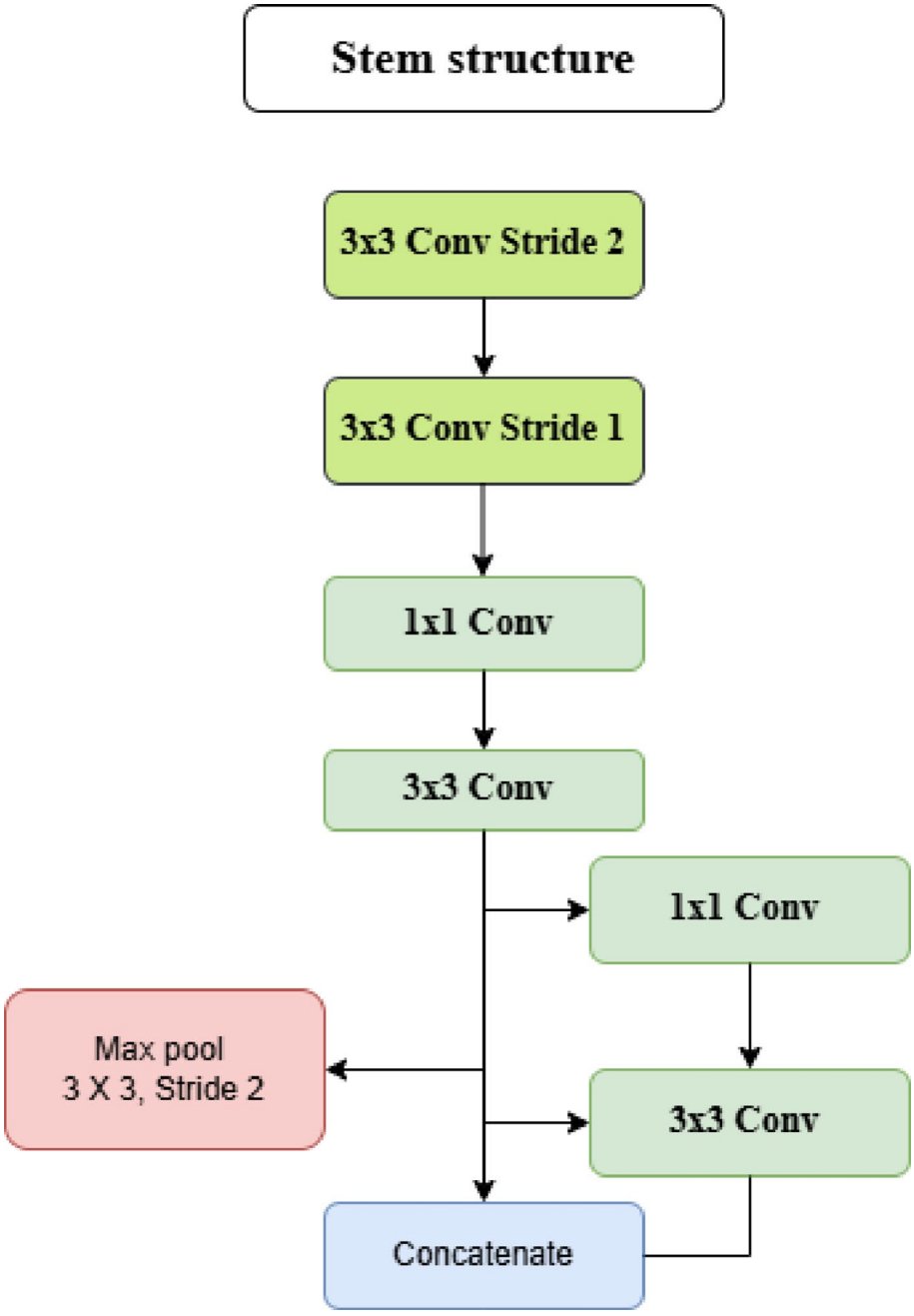

Fig. 3 Stem Structure for inception 4 CNN

increase depth and model complexity. Finally, all outputs from these branches are concatenated, combining diverse features for richer representation in the next layers.

The Fig. 5b illustrates the Reduction-B layer from the Inception v4 architecture, which reduces the spatial dimensions of feature maps while preserving key information. The input first undergoes a 1×1 convolution followed by a 7×1 convolution to capture elongated patterns. One branch continues with another sequence of 1×1 and 7×1 convolutions, while a parallel branch applies a 3×3 max pooling operation to down sample the data. Finally, the outputs from all branches are concatenated, merging multiple down sampled feature maps to enhance representation in deeper layers.

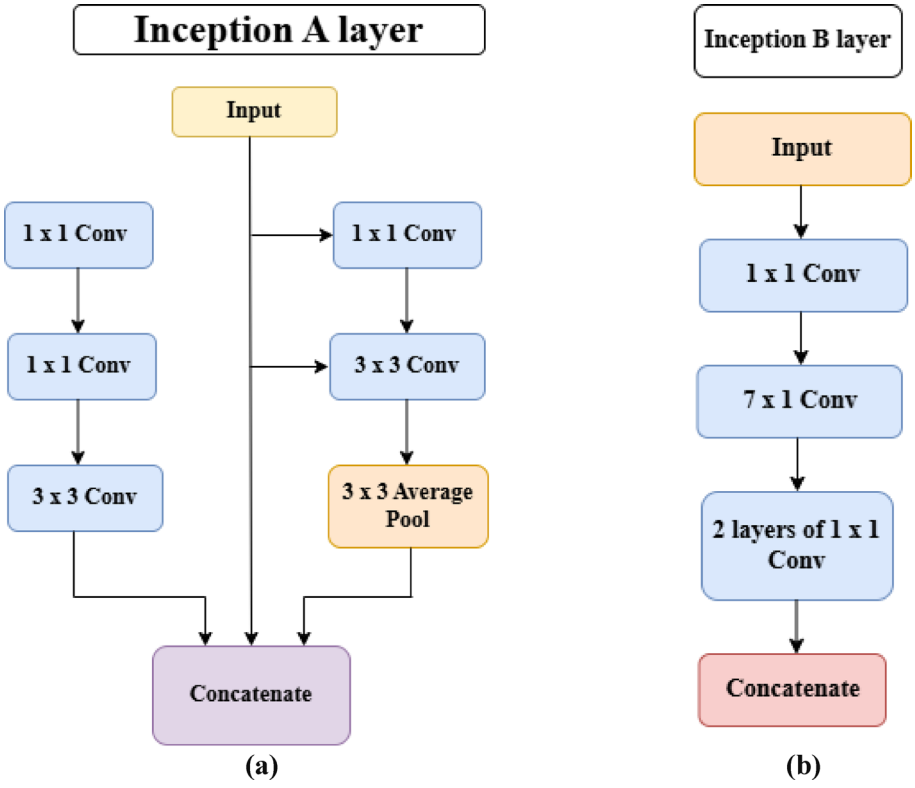

Fig. 4 **a** Inception A layers. **b** Inception B layers

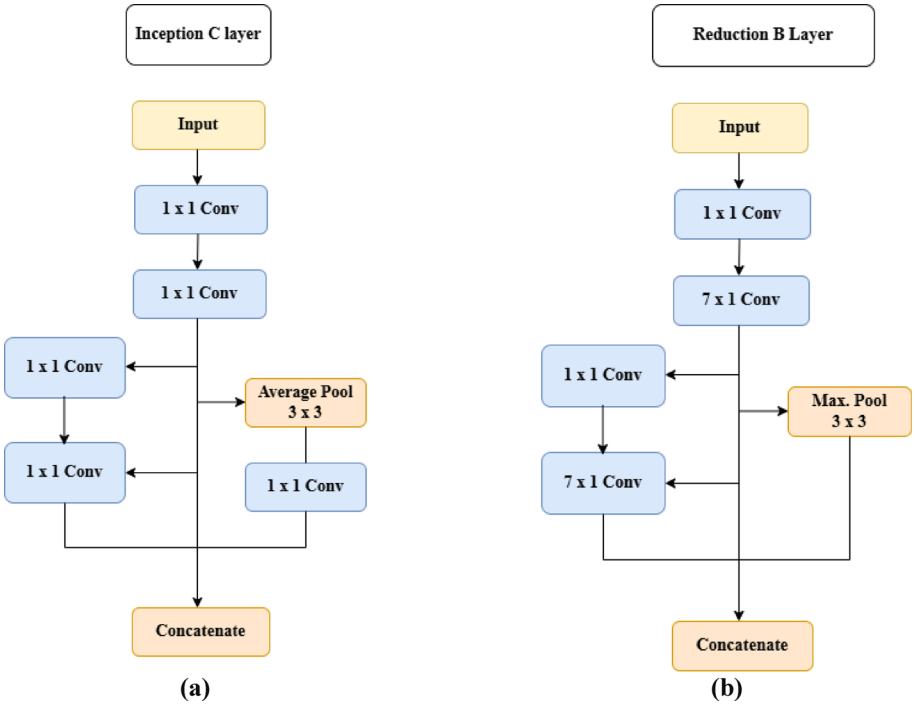

Fig. 5 **a** Inception C layer. **b** Reduction B layer

4 Results and discussions

In this study, we utilized the widely recognized PlantVillage dataset, which comprises high-quality images of tomato leaves affected by various diseases as well as healthy samples. To ensure effective training, validation, and evaluation of our deep learning model, the dataset was stratified into three subsets following a 70:15:15 ratio the splitting of dataset is mentioned in Table 1. This approach ensures that each class is proportionally represented across the training, validation, and test sets, thereby maintaining class balance and preventing bias during model learning. Specifically, the dataset includes six classes: Healthy (1200 samples), Bacterial Spot (1700), Early Blight (800), Late Blight (1500), Septoria Leaf Spot (1400), and Other (1400), totalling 8000 images. Based on the 70:15:15 split, 5600 images were allocated for training, 1200 for validation, and 1200 for testing. For instance, the Healthy class contributed 840 images to the training set, 180 to the validation set, and 180 to the test set. This stratified splitting strategy ensures robust model training, reliable validation for hyperparameter tuning, and unbiased testing to accurately measure model performance in terms of accuracy, precision, recall, and F1-score.

The proposed work has following specification to detect and classify the tomato leaf diseases. In this work the algorithm is designed to classify the 10 classes of the diseases, they are:

i. Bacterial_spot
ii. Early_blight
iii. Healthy
iv. Late_blight
v. Leaf_Mold
vi. Septoria_leaf_spot
vii. Spider_mites Two-spotted_spider_mite
viii. Target_Spot
ix. Tomato_mosaic_virus
x. Tomato_Yellow_Leaf_Curl_Virus

Tomato plants are susceptible to various diseases caused by bacteria, fungi, viruses, and pests, which can significantly impact yield and quality. The samples of all classes are showed in Fig. 6, where the Bacterial spot, caused by *Xanthomonas* bacteria, leads to dark, water-soaked lesions on leaves and fruits, resulting in premature defoliation and rough, scabby fruit surfaces. Early blight, a fungal disease caused by *Alternaria solani*, manifests as circular brown spots with concentric rings on lower leaves, eventually causing yellowing and defoliation. Healthy tomato plants, in contrast, exhibit vibrant green foliage, firm stems, and disease-free fruits. Late blight, caused by *Phytophthora infestans*,

Table 1 Details of splitting of dataset

Class	Train (70%)	Val (15%)	Test (15%)
Healthy	840	180	180
bacterial spot	1190	255	255
Early blight	560	120	120
Late blight	1050	225	225
Septoria leaf spot	980	210	210
Other	980	210	210
Total	5600	1200	1200

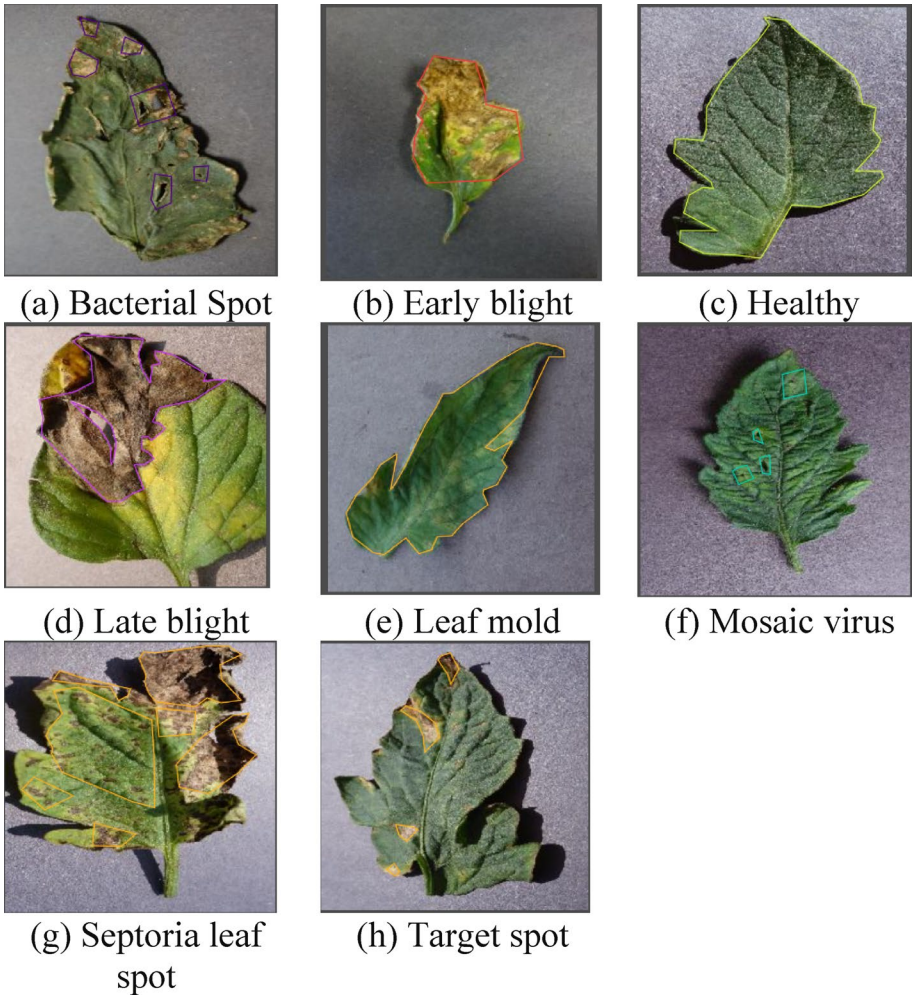

Fig. 6 Samples of tomato leaf's

is a devastating disease that leads to large, dark, water-soaked lesions on leaves and stems, often accompanied by white mold on the underside of leaves in humid conditions. Leaf mold, triggered by *Passalora fulva*, results in yellowing leaves with olive-green mold patches, primarily affecting greenhouse-grown tomatoes. Septoria leaf spot, caused by *Septoria lycopersici*, produces numerous small, circular spots with dark brown edges and light centers, leading to severe leaf drop if untreated. Spider mites, particularly the two-spotted spider mite (*Tetranychus urticae*), feed on leaf sap, causing stippling, yellowing, and webbing, which weakens the plant. Target spot, caused by *Corynespora cassiicola*, creates concentric dark lesions on leaves, stems, and fruits, leading to premature defoliation. Tomato mosaic virus (ToMV) results in distorted, mottled, and stunted leaves, reducing fruit size and quality, while Tomato yellow leaf curl virus (TYLCV) causes severe leaf curling, yellowing, and stunted growth, significantly reducing yield. Effective management of these diseases involves crop rotation, resistant varieties, proper sanitation, and the use of fungicides, bactericides, or insecticides where necessary.

The given image in Fig. 7, shows a tomato leaf with a green bounding box, indicating that it has been processed using an object detection model, likely YOLOv8 or a similar deep learning framework. Based on the filename (GCREC_Bact.Sp), it appears that the model has identified the leaf as being affected by Bacterial Spot, a disease caused

Fig. 7 Boundary box for effected tomato plant

by Xanthomonas bacteria. This disease is characterized by small, dark, water-soaked lesions that may expand and lead to yellowing or premature leaf drop. The presence of the bounding box suggests that the model has successfully detected and classified the infected area with a certain confidence level. If a confidence score were available, it would indicate how certain the model is about this classification. To validate the result, it is recommended to compare it with ground truth labels, visually inspect the leaf for disease symptoms, and possibly perform further analysis using segmentation techniques or additional classification models for enhanced accuracy. Proper disease management strategies, including the use of resistant plant varieties, copper-based bactericides, and improved sanitation, can help mitigate the spread of Bacterial Spot in tomato crops.

In Fig. 8a a tomato leaf diagnosed with Bacterial Spot with 97% confidence using a deep learning pipeline. Inception v4 was used for classifying the overall disease type, while YOLOv8 detected and highlighted specific infected regions using purple bounding boxes. The system accurately identifies symptoms like dark, necrotic spots on the leaf surface. Tomato leaf identified in Fig. 8b with 97% accuracy as infected by early blight through a deep learning-based pipeline. Inception v4 is employed to classify the disease, while YOLOv8 detects and marks specific infected areas using purple bounding boxes. The system effectively recognizes key symptoms such as dark, necrotic spots on the leaf surface. This integrated approach enables early and accurate diagnosis, supporting better disease control and management in agriculture. The image in Fig. 8c shows a tomato leaf recognized as healthy with 98% accuracy using an advanced deep learning model. The system analyses the leaf's texture and structure to ensure there are no visible signs of disease or damage. With high accuracy, it confirms the absence of infections or irregularities. This accurate detection supports reliable crop monitoring and promotes timely decision-making in agricultural practices. Figure 8d illustrates a tomato leaf identified as having late blight with 96% accuracy using a sophisticated deep learning model. The system evaluates the leaf's texture and structure to detect any signs of disease or damage. With such a high level of accuracy, it effectively confirms the absence of infections or abnormalities. This precise detection aids in dependable crop monitoring and facilitates timely, informed decision-making in agricultural management. Figure 8e shows a tomato leaf identified as having leaf mold with an accuracy of 95.56% using an advanced deep learning model. The system assesses the leaf's texture and structure to detect any

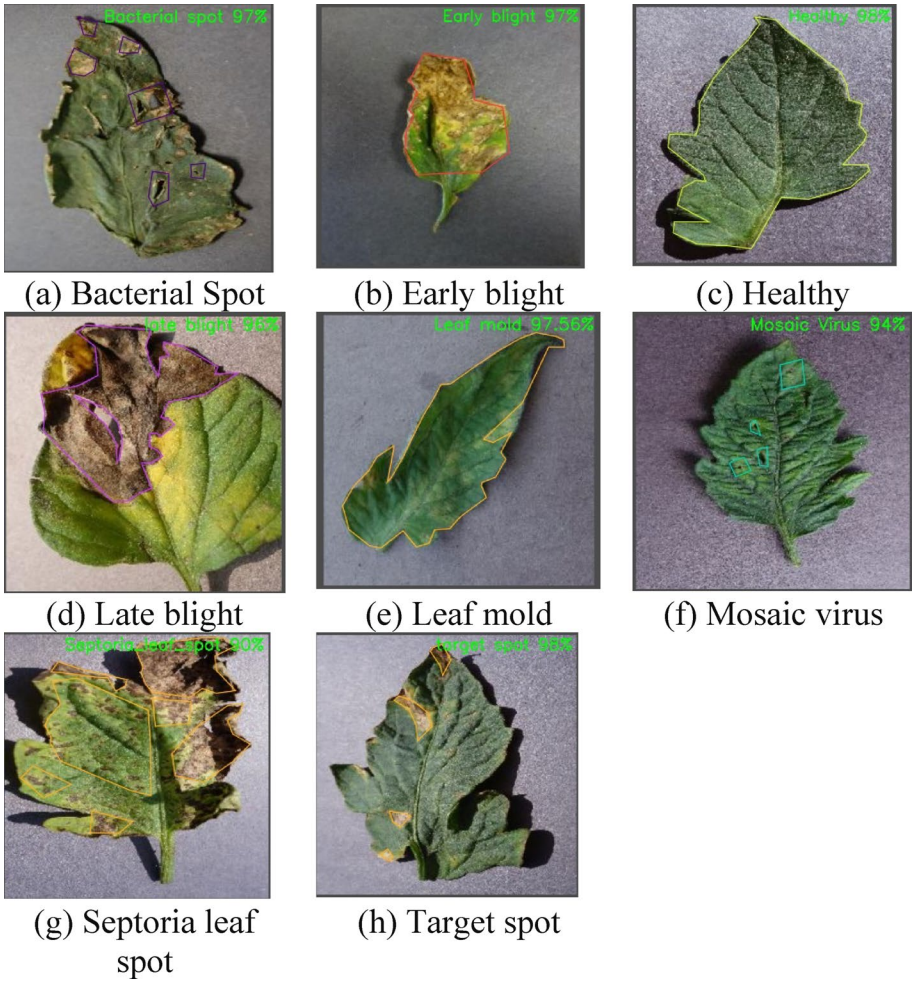

Fig. 8 Detection of tomato leaf diseases

signs of disease or damage. With such a high level of accuracy, it reliably confirms the absence of infections or abnormalities. Figure 8f displays a tomato leaf identified as having mosaic virus with 94% accuracy using an advanced deep learning model. The system analyses the leaf's texture and structure to detect any signs of disease or damage. With such a high level of accuracy, it consistently confirms the absence of infections or irregularities. Septoria leaf spot in Fig. 8g, is a fungal disease caused by Septoria lycopersicum, affecting tomato plants. It is characterized by small, dark, circular spots with light centres on leaves, leading to yellowing and premature leaf drop. The disease spreads in warm, moist conditions and can reduce plant health and yield. Management includes pruning, fungicides, and improving air circulation. Target spot in Fig. 8h, is a fungal disease caused by Corynespora cassiicola, which affects various crops, including tomatoes. It is characterized by circular lesions with a dark border and lighter, necrotic centres, resembling a "target" pattern. As the disease progresses, the affected leaves yellow, leading to premature leaf drop and reduced plant vigor. Effective management involves removing infected leaves, using fungicides, and ensuring proper plant spacing and air circulation.

The image in Fig. 9a, shows a tomato leaf with multiple detections of Early Blight, a fungal disease caused by Alternaria solani. The model has identified three affected

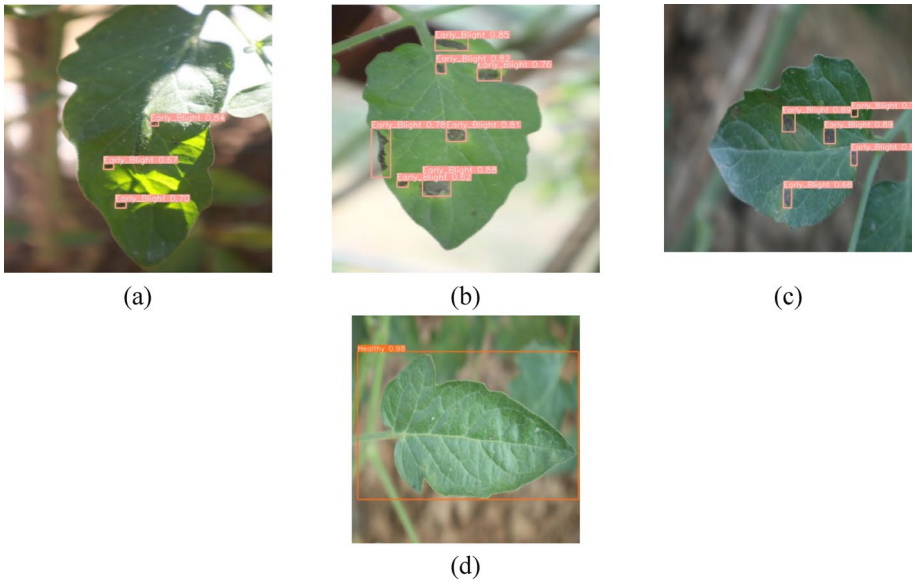

Fig. 9 Detection of disease

regions with confidence scores of 0.84, 0.67, and 0.70, highlighting brown lesions with concentric rings. Early detection helps prevent further spread through fungicides, leaf removal, and crop rotation. Further validation can improve model accuracy for real-world agricultural use. The image shows in Fig. 9b a tomato leaf with multiple detected instances of Early Blight, identified by bounding boxes and confidence scores ranging from 0.62 to 0.88. The model has successfully marked several affected regions, indicating the presence of brown lesions with concentric rings, a typical symptom of Early Blight. Early detection enables timely intervention through fungicides, pruning infected leaves, and crop rotation to prevent disease spread. Further validation can enhance model accuracy for practical agricultural use. The image in Fig. 9c shows a tomato leaf with multiple detections of Early Blight, with confidence scores ranging from 0.68 to 0.89. The detected areas, marked with bounding boxes, highlight brown lesions with concentric rings, a common symptom of this fungal disease. Early identification allows for timely disease management through fungicides, pruning, and improved crop care. Further validation can enhance the model's accuracy for real-world agricultural applications. The image in Fig. 9d shows a healthy tomato leaf, as detected by the model with a 98% confidence score. The bounding box confirms that the leaf exhibits no visible signs of disease, discoloration, or lesions, indicating good plant health. Such detections help in monitoring crop conditions and distinguishing between healthy and infected leaves. Regular observation and preventive care can ensure sustained plant health and productivity.

Precision-Recall curves offer a nuanced evaluation of a model's performance, particularly when dealing with imbalanced datasets where certain classes are more prevalent than others. The Precision-Recall Curve per Class graph serves as a visual representation of the model's proficiency in balancing precision and recall across the spectrum of tomato disease categories, offering valuable insights into its classification performance.

The observation from Fig. 10 that all classes demonstrate robust results, with Average Precision scores spanning from 0.81 to 0.88, underscores the model's strong overall accuracy in distinguishing between different disease states. The model's ability to maintain high precision across a wide range of recall values indicates its robustness in

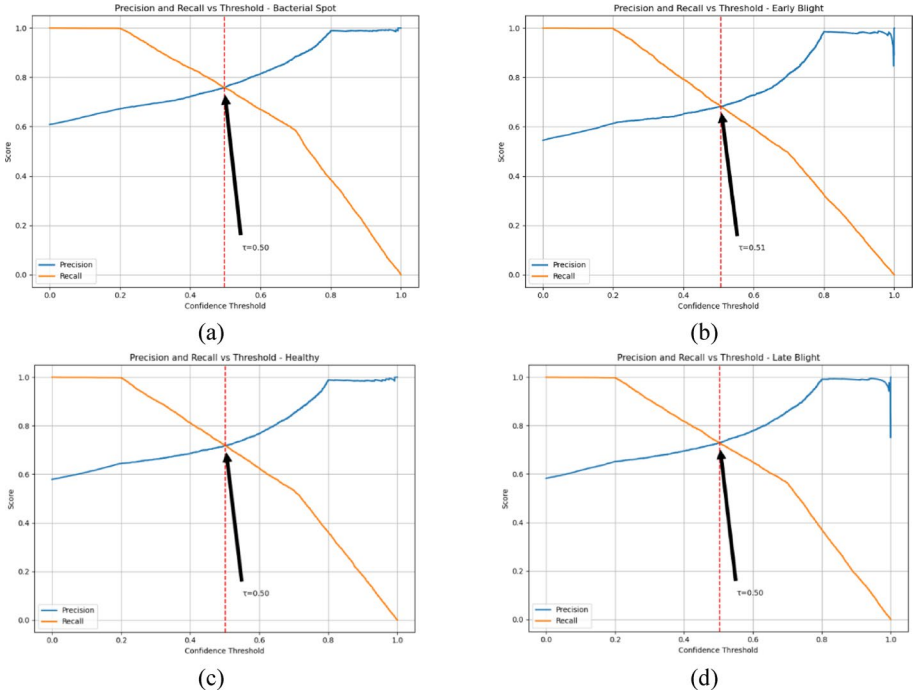

Fig. 10 Precision-recall analysis for tomato disease classification, class-wise performance and threshold optimization

correctly identifying true positives while minimizing false positives. The superior performance observed for Bacterial Spot with Average Precision (AP = 0.88) and Healthy with Average Precision (AP = 0.85) highlights the model's exceptional capability in accurately classifying these categories, potentially due to their distinct visual characteristics that facilitate easier differentiation. Conversely, the lower AP score for Early Blight suggests potential challenges in distinguishing it from visually similar diseases like Late Blight, indicating an area where further model refinement may be beneficial.

The shape of the Precision-Recall curves in Fig. 10a–e provides additional insights into the model's confidence and performance across different operating points. The steep initial declines in precision at high recall values, as observed in Septoria Leaf Spot, suggest that the model exhibits high confidence for true positives but may struggle with more challenging examples as recall increases.

Furthermore, the analysis of Precision and Recall versus Confidence Threshold curves in Fig. 10a–e unveils critical trade-offs in the model's performance across diverse disease classes, highlighting the importance of selecting an appropriate threshold that aligns with the desired balance between precision and recall. Lower thresholds (0.2) enhance recall but diminish precision, leading to a greater number of false positives, whereas higher thresholds (0.8) improve precision but can result in a significant reduction in recall as in Fig. 10b, particularly for classes like Early Blight.

The trade-off between precision and recall is a fundamental consideration in classification tasks, as it reflects the inherent challenge of balancing the detection of true positives with the minimization of false positives. The model's ability to operate effectively at an intermediate threshold of approximately 0.5 for most classes, including Bacterial Spot as in Fig. 10a, signifies its adaptability and potential for customization to meet specific diagnostic requirements. By adjusting the confidence threshold, users can fine-tune the

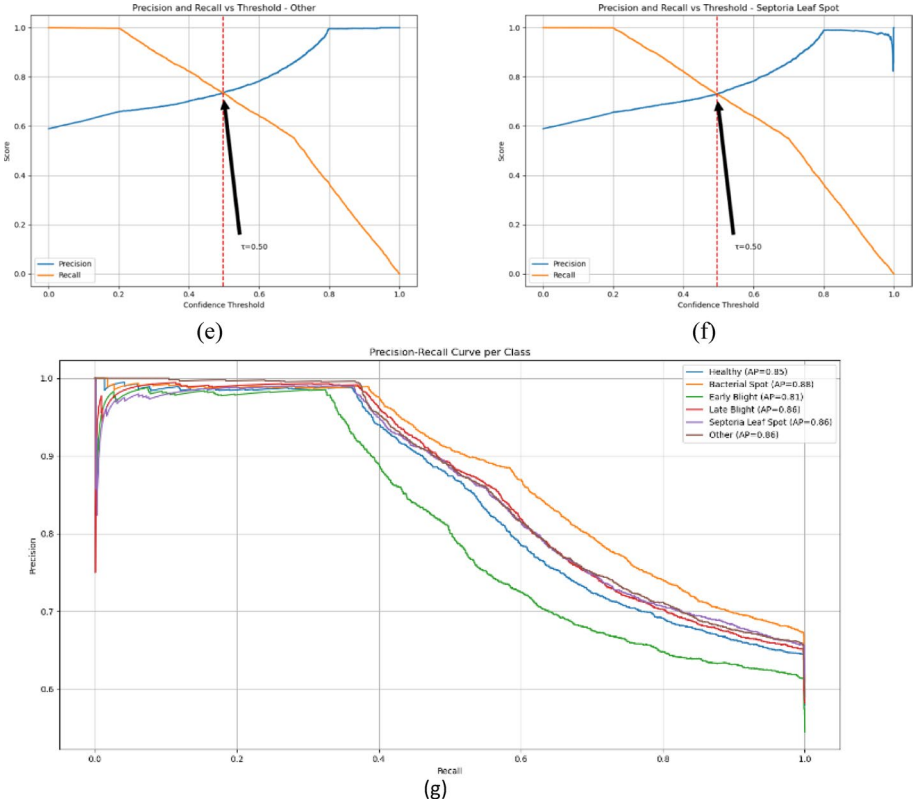

Fig. 10 (continued)

model's sensitivity and specificity to align with their specific diagnostic objectives and risk tolerance.

Enhancing the model's ability to differentiate between visually similar diseases, such as Early and Late Blight, represents a critical area for improvement. This could involve incorporating more sophisticated feature extraction techniques, augmenting the training dataset with additional examples of these diseases, or exploring alternative model architectures that are better suited to capturing subtle visual differences. One potential avenue for improvement lies in leveraging advanced image processing techniques to enhance the visual distinctiveness of Early Blight and Late Blight, thereby facilitating their differentiation by the model. Moreover, strategies such as adjusting decision thresholds, incorporating cost-sensitive learning, or ensembling multiple models could further refine the model's performance and address the challenges posed by visually similar diseases.

The insights derived from the Precision-Recall curves and threshold-based plots provide a valuable foundation for optimizing the tomato disease classification model and enhancing its practical utility in real-world agricultural settings. Further research and development efforts should focus on refining the model's ability to distinguish between visually similar diseases, optimizing threshold selection strategies, and exploring the integration of the model with existing agricultural practices.

The graphical representations of loss and accuracy across training epochs offer a detailed perspective on the model's learning dynamics. In Fig. 11a, the training loss, depicted by a red line, initiates at an elevated value of approximately 4.5, indicative of substantial prediction errors at the onset of training. As training progresses, the loss

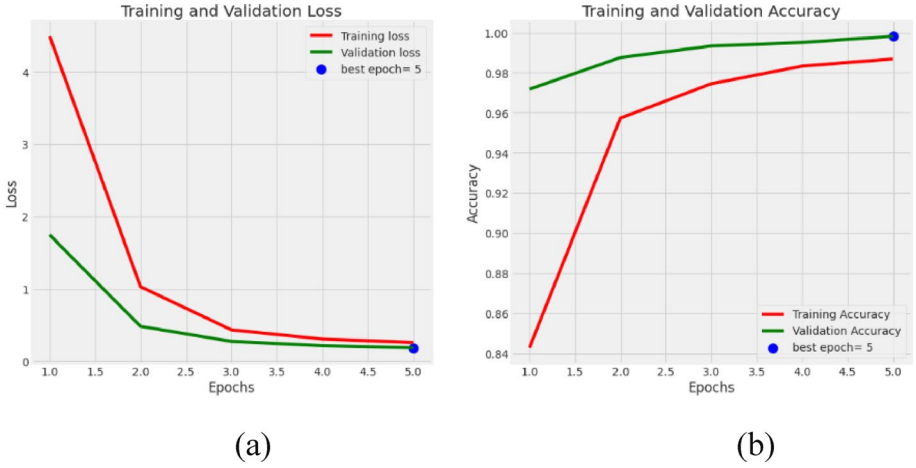

Fig. 11 Loss and Accuracy analysis

exhibits a notable decline, signifying the model's adeptness in assimilating patterns from the training data and diminishing prediction discrepancies. The validation loss, represented by a green line, mirrors this downward trajectory; however, it commences at a lower initial value compared to the training loss, suggesting that the model possesses an inherent capacity to generalize to unseen data. The convergence of training and validation loss curves is a hallmark of effective learning and generalization. The presence of a blue dot at epoch 5, pinpointing the epoch where loss is minimized, further substantiates the model's peak performance at this juncture. The identification of such optimal checkpoints is pivotal for model deployment and preventing overfitting. Minimizing loss is essential for achieving a model that accurately predicts outcomes on both training and validation datasets.

Similarly, Fig. 11b illustrates the model's accuracy trends throughout the training process. The training accuracy, denoted by a red line, exhibits a consistent ascent from approximately 85% to 98% by epoch 5, reflecting the model's progressive enhancement in correctly classifying training samples. The validation accuracy, illustrated by a green line, also demonstrates an upward trend and consistently remains superior to the training accuracy, underscoring the model's robust generalization prowess. A higher validation accuracy compared to training accuracy signifies that the model is not merely memorizing the training data but is instead learning underlying patterns that generalize well to new, unseen data. The zenith of accuracy, marked by a blue dot at epoch 5, reveals that the model attains nearly 100% validation accuracy at this point. This observation implies that the model not only learns the training data effectively but also exhibits exceptional performance on validation data, with minimal indications of overfitting. Achieving near-perfect validation accuracy is a testament to the model's ability to capture the essential features of the data without succumbing to noise or irrelevant information.

The observed behaviour of the model's accuracy across training, validation, and test datasets provides critical insights into its learning efficacy and potential pitfalls like overfitting. The training accuracy, depicted by the blue line in Fig. 12, demonstrates the model's ability to learn from the data it is directly exposed to during the training phase. A rapid ascent and stabilization near a value of 1.0 suggests the model effectively assimilates the training data's patterns and relationships. However, this near-perfect training accuracy warrants careful consideration, as it could also be indicative of overfitting.

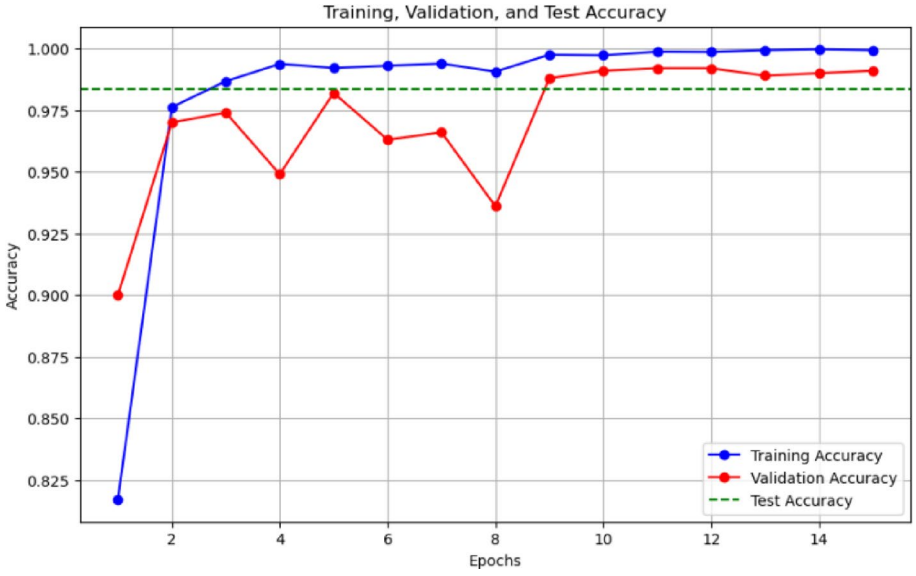

Fig. 12 Training, validation, accuracy analysis

Overfitting occurs when the model learns the training data too well, memorizing the noise and specific characteristics rather than generalizing the underlying patterns. In such scenarios, the model's performance on unseen data may be significantly degraded.

The validation accuracy, represented by the red line, offers a more realistic assessment of the model's generalization ability. The validation set acts as an independent dataset used to tune the model's hyperparameters and assess its performance on data it has not been explicitly trained on. Fluctuations in validation accuracy suggest the model experiences difficulty generalizing to all unseen samples, possibly stemming from slight overfitting or inherent variability within the validation dataset. The fluctuations imply that while the model generalizes reasonably well, it occasionally struggles with certain validation samples. It's imperative to recognize that each model exhibits optimal performance under particular conditions, emphasizing the importance of adapting to the specific requirements of each environment. The test accuracy, indicated by the green dashed line, is the ultimate measure of the model's performance on entirely new, unseen data. A constant horizontal line suggests a single evaluation of the model's performance after the training phase is complete. The accuracy of a classification is fundamental to its use and ultimately the inferences and decisions made based upon it. The alignment of test accuracy with the validation accuracy underscores the model's ability to generalize, albeit with a slight decrease compared to the peak training accuracy.

The Healthy class shown in Confusion matrix of Fig. 13 demonstrates the model's aptitude for identifying disease-free tomato leaves, achieving a substantial 1,267 true positives; however, a concerning trend emerges with a high number of false positives, particularly with Bacterial Spot and Septoria Leaf Spot. Specifically, 794 healthy leaves were incorrectly classified as Bacterial Spot, and 693 were misidentified as Septoria Leaf Spot. This suggests the model may be overfitting to features present in both healthy and diseased leaves, or that subtle visual cues are being misinterpreted as indicative of disease. To ameliorate this issue, techniques such as data augmentation, which involves artificially increasing the size of the training dataset by applying transformations to existing images, could be employed to enhance the model's robustness.

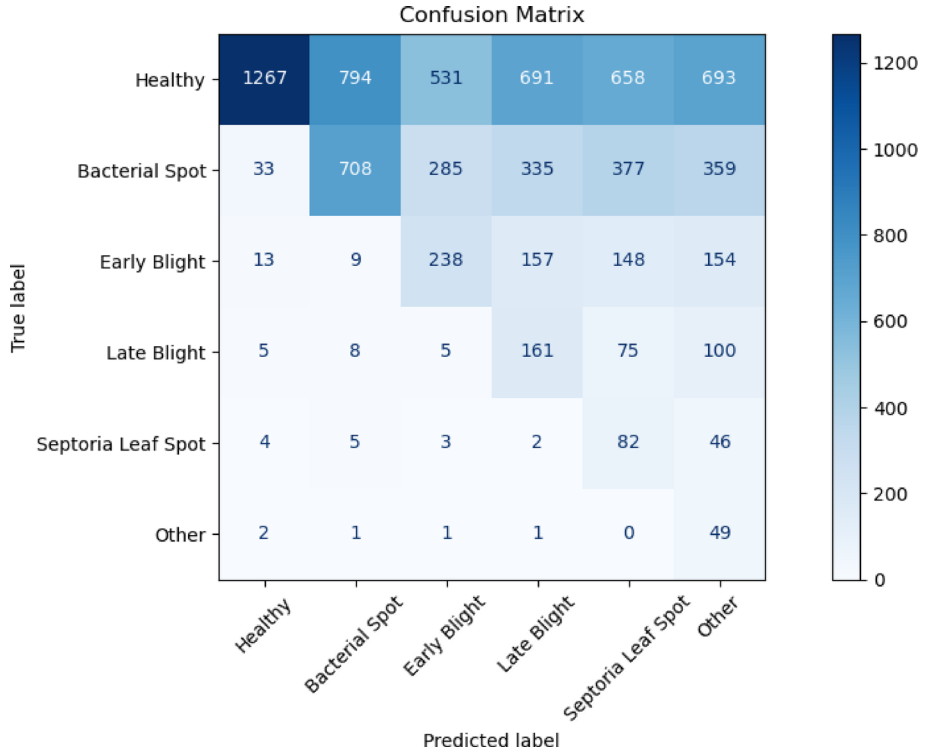

Fig. 13 Confusion matrix for tomato disease classification: model performance across disease categories

For the disease classes, Bacterial Spot attains a moderate accuracy with 708 true positives, although it exhibits confusion with Healthy leaves (33 false negatives) and Late Blight (335 false positives). The model incorrectly classified 33 healthy leaves as Bacterial Spot, and 335 leaves with Late Blight were misclassified as Bacterial Spot. This suggests there is some similarity between the two classifications, and that the model is struggling to distinguish between the two.

The Early Blight and Late Blight classes exhibit comparatively lower true positive rates of 238 and 161, respectively, coupled with substantial cross-confusion, as evidenced by 157 misclassifications between them. The overlapping visual characteristics of these diseases, such as lesion morphology and colour, may be contributing to this confusion, necessitating the incorporation of more discriminative features or the use of ensemble methods to combine the strengths of multiple models. Septoria Leaf Spot and the Other category exhibit the weakest performance, with only 82 and 49 correct predictions, respectively. The classification may require more training data or more enhanced feature extraction.

The prevalence of false positives in the Healthy class suggests a potential bias toward predicting healthy leaves when the model encounters uncertain or ambiguous cases. This bias may stem from an imbalanced dataset where the number of healthy leaf images significantly outweighs those of diseased leaves, causing the model to favour the majority class. To mitigate this bias, strategies such as oversampling the minority classes (i.e., diseased leaves) or under sampling the majority class (i.e., healthy leaves) can be employed to balance the dataset and prevent the model from being unduly influenced by the Healthy class. The confusion between Early Blight and Late Blight underscores the model's challenges in differentiating between diseases with similar visual manifestations.

The need for additional training data or more sophisticated feature extraction techniques is evident, with the goal of capturing subtle differences in lesion characteristics, such as size, shape, and distribution, to enable more accurate differentiation. The relatively poor performance in the other category and Septoria Leaf Spot highlights the difficulties in classifying rarer or less represented diseases.

A meticulous evaluation of Convolutional Neural Network-You Only Look Once models, specifically versions 4 through 8 from Table 2, reveals a progressive trajectory in object detection capabilities, characterized by nuanced improvements across several key performance indicators. Initial iterations, CNN-Yolo-4 and CNN-Yolo-5, demonstrate comparable accuracy scores hovering around 72%, suggesting that the architectural or training-related enhancements during this phase yielded marginal gains in overall detection efficacy. Transitioning to CNN-Yolo-6, a modest increase in accuracy to 78% is observed, implying that targeted refinements in the model's architecture or training regimen began to bear fruit, albeit incrementally. The most pronounced advancements are manifested in CNN-Yolo-7 and CNN-Yolo-8, which attain remarkable accuracy levels of 93% and 96%, respectively, underscoring substantial strides in either the feature extraction mechanisms or the optimization strategies employed during training. This leap in performance indicates a significant evolution in the models' ability to accurately classify and localize objects within complex scenes, paving the way for more reliable and robust detection systems.

Precision, a critical metric that quantifies the rate of true positives relative to the total number of positive predictions, presents a contrasting narrative across the model versions. CNN-Yolo-4 and CNN-Yolo-6 exhibit moderate precision scores of approximately 73%, signalling a higher propensity for false positive detections, wherein the models erroneously identify objects that are not actually present in the scene. Conversely, CNN-Yolo-5, CNN-Yolo-7, and CNN-Yolo-8 showcase marked improvements in precision, with CNN-Yolo-8 nearing an impressive 99%. This ascendancy in precision suggests that the later model versions have benefited from enhanced localization strategies that effectively minimize the occurrence of false alarms, thus improving the reliability of the detection outputs. The F1-score, a harmonic mean of precision and recall, serves as a holistic metric that balances the trade-off between these two competing objectives, providing a comprehensive measure of the models' overall detection prowess. CNN-Yolo-5 registers the lowest F1-score of 0.79, primarily attributable to its subpar recall performance of 68%, indicating a tendency to miss a significant proportion of actual objects present in the scene. This deficiency may stem from overfitting during training or a lack of generalization capability, causing the model to struggle with detecting objects under varying conditions or perspectives.

In stark contrast, CNN-Yolo-7 and CNN-Yolo-8 achieve near-perfect F1-scores of 0.98, signifying optimized performance in both precision and recall, thereby attesting to

Table 2 Performance comparison table of CNN-YOLO models

Model	Accuracy (%)	Precision (%)	F1-Score	Recall (%)
CNN-Yolo-4	72	73	0.85	98
CNN-Yolo-5	72	90	0.79	68
CNN-Yolo-6	78	73	0.85	98
CNN-Yolo-7	93	97	0.98	98
CNN-Yolo-8	96	99	0.98	98

their ability to strike an optimal balance between minimizing false positives and maximizing true positive detections. Recall, which measures the proportion of actual objects correctly identified by the model, highlights a notable disparity in performance, with CNN-Yolo-5 exhibiting a significant struggle at 68%, while all other models maintain a consistent recall rate of approximately 98%. This consistency across CNN-Yolo-4, CNN-Yolo-6, CNN-Yolo-7, and CNN-Yolo-8 underscores their robust detection sensitivity and their ability to minimize the occurrence of missed objects, ensuring that the vast majority of objects present in the scene are accurately detected.

Concluding the evaluation, CNN-Yolo-8 emerges as the frontrunner, demonstrating superior performance across all evaluated metrics, including an accuracy of 96%, a precision of 99%, an F1-score of 0.98, and a recall of 98%. Conversely, CNN-Yolo-5 is identified as the weakest model due to its suboptimal recall performance. The progressive refinement observed from versions 6–8, particularly in CNN-Yolo-7 and CNN-Yolo-8, suggests that advancements in training techniques or architectural depth have led to superior classification and detection reliability, making them more suitable for deployment in real-world applications where accuracy and robustness are paramount.

The results of all the work done related to detection and classification of various plant disease is summarised in Table 3.

Although numerous methods achieve high accuracy in plant disease detection as mentioned in Table 3, they suffer from limitations that reduce their effectiveness for tomato leaf diagnostics. IoT-based RSNN methods like APDDCM-SHODL [1] involve complex deployment and are not optimized for high-resolution images, while object detectors such as enhanced YOLOv7 [2] and AKConvNet [3] lack the advanced multi-scale feature fusion of architectures like Inception-4. Crop-specific models designed for grapes [7], sugarcane [23, 24], and maize [28] require adaptation to tomato characteristics. Transformer-based approaches—CSGT [9], Swinv2-Base [21], and DeiT3-Small [24]—and tensor-analysis frameworks like HOWSVD-TEDA [4] offer robustness but tend to be computationally heavy or lack real-time feasibility. Hybrid and lightweight models such as MobileNetV3 [5], FL-ToLeD [6], EGWT [11], SE-Caps ResNet [12], zero-shot frameworks [14], PCA DeepNet [13], drone-focused CNNs [18], attention-based models [16, 19], EfficientNetV2_m [20], Res2Next50 [22], and transformer-CNN hybrids [10] offer improvements in speed or accuracy but still struggle with either resource constraints or the complexity of tomato disease patterns. In contrast, our proposed method—combining CNN Inception-4 for deep multi-scale feature extraction with YOLOv8 for precise localization and classification—effectively balances performance and computational needs, offering a robust, specialized, and real-time solution for tomato leaf disease detection in agricultural environments.

5 Conclusion and future scope

The comprehensive evaluation of deep learning models for tomato disease detection underscores the superior performance of the integrated Inception-v4 and YOLOv8 framework. This hybrid approach demonstrated high classification and localization accuracy, with average precision scores ranging from 0.81 to 0.88 across multiple disease classes. Notably, the model achieved outstanding results in detecting Bacterial Spot and Healthy leaves, suggesting strong generalization for visually distinct categories. However, it faced challenges in distinguishing between visually similar diseases like Early and

Table 3 Comparison table for the recognition of tomato plant disease with proposed methodology

References	Plant type/ disease focus	Feature extraction algorithm	Classification technique	Accuracy (%)	Precision (%)	F1 Score (%)	Recall (%)
[1]	General (IoT-Enabled)	DenseNet-201	RSNN with SHO	99.2	98.8	99.0	99.1
[2]	Tomato	Enhanced YOLOv7	YOLOv7 with Focus modules	98.8	98.9	98.8	98.7
[3]	Tomato	AKConv	AKConvNet with EIoU	99.02	98.94	98.98	98.91
[7]	Grapes	Coordinate Attention + YOLOv8	YOLOv8-ACCW	98.6	98.4	98.5	98.7
[9]	Multi-Crop	Convolution Self-Guided Transformer (CSGT)	Transformer + Convolution Blocks	98.7	98.9	98.8	98.6
[4]	Multi-Crop	HOWSVD	TEDA	98.9	98.8	98.85	98.9
[10]	General	Dual-Track Swin Transformer	DAMFN	99.4	99.1	99.2	99.3
[5]	Tomato	MobileNetV3	CNN classifier	97.8	97.6	97.7	97.9
[6]	Tomato	FL-ToLeD (Lightweight)	Lightweight CNN	99.04	98.85	98.94	99.02
[11]	Tomato	EGWT (Efficient Gradient Weighted Transformer)	CNN + Transformer Fusion	99.8	99.6	99.7	99.9
[12]	Multi	SE-SK + CapsuleNet + ResNet	SE-SK-CapResNet	99.1	98.9	99.0	99.2
[14]	Multi-Crop	Transfer Learning (Zero-Shot)	TL-based CNN	98.3	98.2	98.2	98.3
[15]	Tomato/ Cotton	Lightweight 2D CNN	Shallow CNN	96.4	96.1	96.2	96.3
[13]	General (India)	PCA + DeepNet	PCA DeepNet	98.7	98.6	98.6	98.8
[18]	Multi-Crop (Drone)	Lightweight CNN	Drone Deployable CNN	97.3	97.1	97.2	97.2
[16]	Tomato	CNN + Attention + GAN loss	Attentive GAN-CNN	98.9	98.7	98.8	98.9
[19]	Multi	Few-Shot + Adversarial CNN	Lightweight GAN	98.5	98.2	98.3	98.6
[20]	Apple	EfficientNetV2_m	CNN Classifier	99.4	99.2	99.3	99.4
[21]	Grapes	Swinv2-Base	Transformer	100	100	100	100
[22]	Tomato	Res2Next50	Enhanced CNN	99.85	99.7	99.75	99.8
[23]	Sugarcane	EfficientNet-b6, InceptionV4	Ensemble CNN	98.9	98.8	98.85	98.9
[24]	Multi	DeiT3-Small (ViT)	Vision Transformer	93.79	93.4	93.5	93.8
[25]	Multi-Crop (Review)	–	–	–	–	–	–
[26]	Multi	MobileNetV2 + ResNet50V2	GSA-enhanced CNN	99.2	99.1	99.15	99.2
[27]	Multi	MultiRes Blocks + CBAM + DSPP	CNN with Attention	99.35	99.34	99.29	99.25
[28]	Maize + Tomato	MobileNetV2 + PoolFormer	MSCPNet	97.44 (maize)/99.32 (tomato)	96.76	97.04	97.37
[29]	General	Depthwise CNN + SE	SE-Residual CNN	98.0	98.0	98.2	98.1
Proposed technique	Tomato plant disease	CNN inception 4	Yolo V8	96	99	98	98

Late Blight, indicating the need for further refinement in feature extraction or training data diversity. The Precision-Recall and threshold-based analyses revealed the model's adaptability across various operating points, with an optimal confidence threshold near 0.5. Confusion matrix analysis highlighted overfitting tendencies and misclassifications in underrepresented classes, emphasizing the importance of dataset balancing and advanced augmentation. Additionally, YOLOv8's progression over earlier versions, reflected in its exceptional accuracy (96%) and F1-score (0.98), positions it as the most effective detector in the CNN-YOLO family. Overall, the findings validate the robustness and practicality of the proposed pipeline for reliable tomato disease diagnosis in precision agriculture.

Acknowledgements
We would like to express our sincere gratitude to Sri Siddhartha University for providing the necessary research facilities and a supportive academic environment that greatly contributed to the successful completion of this work.

Author contributions
Sowmya B-conceptualization, original research writing, implementation, visualization Guruprasad S—review of the manuscript, supervision & administration.

Funding
This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data availability
Response: The labeled datasets utilized to support the findings of this study are publicly available and can be accessed at the following link: https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset.

Declarations

Ethics approval
Not applicable.

Consent to participate
Not applicable.

Consent to publish
Not applicable.

Clinical trial number
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 1 May 2025 / Accepted: 19 September 2025
Published online: 22 October 2025

References

1. Al-Shahari EA, Aldehim G, Aljebreen M, Alqurni JS, Salama AS, Abdelbagi S. Internet of things assisted plant disease detection and crop management using deep learning for sustainable agriculture. IEEE Access. 2025;13:3512–20. https://doi.org/10.1109/ACCESS.2024.3397619.
2. Umar M, Altaf S, Ahmad S, Mahmoud H, Mohamed ASN, Ayub R. Precision agriculture through deep learning: tomato plant multiple diseases recognition with CNN and improved YOLOv7. IEEE Access. 2024;12:49167–83. https://doi.org/10.1109/ACCESS.2024.3383154.
3. Ji B, et al. Lightweight tomato leaf intelligent disease detection model based on adaptive kernel convolution and feature fusion. IEEE Trans Agrifood Electron. 2024;2(2):563–75. https://doi.org/10.1109/TAFE.2024.3445119.
4. Ouamane, et al. Knowledge pre-trained CNN-based tensor subspace learning for tomato leaf diseases detection. IEEE Access. 2024;12:168283–302. https://doi.org/10.1109/ACCESS.2024.3492037.
5. Ndovie LK, Masabo E. Leveraging MobileNetV3 for in-field Tomato disease detection in Malawi via CNN. SAIEE Afr Res J. 2024;115(3):74–85. https://doi.org/10.23919/SAIEE.2024.10551304.
6. Alnamoly MH, Hady AA, Abd El-Kader SM, El-Henawy I. FI-toled: an improved lightweight attention convolutional neural network model for tomato leaf diseases classification for low-end devices,. IEEE Access. 2024;12:73561–80. https://doi.org/10.1109/ACCESS.2024.3401733.
7. Chen Z, Feng J, Zhu K, Yang Z, Wang Y, Ren M. YOLOv8-ACCW: lightweight grape leaf disease detection method based on improved YOLOv8. IEEE Access. 2024;12:123595–608. https://doi.org/10.1109/ACCESS.2024.3453379.
8. Hosny KM, El-Hady WM, Samy FM, Vrochidou E, Papakostas GA. Multi-class classification of plant leaf diseases using feature fusion of deep convolutional neural network and local binary pattern. IEEE Access. 2023;11:62307–17. https://doi.org/10.1109/ACCESS.2023.3286730.

9. Li H, Li N, Wang W, Yang C, Chen N, Deng F. Convolution self-guided transformer for diagnosis and recognition of crop disease in different environments. IEEE Access. 2024;12:165903–17. https://doi.org/10.1109/ACCESS.2024.3495529.
10. Karthik R, Ajay A, Bisht AS, Illakiya T, Suganthi K. A deep learning approach for crop disease and pest classification using Swin Transformer and dual-attention multi-scale fusion network. IEEE Access. 2024;12:152639–55. https://doi.org/10.1109/ACCESS.2024.3481675.
11. Feng J, Ong WE, Teh WC, Zhang R. Enhanced crop disease detection with EfficientNet convolutional group-wise transformer. IEEE Access. 2024;12:44147–62. https://doi.org/10.1109/ACCESS.2024.3379303.
12. Zhang X, Mao Y, Yang Q, Zhang X. A plant leaf disease image classification method integrating capsule network and residual network. IEEE Access. 2024;12:44573–85. https://doi.org/10.1109/ACCESS.2024.3377230.
13. Roy K, et al. Detection of tomato leaf diseases for agro-based industries using novel PCA DeepNet. IEEE Access. 2023;11:14983–5001. https://doi.org/10.1109/ACCESS.2023.3244499.
14. Satya Rajendra Singh R, Sanodiya RK. Zero-shot transfer learning framework for plant leaf disease classification. IEEE Access. 2023;11:143861–80. https://doi.org/10.1109/ACCESS.2023.3343759.
15. Peyal HI, et al. Plant disease classifier: detection of dual-crop diseases using lightweight 2D CNN architecture. IEEE Access. 2023;11:110627–43. https://doi.org/10.1109/ACCESS.2023.3320686.
16. Ahmed S, Hasan MB, Ahmed T, Sony MRK, Kabir MH. Less is more: lighter and faster deep neural architecture for tomato leaf disease classification. IEEE Access. 2022;10:68868–84. https://doi.org/10.1109/ACCESS.2022.3187203.
17. Sowmya BJ, et al. Leveraging machine learning for intelligent agriculture. Discov Internet Things. 2025. https://doi.org/10.1007/s43926-025-00132-6.
18. Özbılge E, Ulukök MK, Toygar Ö, Ozbılge E. Tomato disease recognition using a compact convolutional neural network. IEEE Access. 2022;10:77213–24. https://doi.org/10.1109/ACCESS.2022.3192428.
19. Wu Y, Feng X, Chen G. Plant leaf diseases fine-grained categorization using convolutional neural networks. IEEE Access. 2022;10:41087–96. https://doi.org/10.1109/ACCESS.2022.3167513.
20. Kunduracioglu I. CNN models approaches for robust classification of apple diseases. Comput Decis Mak Int J. 2024;1:235–51. https://doi.org/10.59543/comdem.v1i.10957.
21. Kunduracioglu I, Pacal I. Advancements in deep learning for accurate classification of grape leaves and diagnosis of grape diseases. J Plant Dis Prot. 2024;131:1061–80. https://doi.org/10.1007/s41348-024-00896-z.
22. Kunduracioglu I. Utilizing resnet architectures for identification of tomato diseases. J Intell Deci Mak Inf Sci. 2024;1:104–19. https://doi.org/10.59543/jidmis.v1i.11949.
23. Kunduracioğlu I, Pacal I. Deep learning-based disease detection in sugarcane leaves: evaluating EfficientNet models. J Operations Intell. 2024;2:321–235. https://doi.org/10.31181/jopi21202423.
24. Paçal İ, Kunduracıoğlu İ. Data-efficient vision transformer models for robust classification of sugarcane. J Soft Comput Decis Anal. 2024;2(1):258–71. https://doi.org/10.31181/jscda21202446.
25. Pacal I, Kunduracioglu I, Alma MH, et al. A systematic review of deep learning techniques for plant diseases. Artif Intell Rev. 2024;57:304. https://doi.org/10.1007/s10462-024-10944-7.
26. Al-Gaashani MS, Samee NA, Alkanhel R, Atteia G, Abdallah HA, Ashurov A, Muthanna MS. Deep transfer learning with gravitational search algorithm for enhanced plant disease classification. Heliyon. 2024;10(7). https://www.cell.com/heliyon/fulltext/S2405-8440(24)04998-3.
27. Al-Gaashani MSAM, Muthanna A, Chelloug SA, Kumar N. EAMultiRes-DSPP: an efficient attention-based multi-residual network with dilated spatial pyramid pooling for identifying plant disease. Neural Comput Appl. 2024;36(26):16141–61.
28. Al-Gaashani MS, Alkanhel R, Ali MA, Muthanna MS, Aziz A, Muthanna A. MSCPNet: a multi-scale convolutional pooling network for maize disease classification. IEEE Access. 2025. https://ieeexplore.ieee.org/abstract/document/10819373.
29. Ashurov AY, Al-Gaashani MSAM, Samee NA, Alkanhel R, Atteia G, Abdallah HA, et al. Enhancing plant disease detection through deep learning: a Depthwise CNN with squeeze and excitation integration and residual skip connections. Front Plant Sci. 2025;15:1505857. https://doi.org/10.3389/fpls.2024.1505857.

Publisher's Note