Beyond Classification: Benchmarking Object Detection Models for Efficient Tomato Leaf Disease Identification on a Real-World Dataset

Fahim Mahafuz Ruhad[a,b], Md Fahim[b], Mir Sazzat Hossain[b], Md. Fahad Monir[b], Ashraful Islam[b,*], M. Ashraful Amin[b]

[a] *Department of Agricultural Construction and Environmental Engineering, Sylhet Agricultural University, Sylhet, 3100, Bangladesh*
[b] *Center for Computational & Data Sciences, Independent University, Bangladesh, Dhaka, 1229, Bangladesh*

Abstract

Tomato is one of the most economically significant horticultural crops worldwide, yet its cultivation is frequently hindered by foliar diseases such as Late Blight, Bacterial Spot, and Septoria Leaf Spot. These diseases not only reduce yield but also result in substantial financial losses, particularly for smallholder farmers. Traditional manual inspection methods are labor-intensive and error-prone, highlighting the need for automated and accurate disease detection systems. While image classification models have been widely explored for plant disease identification, they often fail to capture spatial symptom details, leading to reduced interpretability and accuracyespecially for visually ambiguous diseases. In this study, we propose an object detection-based approach to localize and identify tomato leaf diseases more effectively. To facilitate this, we introduce a novel, regionally diverse dataset comprising nine common tomato diseases collected across three Bangladeshi regions over three months. We benchmark a range of classification and object detection models, demonstrating that YOLO-v9 achieves superior performance (F1-score: 97.86, mean Average Precision (mAP): 79.44). Furthermore, for real-time deployment on resource-constrained edge devices, we present YOLO-v8-CSwin, a lightweight model with competitive accuracy and significantly fewer parameters (11.12M). Our findings highlight the advantages of object detection over classification for plant disease diagnosis and offer practical insights for deploying AI-driven solutions in real-world agricultural settings.

Keywords: Classification, Detection, Leaf Detection, Edge Computing

1. Introduction

Tomato is one of the most widely cultivated and economically vital horticultural crops worldwide. It plays a critical role in global food systems, contributing to food security, nutrition, and the livelihoods of rural populations. In Bangladesh, tomato cultivation has experienced significant growth, with production

*Corresponding author
Email address: ashraful@iub.edu.bd (Ashraful Islam)

projected to exceed 487 thousand metric tons annually by 2026[1]. Major tomato-producing regions include Sylhet, Cumilla, and Chittagong. However, the yield and quality of tomato crops are consistently threatened by a wide range of foliar diseases, such as Late Blight, Bacterial Spot, and Septoria Leaf Spot [1]. These diseases manifest through symptoms like leaf discoloration, wilting, defoliation, and fruit rot, ultimately impairing plant health and productivity. Early and precise detection of such diseases is essential for minimizing crop losses and improving the efficiency of agricultural resource allocation. Additionally, timely intervention supports environmental sustainability by reducing excessive pesticide use and associated pollution. For smallholder farmers, these diseases not only lower crop yields but also lead to considerable economic hardship.

The critical importance of early disease detection was underscored during the devastating tomato epidemic in South Asia and India[2], where an undiagnosed outbreak of Late Blight decimated crops across multiple regions, leading to widespread supply chain disruptions and sharp increases in market prices [2]. Traditional manual inspection methods are often time-consuming, inconsistent, and prone to human error, rendering them ineffective for real-time disease management. Consequently, automated disease diagnosis systems leveraging computer vision and machine learning have gained substantial attention in recent years [3, 2]. These systems enable disease classification or detection directly from leaf images using deep learning techniques.

Although extensive research has been conducted on image classification methods for plant disease identification [4, 3, 5, 1, 6], these approaches are inherently limited. Image classifiers generally assign a single label to an entire image, often disregarding the spatial characteristics of disease symptoms. In our experiments, even advanced classification models like ConvNext and ViT-Large achieved strong F1-scores on clearly defined diseases such as Late Blight but struggled with visually ambiguous categories like Mosaic Virus and Yellow Leaf, with F1-scores dropping to as low as 0.52. Grad-CAM visualizations further revealed that these models frequently focused on irrelevant background areas, leading to misclassification and reduced interpretability.

To address these shortcomings, object detection has emerged as a more suitable alternative, especially for tasks that require spatial localization of symptoms. Unlike classification, object detection models can both identify and locate disease-affected regions within an image. The YOLO (You Only Look Once) family of models, in particular, has demonstrated strong performance across a variety of agricultural applications [1, 7, 8, 9]. However, there remains a research gap in comprehensive studies evaluating different state-of-the-art object detection models specifically for tomato leaf disease. To bridge this gap, we developed a novel and robust dataset specifically for tomato leaf disease detection. The dataset comprises samples collected across three distinct agro-climatic regions of BangladeshSylhet, Cumilla, and Chittagongover the months of

[1] [https://www.reportlinker.com/clp/country/484797/726290

[2] [https://openknowledge.fao.org/server/api/core/bitstreams/30c0d98d-1c21-48ef-b5d9-8d988e6fa6f2/content

November to January. It includes nine common tomato foliar diseases, annotated with bounding boxes to facilitate both classification and detection tasks.

We conducted a thorough evaluation of both classification and detection approaches using this dataset. Our results show that YOLO-v9 outperforms earlier detection models such as Faster R-CNN and RetinaNet, achieving an F1-score of 97.86 and mAP of 79.44. In contrast to classification models, YOLO-v9 offers enhanced interpretability by accurately localizing disease-affected areas, as confirmed by improved Grad-CAM visualizations.

Despite the high performance of YOLO-v9, its deployment in real-world agricultural settingsoften constrained by limited connectivity, low computational resources, and cost concernsposes practical challenges. To enable field-ready implementation, lightweight models suitable for edge devices such as smartphones and drones are essential. In response, we experimented with parameter-reduction strategies using different YOLO backbones. Notably, the YOLO-v8 model integrated with the CSwin Transformer backbone (YOLO-v8-CSwin) achieves competitive performance with an F1-score of 93.20, while reducing the parameter count to just 11.12 millionless than half that of YOLO-v9. This makes YOLO-v8-CSwin a promising solution for real-time, on-device inference. In this study, our key contributions can be summarized as follows:

1. **Dataset Creation**: We introduce a novel object detection dataset for tomato leaf diseases, compiled from three agro-climatic regions in BangladeshSylhet, Cumilla, and Chittagongspanning November to January. The dataset features nine commonly observed foliar diseases with precise bounding box annotations.
2. **Classification Benchmarking**: We evaluate a range of CNN and Transformer-based classification models, such as ConvNext and ViT-Large. Although they attain reasonable average F1-scores (up to 0.76), performance on visually ambiguous classes remains limited, with reduced interpretability as evidenced by Gradient-weighted Class Activation Mapping (Grad-CAM) analysis.
3. **Detection Benchmarking**: We compare the performance of classical (e.g., Faster R-CNN, Mask R-CNN) and modern (e.g., YOLO-v3 to YOLO-v9) detection models. YOLO-v9 achieves the highest mAP and F1-score, establishing itself as the most effective architecture in our evaluation.
4. **Edge Optimization**: To support real-world deployment, we experiment with lightweight YOLO variants for edge computation. Our results show that YOLO-v8-CSwin offers an optimal trade-off between accuracy and model size, enabling efficient inference on resource-constrained devices.

2. Related Work

ML Approaches for Plant Disease Detection. The application of machine learning and deep learning techniques to plant disease detection has received significant attention in recent years. Early approaches often utilized Convolutional Neural Networks (CNNs) in combination with other classifiers. For instance,

Çuu et al. [4] extracted leaf features using CNNs and trained a Support Vector Machine (SVM) to classify 57 tree species, achieving high accuracy and highlighting the effectiveness of hybrid models. Similarly, Mohanty et al. [5] used GoogleNet and AlexNet on a dataset of 54,306 plant disease images, reaching an accuracy of 99.3%. However, they noted that poor background contrast can reduce classification effectiveness, emphasizing the importance of high-quality datasets. Hu et al. [3] modified the CIFAR10-quick model for detecting tea leaf diseases and achieved an average accuracy of 92.25%, showcasing the potential of lightweight deep learning models in agriculture. These studies collectively underscore the viability of CNN-based models for plant disease detection and the critical role of dataset quality and model optimization in achieving robust results.

ML Models for Tomato Disease Detection. Focusing specifically on tomato diseases, Brahimi et al. [1] proposed a deep learning classifier trained on 14,828 images covering nine tomato leaf conditions, achieving a classification accuracy of 99.18%. They also employed occlusion techniques for disease localization, contributing to methods for both identification and spatial diagnosis of plant diseases. Huang et al. [6] developed a complex fuzzy Mask R-CNN model for evaluating cherry tomato maturity stages. Integrating the Hough transform, Mask R-CNN, and fuzzy c-means algorithms, the model reached an accuracy of 98.00%. However, the high computational demand of this approach limits its practicality for real-time field applications.

Object Detection Models in Agricultural Disease Identification. Recent research has increasingly leveraged object detection architectures, particularly the YOLO family of models, for plant disease identification. Liu and Wang [7] improved the YOLO-v3 model by incorporating image pyramids, object bounding box clustering, and multi-scale training. Their enhanced model achieved 92.39% accuracy with a fast processing time of 20.39 ms, making it suitable for real-time agricultural environments. Qi et al. [8] further advanced this line of work by introducing SE-YOLOv5, an improved version of YOLOv5, achieving 91.07% accuracy and 94.10% mean average precision on a dataset of tomato disease images captured via mobile phones. Their model demonstrated practical applicability in field conditions. In a more recent study, Soeb et al. [9] applied YOLO-v7 for detecting tea leaf diseases, training on 4000 images collected from tea gardens in Bangladesh. The model achieved 97.3% accuracy and aimed to assist entomologists while improving agricultural productivity in developing regions.

3. TomDet : Dataset Creation

3.1. Image Collection

To develop a robust tomato leaf disease detection model, a diverse image dataset was collected from three agricultural regions in Bangladesh: Sylhet, Cumilla, and Chattogram. These regions were selected due to their active involvement in tomato cultivation and the availability of different environmental conditions that

may influence disease manifestation. In Bangladesh, tomato cultivation typically occurs between November and March, aligning with the cooler, dry season that is suitable for tomato growth. For this study, images were primarily collected during the months of November, December, and January, when plants are in active growth and more susceptible to visible disease symptoms.

Images were captured using a variety of mobile devices to ensure variability in image quality and resolution, reflecting real-world application scenarios. Specifically, photos were taken with mobile phones including the Samsung Galaxy A52, Xiaomi Redmi Note 11, and Realme Narzo 50. This diversity in image sources helps simulate conditions under which farmers or agricultural workers might collect data in practice. Finally, 6410 images were collected at the end of the process.

3.2. Data Filtering and Cleaning

After collecting images from various mobile devices and online sources, several filtering and cleaning steps were applied to ensure dataset quality and consistency. Low-quality or blurry images, common in field conditions, were removed through both manual inspection and an automated Laplacian variance method. Duplicate images were identified and removed from the dataset to ensure data quality. In addition, images that either contained only background or were significantly blurred were excluded, as they could negatively impact the accuracy and reliability of the model.

To minimize distractions caused by complex field backgrounds such as soil, hands, or tools, basic cropping and background reduction techniques were applied when possible, with some images processed using simple segmentation to isolate leaf regions. All images were standardized to the RGB color space and saved in JPEG format, while histogram equalization was selectively applied to improve contrast in underexposed images. Following the data filtering and cleaning process, a total of 850 images were removed. As a result, the final dataset consists of 5,560 data samples.

3.3. Disease Categorization

Tomato plants are susceptible to a wide range of diseases and pests that impact leaf health and, consequently, crop productivity. These diseases are commonly categorized into four major groups based on their causal agents: bacterial, fungal, viral, and pest-induced conditions [10], [11], [12]. This categorization is widely accepted in plant pathology and supports the development of targeted detection and control strategies. In this project, the dataset includes images of tomato leaves affected by nine common conditions to facilitate comprehensive disease classification.

- **Bacterial Spot:** Caused by *Xanthomonas campestris* pv. *vesicatoria*, bacterial spot presents as water-soaked lesions that may become necrotic. It thrives in warm, humid climates and can rapidly spread under favorable conditions, leading to significant leaf damage and yield loss [13].

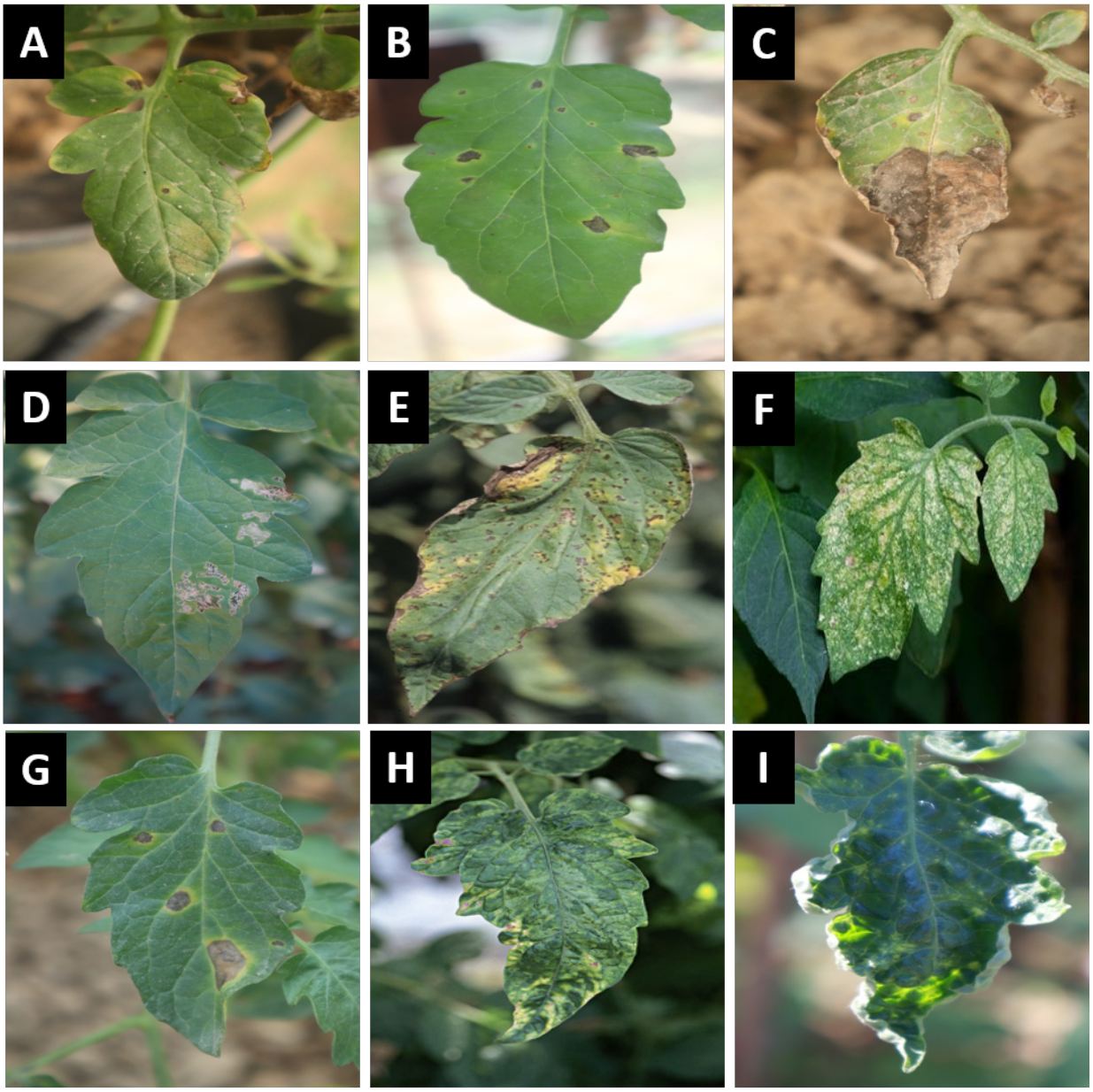


Figure 1: Images of tomato leaf diseases: (a) Bacterial Spot, (b) Early Blight, (c) Late Blight, (d) Leaf Mold, (e) Septoria Leaf Spot, (f) Spider Mites, (g) Target Spot, (h) Mosaic Virus, (i) Yellow Leaf Curl

- **Fungal Diseases Early and Late Blight:**
  - *Early blight*, caused by *Alternaria solani*, appears as dark concentric rings on older leaves and can lead to defoliation.
  - *Late blight*, caused by *Phytophthora infestans*, leads to rapidly spreading water-soaked lesions and is considered one of the most destructive tomato diseases.

Without timely intervention, both can significantly reduce plant vigor and fruit yield [14].

- **Leaf Mold:** Induced by the fungus *Cladosporium fulvum*, leaf mold is characterized by yellow spots on the upper leaf surface and olive-green mold on the underside. It thrives in greenhouse conditions and can negatively affect both fruit quality and yield [15].
- **Septoria Leaf Spot:** *Septoria lycopersici* causes small, circular lesions with tan centers and dark margins, primarily on lower, older leaves. This leads to premature defoliation, reducing photosynthesis and fruit development [16].
- **Target Spot:** Caused by *Corynespora cassiicola*, target spot produces distinctive concentric rings on the leaf surface. The disease can persist in crop residues and is known for its increasing resistance to fungicides, making it a challenge for integrated disease management [17].
- **Pest Damage Spider Mites:** Infestation by spider mites, primarily *Tetranychus urticae*, causes stippling, yellowing, and webbing on leaves. These arachnid pests feed on plant sap, leading to reduced vigor and potential yield loss during heavy infestations [18].
- **Viral Diseases Mosaic Virus and Yellow Leaf Curl:**
  - *Tomato Mosaic Virus (ToMV)* belongs to the *Tobamovirus* genus and causes mottled leaf patterns, deformation, and stunted growth.
  - *Tomato Yellow Leaf Curl Virus (TYLCV)* is a *Begomovirus* transmitted by whiteflies, characterized by leaf curling, yellowing, and reduced fruit set.

  These viral infections are highly contagious and can significantly impair plant productivity [19, 20].

Figure 1 presents a representative image for each disease in our dataset for visualization purposes. This helps provide a visual understanding of the variations and characteristics associated with each tomato disease.

Why did we exclude healthy leaves? We excluded healthy leaves from our dataset because our detection models, like the YOLO-based models, were specifically designed to detect and classify diseased lesions on tomato leaves, not healthy tissue. In object detection, regions without labels (such as healthy tissue or background) are treated as negative data [21, 22], so no healthy class was assigned. This approach, which focuses the model on symptomatic regions, is supported by recent studies. For instance, Wang et al. [23] excluded healthy-leaf images from their tomato dataset, as healthy leaves are not relevant for disease detection. Similarly, in medical imaging, a YOLO-based breast cancer detector was trained exclusively on lesions, with healthy images omitted [21]. These examples demonstrate that training object detectors on anomalies alone, while treating normal tissue as background, is both common and effective. Our results confirm that omitting healthy samples did not impact model performance.

Disease Name	Kappa(κ)
Bacterial Spot	0.80
Early Blight	0.72
Late Blight	0.89
Leaf Mold	0.78
Septoria Leaf Spot	0.82
Spider Mites	0.84
Target Spot	0.78
Mosaic Virus	0.73
Yellow Leaf	0.85
Avg	0.801

Table 1: Inter-Annotator Agreement Score

3.4. Data Annotation

To ensure high-quality annotations for both classification and object detection tasks, a structured manual labeling process was implemented. Two undergraduate students from a reputed agricultural university in Bangladesh, both with foundational knowledge in plant pathology and tomato diseases, were hired as annotators. Each annotator was compensated at a rate of BDT 3 per image, reflecting fair remuneration for their time and effort.

The annotation process was divided into two stages: image-level classification and region-level detection. In the first stage, annotators were tasked with identifying the disease category present in each image based on visible symptoms such as spots, discoloration, mold, and deformation. This classification followed the disease categories described in Section 3.3. Once classification was complete, annotators proceeded to the object detection stage, where they were instructed to draw bounding boxes around the affected areas of the leaf. For this purpose, we used the open-source Python-based annotation tool `labelImg`, which is commonly used for object detection tasks due to its simplicity and compatibility with popular deep learning frameworks such as YOLO.

To minimize background noise and ensure that the bounding boxes focused on the actual diseased regions, annotators were advised to enclose only the visibly infected portions of the leaf. In cases where symptoms were spread across multiple parts of the leaf, multiple bounding boxes were used. The final annotations were saved in both YOLO-format text files (for detection) and structured CSV files (for classification), ensuring compatibility with downstream training pipelines.

3.5. Annotation Verification

To verify the accuracy of the annotations, we randomly selected 1,000 samples and asked both annotators to independently label them according to the identified disease categories. To evaluate the consistency and quality of these annotations, we computed the inter-annotator agreement using Fleiss's Kappa score [24], a widely accepted statistical measure for assessing reliability among multiple raters. The agreement scores for each disease category are presented in Table 1. Notably, all scores were above 0.62, with an average agreement score of 0.801. According to the interpretation guidelines in [24], an agreement score of 0.62 or higherparticularly when considering multiple annotatorsindicates a strong level of consistency across the dataset.

4. TomDet : Data Analysis

In this section, we present a comprehensive analysis of our dataset, which consists of a total of 5,560 annotated images of tomato leaves affected by various diseases. The data has been explored from three perspectives: disease-wise, region-wise, and month-wise. This multi-dimensional analysis provides deeper insights into the spatial and temporal patterns of disease occurrence. Detailed discussions are provided below.

Disease Distribution. The disease distribution is illustrated using a donut chart in Figure 2. The dataset contains nine types of tomato leaf diseases. Among these, Early Blight and Late Blight are the most common, with 1,112 and 1,001 samples respectively, together representing approximately 38 percent of the dataset. These diseases are typically widespread in tropical climates, which aligns with their prominence in our data.

Septoria Leaf Spot and Spider Mites follow with 834 and 667 samples respectively. Mosaic Virus (556 samples), Yellow Leaf Curl Virus (500 samples), and Leaf Mold (334 samples) show moderate presence. Target Spot and Bacterial Spot are the least represented, with 278 samples each, comprising around five percent of the total.

Region-wise Distribution. The region-wise sample distribution is shown in Figure 3a, with data collected from three major agricultural regions in Bangladesh: Sylhet (2,002 samples), Chattogram (1,890 samples), and Cumilla (1,668 samples). A more detailed region-wise disease distribution is presented in Figure 4.

From the figures, it is evident that Sylhet experienced the highest concentration of fungal diseases, particularly Early Blight (400 samples) and Late Blight (360 samples), likely due to its cooler and more humid early winter climate. Chattogram displayed a relatively balanced distribution of all diseases, with moderate counts across the board, suggesting more stable agro-climatic conditions. Cumilla, on the other hand, exhibited increased occurrences of viral and bacterial diseases, such as Yellow Leaf Curl Virus (150

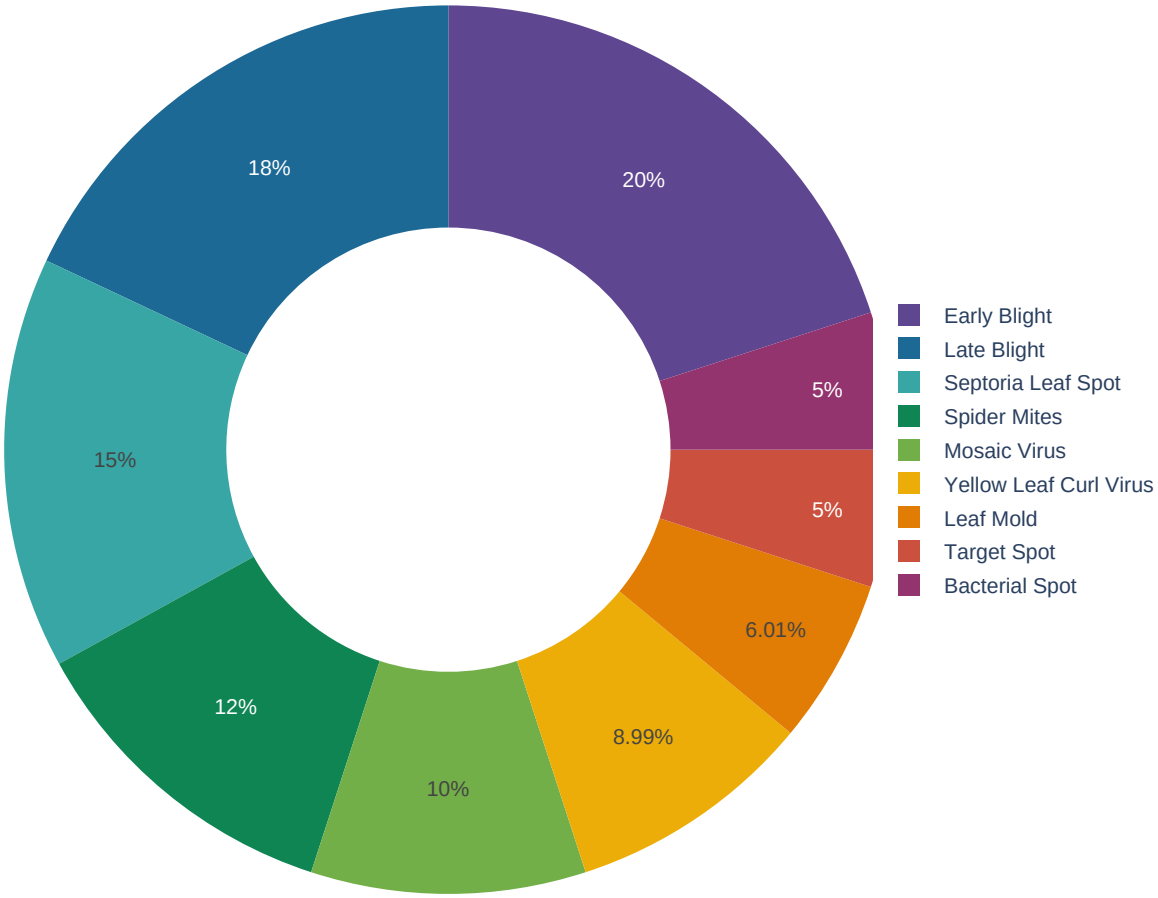
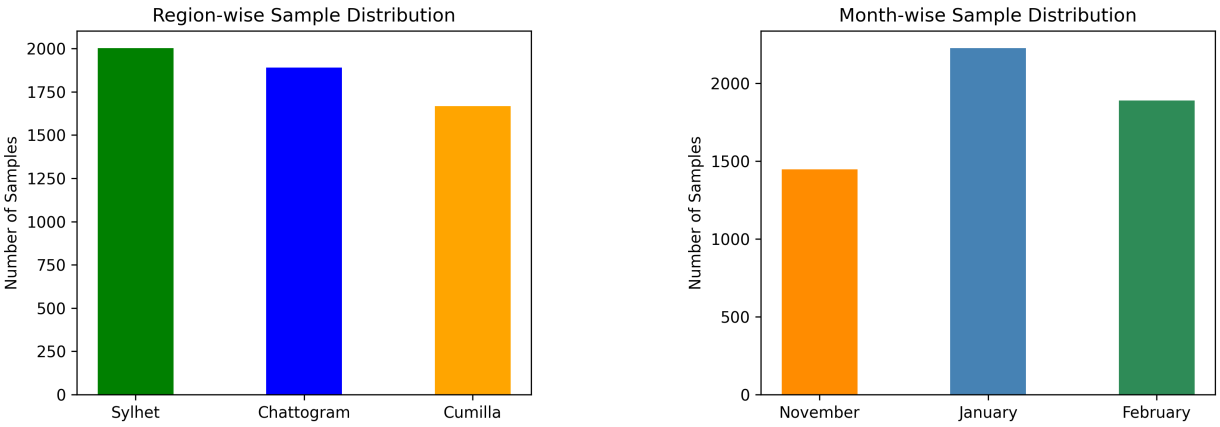


Figure 2: Tomato Disease distribution of TomDet .

samples) and Bacterial Spot (100 samples). These patterns may be influenced by localized agricultural practices, crop varieties, or environmental conditions.

From Figure 4, Sylhet showed the highest incidence of blight diseases, particularly Early and Late Blight, with 400 and 360 samples, respectively. Chattogram displayed a relatively balanced distribution, while Cumilla had notably higher occurrences of Yellow Leaf Curl Virus (150 samples) and Bacterial Spot (100 samples), suggesting potential environmental or crop management factors influencing disease patterns. Cross-sectional visualizations revealed additional insights: Sylhet had consistently higher counts of fungal diseases, especially during the early winter months. Chattogram exhibited more stable distribution across most diseases, possibly due to more balanced agro-climatic conditions. Cumilla showed elevated viral and bacterial diseases, possibly linked to local agricultural practices or varietal susceptibility.

Month-wise Distribution. The temporal distribution of the samples is illustrated in Figure 3b, with the highest number of samples recorded in January (2,225), followed by February (1,890) and November (1,446).



(a) Region-wise Data Sample Distribution

(b) Month-wise Data Sample Distribution

Figure 3: Data Sample Distribution for Our TomDet Dataset

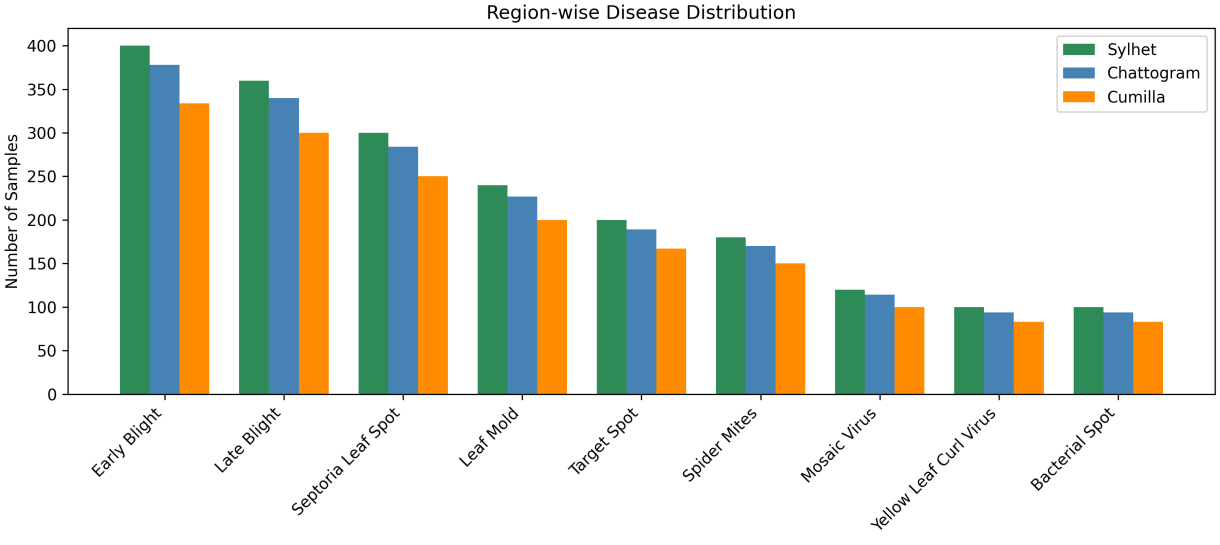


Figure 4: Region-wise Disease Distribution Plot

Further analysis of month-wise disease distribution is presented in Figure 5.

Blight-type diseases, such as Early Blight and Late Blight, were more commonly observed in January, with 445 and 401 samples, respectively, followed by November with 289 and 260 samples. These months are typically cooler and more humid, creating favorable conditions for fungal infections. Leaf Mold and Target Spot showed increased presence in February, with 170 and 114 samples, respectively, suggesting their tendency to emerge later in the winter season. Viral infections, including Yellow Leaf Curl Virus and Mosaic Virus, maintained a mild but consistent presence across all months, with moderate counts in each month: 130-145 samples in November, 200-222 samples in January, and 170-189 samples in February, indicating their broader environmental adaptability.

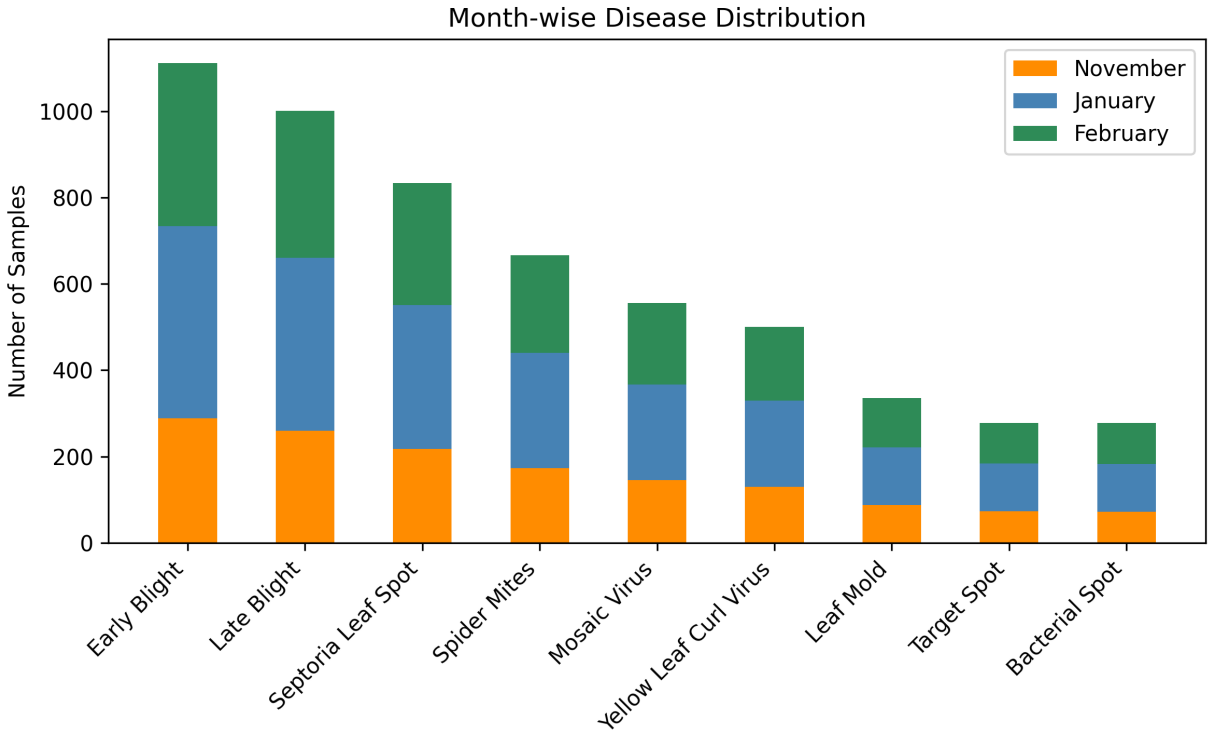


Figure 5: Month-wise Disease Distribution Plot

Dataset Split. For training and evaluating the models on our dataset, we divided the data into two subsets: (i) training set and (ii) testing set, following an 80:20 split ratio. To ensure that the distribution of disease categories was preserved in both subsets, we employed stratified sampling. Specifically, the stratified split was performed independently for each disease category to maintain proportional representation in both training and test sets. As a result, the final dataset consisted of 4,445 samples in the training set and 1,115 samples in the test set.

5. Methodology

In our study, we initially evaluate various classification models and report the cases where they fail. We then experiment with different detection models. Additionally, we fine-tune the model backbone in the detection models to improve their speed and enhance their usability on end devices. In this section, Section 5.1 outlines the necessary steps for data preprocessing, while Section 5.2 provides details on the classification models. Section 5.3 focuses on the detection models and evaluation metrics, and Section 5.4 and Section 5.5 present the experimental setup and results, respectively. An overview of our model architecture is depicted in Figure 6.

5.1. Preprocessing

Our preprocessing process consists of two main steps: i) Resizing and ii) Augmentation. Each model requires a specific input image size. For an image I, it is resized to $H \times W$, typically 224×224 if the model does not have a specific resizing requirement, resulting in $I \in \mathbb{R}^{H \times W \times C}$. We then apply a variety of augmentation techniques $Aug(\cdot)$, such as random blur, contrast adjustment, and grayscale conversion, to enhance the diversity of the training data. After applying the augmentation techniques, the resulting augmented image is $I_i^{'} = Aug(I_i)$. This augmented image is then passed through the model to obtain predictions and detections. These augmentations help the model generalize better and improve its performance. A total of four different augmentation techniques are randomly applied to each image in the training set.

- **Random Blur:** This linear filter blurs the image. It was applied with a probability of 0.01 and a blur limit ranging from 3 to 7. This technique helps to reduce noise in the image and improve the model's performance.
- **Median Blur:** This is another blurring technique; unlike the random blur, it is a non-linear filter that removes noise from the image. While blurring the image, it retains the edges of the objects in the image. It was applied with the same probability and limit as the random blur.
- **Random Grayscale:** This technique converts the image to grayscale with a probability of 0.01. This helps the model to learn better features from the image.
- **CLAHE:** Contrast Limited Adaptive Histogram Equalization (CLAHE) is a technique used to improve the image's contrast by redistributing the intensity values. It was applied with a probability of 0.01, a clip limit ranging from 1 to 4.0, and a tile grid size 8×8. This technique helps to improve the image's contrast and make the objects more distinguishable.

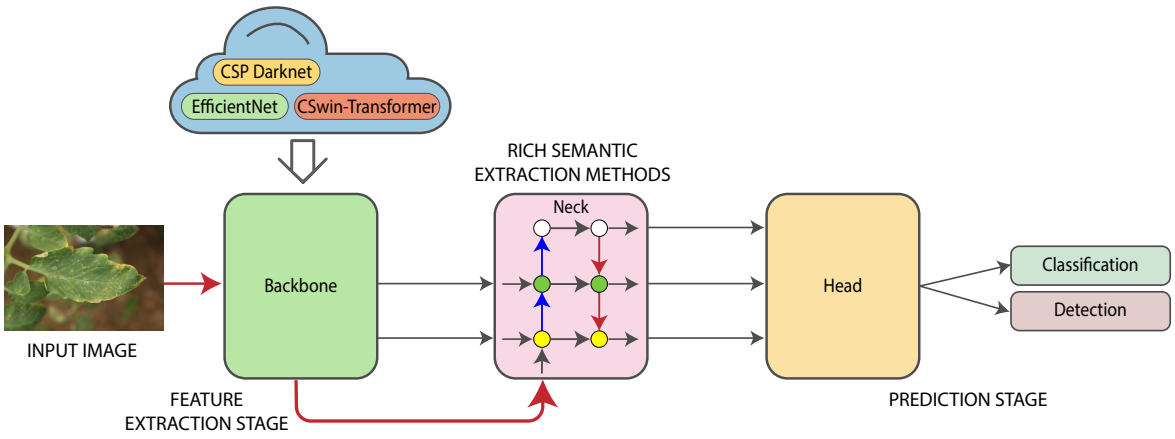


Figure 6: Model Architecture Diagram

5.2. Classification Models

This section describes the training procedure for the image classification models used in our study. The classification model, denoted as θ_{clf}, is composed of two primary components: the backbone B and the classifier head H. The backbone B processes the input image I, extracting high-level feature maps $F_B(I)$. Following the extraction of feature maps by the backbone, an aggregation method is applied to reduce the spatial dimensions and condense the information into a compact vector. We employ Global Average Pooling (GAP) for this purpose. GAP computes the average value of each feature map (or channel) across all spatial dimensions (height and width). This operation results in a fixed-length feature vector for each image, independent of its original size. Mathematically, for a feature map $F_B(I) \in \mathbb{R}^{H \times W \times C}$, the GAP operation is defined as:

$$\hat{F}_{\text{GAP}} = \frac{1}{H \times W} \sum_{h=1}^{H} \sum_{w=1}^{W} F_B(I)_{h,w,c} \quad \forall c \in \{1, \ldots, C\}$$

This produces a vector $F_{\text{agg}} = \hat{F}_{\text{GAP}} \in \mathbb{R}^C$, where each element represents the average across all spatial locations within that channel. For Vision-Transformer (ViT) models, we use the [CLS] token representation as the aggregated feature $F_{\text{agg}} = F_B(I)^{[\text{CLS}]}$, which is then passed into the classifier head.

After feature aggregation, the resulting vector $\hat{F}_{\text{GAP}}$ from GAP is input into the classifier head H, a fully connected (FC) layer. The classifier head transforms the aggregated features into class probabilities $\hat{p} = H(\hat{F}_{\text{GAP}}) = [\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_N]$, where N is the number of classes. The model's parameters θ_{clf} are trained to minimize the cross-entropy loss, which measures the difference between the predicted probabilities $\hat{p}$ and the true class label y. For a given image with true class y, the loss is computed as: $L_{\text{CE}} = -\log(\hat{p}_y)$ where $\hat{p}_y$ is the predicted probability for the true class y.

5.3. Object Detection

This section describes the training procedure for the object detection models used in our study. The object detection model, denoted by θ_{det}, is composed of two main components: a backbone B and a detection head D. The backbone B processes the input image I, extracting feature maps $F_B(I)$, which serve as the basis for object detection.

The detection head D is responsible for both classifying objects and predicting bounding boxes. The output of the detection head consists of two parts: i) Class scores for each detected object, which are probabilities that the object belongs to a particular class. ii) Bounding box coordinates which are continuous values indicating the position and dimensions of the predicted bounding boxes. The detection model uses Region Proposal Networks (RPNs) or similar methods to propose candidate bounding boxes for each object in the image. These candidate boxes are then refined and classified by the detection head. For each proposed bounding box, we calculate two main types of losses: classification loss and regression loss.

The classification loss quantifies the difference between the predicted class probabilities $\hat{p}$ for each bounding box and the ground truth class labels y. The classification is typically treated as a multi-class classification problem, with each proposal being assigned a probability distribution over the possible object classes. Mathematically, the classification loss L_{cls} is defined using cross-entropy: $L_{\text{cls}} = -\sum_{c=1}^{C} y_c \log(\hat{p}_c)$. The regression loss is used to measure how well the predicted bounding boxes match the ground truth boxes. We use a smooth L1 loss function to compute this loss, which combines the advantages of L2 loss (for small errors) and L1 loss (for larger errors). For each bounding box prediction, the regression loss L_{reg} is computed as:

$$L_{\text{reg}} = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L1}(t_i - \hat{t}_i)$$

where t_i are the ground truth bounding box coordinates, $\hat{t}_i$ are the predicted bounding box coordinates.

This loss encourages accurate predictions for both the position (x, y) and size (w, h) of the bounding boxes. The total loss L_{det} for the object detection model is a combination of the classification loss and the regression loss:

$$L_{\text{det}} = L_{\text{cls}} + \lambda_{\text{reg}} L_{\text{reg}}$$

where λ_{reg} is a hyperparameter that balances the importance of the classification loss and the regression loss.

During inference, the model generates a set of predicted bounding boxes with associated class scores. To avoid multiple detections of the same object, we apply Non-Maximum Suppression (NMS). NMS filters out duplicate detections by retaining only the bounding boxes with the highest class score, and removing those that overlap largely with higher-scoring boxes based on an Intersection over Union (IoU) threshold.

5.4. Experimental Setup and Evaluation Metrics

Experiment Setup For the classification model experiments on our dataset, we evaluate several well-known architectures, including VGG [25], ResNeXt [26], ConvNeXt [27], and the Vision Transformer (ViT) [28]. For the object detection models, we consider a range of state-of-the-art approaches: DETR [29], Faster R-CNN [30], Mask R-CNN [31], EfficientDet [32], RetinaNet [33], and various YOLO models with different model backbones.

We adhere to the recommended hyperparameter settings for each model. Specifically, for CNN-based models, we use a learning rate of 1e-3, while for the transformer-based models, such as ViT, we use a learning rate of 2e-5. The models are trained using the AdamW optimizer with a weight decay of 0.0005 to prevent overfitting and improve generalization. All models are trained with a batch size of 16. For the classifier head, we employ a two-layer multi-layer perceptron (MLP) architecture with ReLU activation functions. The trainings of the models were performed on a single NVIDIA Tesla T4 GPU with 16 GB of memory. We used several libraries for our experiment, including PyTorch, OpenCV, Albumentations, Matplotlib, and Seaborn.

Evaluation Metrics After training the models, we evaluated their performance on the test set using the following metrics: Intersection over Union (IoU), Precision, Recall, and F1-score.

- ***Intersection over Union (IoU)***: IoU measures the overlap between the predicted bounding box B_{pred} and the ground truth bounding box B_{gt}. It is defined as:

$$\text{IoU}(B_{\text{pred}}, B_{\text{gt}}) = \frac{|B_{\text{pred}} \cap B_{\text{gt}}|}{|B_{\text{pred}} \cup B_{\text{gt}}|}$$

  where $|\cdot|$ denotes the area of the bounding boxes. Higher IoU values indicate better localization.

- ***Precision***: Precision is the ratio of true positive predictions TP to the total number of predicted positives $TP + FP$ (true positives + false positives) A higher precision means fewer false positive predictions.

- ***Recall***: Recall is the ratio of true positive predictions TP to the total number of ground truth positives $TP+FN$ (true positives + false negatives) A higher recall indicates that the model is good at detecting all true objects.

- ***F1-Score***: The F1-score is the harmonic mean of precision and recall. The F1-score balances both precision and recall, providing a single measure that reflects both false positives and false negatives.

- ***mean Average Precision (mAP)***: The mAP is a widely used metric in object detection. It is calculated by averaging the Average Precision for each class across different IoU thresholds. The mAP is then the mean of the AP values across all classes:

$$\text{mAP} = \frac{1}{C} \sum_{c=1}^{C} \text{AP}_c$$

where C is the number of classes. mAP is a comprehensive evaluation that accounts for both the detection accuracy and the precision-recall trade-off at various IoU thresholds.

These metrics provide a comprehensive evaluation of both classification and localization performance, allowing us to assess how well the model generalizes to the test set.

6. Result Analysis

6.1. Classification Results

Table 2 presents the F1-score performance of various classification models evaluated on the test split of the TomDet dataset. The analysis reveals distinct trends both across disease categories and model architectures, offering key insights into the relative difficulty of each classification task and the strengths of different model types.

Disease-wise. Table 2 reports the F1-score performance of various CNN and Transformer-based classification models on the test split of the TomDet dataset, across nine disease categories. Among all classes, Late Blight consistently emerges as one of the most accurately predicted diseases, with top F1-scores achieved by ConvNext (0.82) among CNNs and ViT-Large (0.83) among Transformer-based models. This suggests that Late Blight presents distinctive visual patterns that generalize well across both convolutional and transformer-based architectures. In contrast, Mosaic Virus (MV) and Yellow Leaf (YL) exhibit comparatively lower performance across most models. MV scores range from 0.52 (VGG-16) to 0.78 (ConvNext), while YL ranges from 0.56 (VGG-16) to 0.67 (ConvNext), indicating these classes may be more visually ambiguous or affected by higher intra-class variability.

Classes like Spider Mites, Septoria Leaf Spot (SLS), and Leaf Mold generally show moderate F1-scores (between 0.630.75) across most models, reflecting a balance of distinguishability and challenge. Notably, ConvNext outperforms other CNN-based models, achieving the highest average F1-score (0.76), while ViT-Large leads among Transformer-based models with an average of 0.74, suggesting strong feature extraction capabilities in deeper transformer architectures.

Model-wise Insights. When comparing model families, Transformer-based architectures demonstrate a clear advantage over CNN-based models in terms of average F1-score. ViT-Large achieves the highest average score of 0.76, outperforming the best CNN model, ConvNext (0.76), and notably ahead of earlier CNNs like VGG-16 (0.63).

Within the CNN family, a clear performance progression is observed, moving from VGG-16 to ResNet-18, ResNet-50, and ConvNext. This improvement reflects the ability of deeper and more modern architectures to better capture the complex visual patterns associated with tomato diseases, such as the subtle discolorations in Septoria Leaf Spot or the irregular lesions of Bacterial Spot. A similar trend is evident among Transformer-

Table 2: The benchmarking of classification models on the test split of the TomDet dataset is reported in terms of F1-score. Here, BS refers to Bacterial Spot, Early to Early Blight, Late to Late Blight, Mold to Leaf Mold, SLS to Septoria Leaf Spot, Mites to Spider Mites, TS to Target Spot, MV to Mosaic Virus, and YL to Yellow Leaf.

Models	**Categories**									
	BS	**Early**	**Late**	**Mold**	**SLS**	**Mites**	**TS**	**MV**	**YL**	**Avg**
CNN Based										
VGG-16	0.62	0.72	0.69	0.63	0.64	0.61	0.68	0.52	0.56	0.63
ResNet-18	0.67	0.77	0.74	0.68	0.69	0.66	0.73	0.57	0.61	0.68
ResNet-50	0.73	0.79	0.81	0.68	0.75	0.73	0.75	0.62	0.64	0.72
ConvNext	**0.79**	**0.82**	**0.86**	**0.74**	**0.72**	**0.76**	**0.78**	**0.67**	**0.69**	**0.76**
Transformer Based										
ViT-Small	0.71	0.74	0.72	0.65	0.68	0.63	0.74	0.60	0.58	0.67
ViT-Base	0.76	0.74	0.80	0.68	0.71	0.69	0.76	0.64	0.63	0.71
ViT-Large	0.76	0.80	0.83	0.71	0.76	0.75	0.77	0.62	0.64	0.74

based models, where classification accuracy increases from ViT-Small (F1-score: 0.67) to ViT-Base and reaches its peak with ViT-Large (F1-score: 0.74).

6.2. Towards Detection Models

While classification models perform well on diseases with distinct visual symptoms, they tend to struggle with categories such as Mosaic Virus and Yellow Leaf, where symptoms are often diffuse, subtle, or resemble environmental stress rather than clear pathological patterns. Through further experimentation, we observed that even the best-performing classification models are highly sensitive to background context, often misclassifying samples where disease cues are entangled with complex or misleading backgrounds. This vulnerability is illustrated in Figure 7, where Grad-CAM visualizations show that model attention is frequently diverted to non-disease regions. Moreover, classes with limited training data or high intra-class variability also exhibit noticeable performance drops. These limitations motivated our exploration of a detection-based approach, allowing models to focus on localized disease regions rather than relying on global image features, ultimately aiming for more robust and interpretable disease identification.

6.3. Object Detection Results

Table 3 presents a detailed comparison of state-of-the-art (SOTA) object detection models evaluated on the TomDet dataset, considering both localization (mAP, IoU) and classification (Recall, Precision,

F1-Score) metrics.

Table 3: Performance Comparison of Different SOTA Object Detection Models on Various Metrics in or TomDet Dataset.

Models	**Localization**		**Classification**		
	mAP	**IoU**	**Recall**	**Precision**	**F1-Score**
Non-YOLO Detection Models					
DETR	65.48	66.71	67.48	66.71	71.78
Faster R-CNN	69.96	73.71	71.26	73.71	92.47
Mask R-CNN	72.77	75.25	74.71	73.25	76.37
EfficientDet	41.81	34.47	53.93	83.35	55.49
RetinaNet	65.59	66.32	69.40	66.32	67.84
YOLO Based Models					
YOLO-v3	45.37	48.20	58.32	60.21	59.80
YOLO-v4	57.34	59.21	70.30	67.00	68.20
YOLO-v5	53.00	52.00	71.40	70.25	71.20
YOLO-v7	62.78	64.15	81.21	83.25	82.35
YOLO-v8	75.27	78.03	97.53	97.05	96.34
YOLO-v9	**79.44**	**81.00**	**98.53**	**98.05**	**97.86**

Among the non-YOLO models, Mask R-CNN demonstrates the strongest overall localization performance with an mAP of 72.77 and IoU of 75.25, followed by Faster R-CNN (mAP: 69.96, IoU: 73.71) and DETR (mAP: 65.48, IoU: 66.71). In terms of classification metrics, Faster R-CNN stands out with an F1-Score of 92.47, indicating a strong balance between precision (73.71) and recall (71.26). EfficientDet, designed for lightweight efficiency, underperforms across all metrics, with the lowest mAP (41.81) and IoU (34.47), suggesting that its optimizations for model size come at the cost of accuracy in this evaluation.

In contrast, the YOLO series shows a clear trend of progressive improvement in both localization and classification. YOLO-v3 and YOLO-v4 exhibit relatively modest performance, with mAP values of 45.37 and 57.34, respectively. YOLO-v5 improves classification metrics, achieving a F1-score of 71.20. YOLO-v7 further enhances performance, with an mAP of 62.78 and F1-Score of 82.35, surpassing most traditional

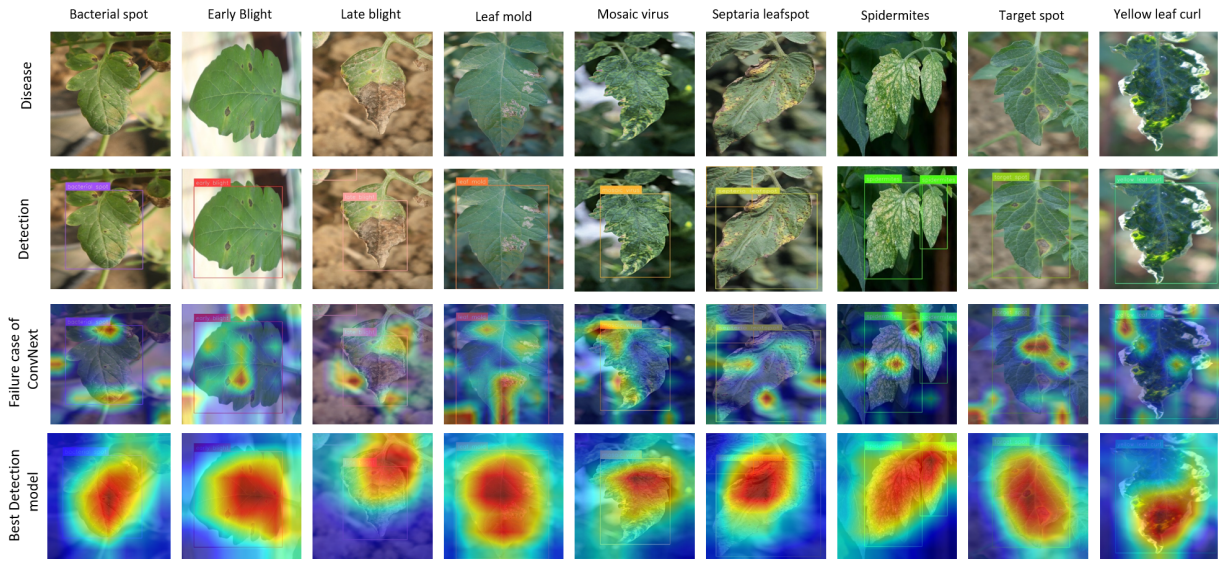


Figure 7: GradCAM Analysis between the Best Performing Detection Model (YOLO-v9) and Best Performing Classification Model (ConvNext). The GradCAM analysis shows that YOLO-v9 can predict the actual region of diseases, whereas ConvNext's activation map is randomly scattered.

detectors. YOLO-v8 marks a notable leap, achieving 75.27 mAP, 78.03 IoU, and a high F1-Score of 96.34, reflecting strong balance across all metrics. The latest model, YOLO-v9, achieves an mAP of 79.44, IoU of 81.00, Recall of 98.53, and an F1-Score of 97.86, representing an approximate 9% improvement in mAP and a 21% increase in F1-Score over YOLO-v7, thereby outperforming all preceding models across all evaluated metrics.

These results demonstrate that while conventional detectors remain competitive in certain aspects, modern YOLO variants (v8 and v9) consistently deliver superior performance across both localization and classification tasks, making them highly suitable for real-world applications demanding both speed and accuracy.

6.4. Grad-CAM Analysis

Grad-CAM [34] is a visualization technique that produces heatmaps to highlight the regions of an image most influential to a models prediction. To better understand the decision-making process, we performed a Grad-CAM analysis, comparing the best-performing classification model (ConvNext) and the top detection model (YOLO-v9) to evaluate their effectiveness in both identifying and localizing disease-relevant regions within the images.

The Grad-CAM visualizations in Fig. 7 highlight the contrasting behavior of YOLO-v9 and ConvNext in localizing disease-affected regions. YOLO-v9 demonstrates precise attention to the relevant areas of the tomato leaves, effectively identifying and localizing the diseased regions. In contrast, ConvNexts activation

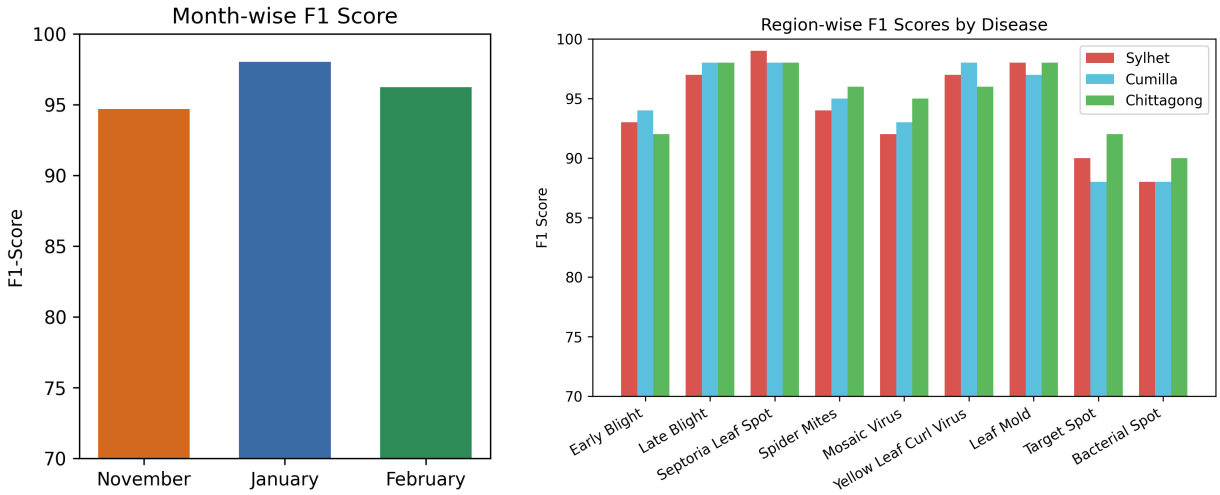
maps reveal a more dispersed focus, often attending to background regions that are not relevant to the classification task. For example, in the case of diseases such as Bacterial Spot, Late Blight, Septoria Leaf Spot, and Target Spot, the model achieves high accuracy for some of them, indicating its effectiveness in recognizing distinct visual patterns associated with certain diseases. However, the performance inconsistency across categories also points to vulnerable explanations, where the model may rely on spurious correlations or background features rather than the actual disease symptoms.

These findings from the Grad-CAM analysis offer valuable insights into each models decision-making process, showcasing YOLO-v9s strength in precise spatial localization while revealing ConvNexts limitations in attending to semantically meaningful regions. This observation motivated us to extend our experimentation beyond classification to include object detection, aiming for a more robust and localized understanding of disease-affected areas in tomato leaves.

6.5. Month & Region wise Analysis

To assess the generalization capability of the YOLO-v9 model, in Figure 8b we evaluated its performance across three regions in BangladeshSylhet, Cumilla, and Chittagongon nine tomato leaf diseases. The model achieved consistently high F1-scores across all regions, with Cumilla showing slightly better performance, particularly in detecting Late Blight, Mosaic Virus, and Septoria Leaf Spot (all at 98%). While minor regional variations were observedsuch as slightly lower scores for Bacterial Spot in Sylhet and Cumilla (88%), overall accuracy remained robust.

In Figure 8a, we also evaluated the month-wise F1 score for the YOLO model. Month-wise analysis revealed strong seasonal consistency, with average F1-scores of 94.68% in November, 98.02% in January,



(a) Month-wise Performance

(b) Region-wise Model Performance

Figure 8: Region and Month-wise Performance Analysis of YOLO-v9 Model

and 96.24% in February. The peak in January likely reflects optimal conditions for image clarity during the dry season. These results highlight YOLO-v9's reliability across different geographic and temporal conditions, supporting its potential for practical deployment in precision agriculture.

6.6. Efficient & Faster Inference on Edge Devices

In real-world agricultural settings, especially in remote or resource-constrained environments, relying on cloud-based solutions is often impractical due to limited internet connectivity and power constraints. Therefore, it becomes essential to deploy disease detection models directly on edge devices such as mobile phones, drones, or low-power embedded systems used by farmers.

While one common approach to achieving lightweight performance is to reduce the number of parameters in the models, we adopted a different strategy. Instead of compressing or pruning models, we explored the impact of using various backbone architectures in object detection frameworks. Given that YOLO models consistently outperformed other detection approaches in earlier evaluations, we focused our experimentation on YOLO variants, comparing different backbones to identify the most efficient configuration for disease detection in tomato leaves. This approach allows us to balance accuracy and computational efficiency, which is crucial for deploying models effectively on edge devices.

Table 4: Performance Analysis of YOLO Models with Different Backbones in the Tomato Leaf Dataset for Faster Inference in Edge Devices. Here cyan color indicates memory efficient faster inference model and gray color indicates the best performing model.

YOLO Models	**Backbone**	**Parameter**	**IoU ↑**	**μ-Pre ↑**	**μ-Recall ↑**	**μ-F1 ↑**
YOLO-v4	CSPDarknet-53	4.6M	59.21	70.30	67.00	68.20
	EfficientNet	3.8M	48.01	63.21	61.25	62.10
YOLO-v5	CSPDarknet-53	11.7M	52.00	71.40	70.25	71.20
	CSwin-Transformer	37.2M	47.05	65.21	64.21	63.35
	EfficientNet	17.83M	49.15	59.31	57.25	58.21
YOLO-v7	CSPDarknet-53	18.60M	64.15	81.21	83.25	82.35
	CSwin-Transformer	31.21M	56.12	78.01	76.65	77.40
	EfficientNet	14.88M	61.20	79.25	78.20	79.34
YOLO-v8	CSPDarknet-53	30.06M	78.03	97.53	97.05	96.34
	CSwin-Transformer	11.12M	75.15	92.33	94.30	93.20
	EfficientNet	10.67M	73.00	90.32	88.32	89.20
YOLO-v9	CSPDarknet-53	25.44M	81.00	98.53	98.05	97.86

Table 4 presents a comprehensive evaluation of YOLO models using different backbone architectures on the Tomato Leaf Dataset, focusing on performance trade-offs between accuracy and computational effi-

ciency for deployment on edge devices. Across the board, YOLO-v9 with CSPDarknet-53 emerges as the best-performing model, achieving the highest IoU (81.00), Precision (98.53), Recall (98.05), and F1-score (97.86), as highlighted in gray. This configuration demonstrates superior accuracy in both localization and classification but comes with a moderate parameter count (25.44M), making it ideal when accuracy is prioritized over lightweight deployment.

In contrast, YOLO-v8 with a CSwin-Transformer backbone, **YOLO-v8-CSwin**, highlighted in cyan, offers a notable balance between performance and efficiency, achieving strong detection results (IoU: 75.15, F1-score: 93.20) with only 11.12M parameters, making it well-suited for edge devices that require faster inference with limited resources. Interestingly, the EfficientNet backbones, while more memory-efficient in most cases, tend to show a drop in performance across all YOLO versionsespecially in earlier versions like YOLO-v4 and YOLO-v5. However, in YOLO-v7 and YOLO-v8, EfficientNet achieves respectable performance (e.g., YOLO-v8 with EfficientNet yields 89.20 F1-score at 10.67M parameters), showing its potential in balancing inference speed and reasonable accuracy.

Overall, this analysis underscores the value of evaluating various backbone strategies within YOLO models to meet specific deployment requirements. While YOLO-v9 is optimal for accuracy-focused applications, YOLO-v8 with CSwin-Transformer offers a more edge-friendly alternative with minimal compromise in detection performance.

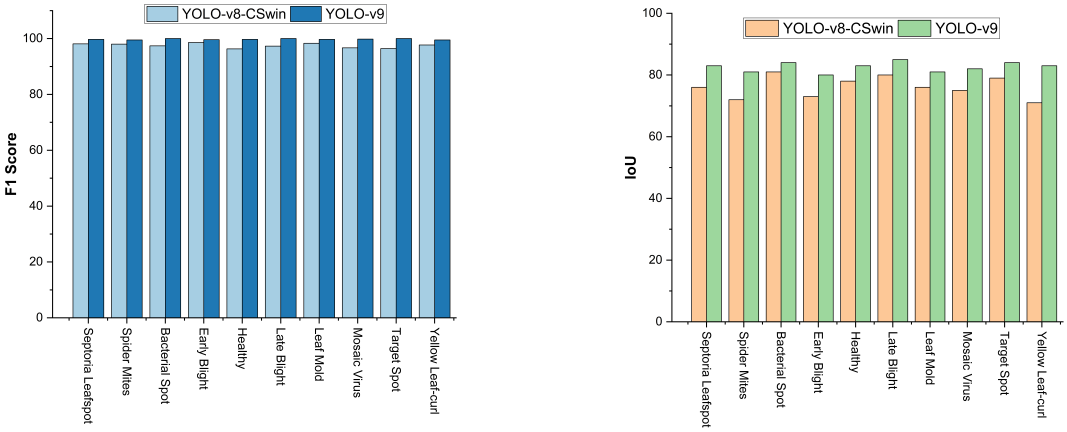
6.7. Comprehensive Comparative Analysis on YOLO-v8-CSwin and YOLO-v9

Given its strong balance between performance and efficiency, YOLO-v8-CSwin is particularly well-suited for deployment on edge devices. Therefore, we conducted a detailed comparison between YOLO-v8-CSwin and the overall best-performing model, YOLO-v9, to gain deeper insights into their respective strengths and limitations. This comparison helps evaluate how well a lightweight, edge-friendly model like YOLO-v8-CSwin performs relative to a high-capacity model in practical disease detection scenarios.

Figure 9a presents a class-wise F1-score comparison between YOLO-v8-CSwin and YOLO-v9, where the light blue bars represent YOLO-v8-CSwin and the dark blue bars correspond to YOLO-v9. The results show that both models perform consistently well across all ten disease categories, with YOLO-v9 demonstrating a marginal but steady advantage in classification accuracy. However, the performance gap is relatively small, underscoring the effectiveness of YOLO-v8-CSwin despite its lighter architecture.

Figure 9b compares the models in terms of Intersection over Union (IoU). As expected, YOLO-v9 outperforms YOLO-v8-CSwin across all classes, showing superior localization capabilitiesparticularly in difficult categories like Bacterial Spot, Early Blight, and Yellow Leaf Curl Virus.

Although YOLO-v9 is the top-performing model in both classification and localization, it comes with higher computational requirements. In contrast, YOLO-v8-CSwin achieves competitive performance with significantly fewer parameters, making it highly suitable for deployment on edge devices that require faster



(a) Class-wise F1-Score Comparison

(b) Class-wise IoU Comparison

Figure 9: Class-wise Performance Comparison between YOLO-v8-CSWin and YOLO-v9. For every class, YOLO-v8-CSwin performs almost similarly to YOLO-v9 but with a little margin behind.

inference and operate under resource constraints. Considering all aspects, YOLO-v8-CSwin offers an excellent trade-off between accuracy and efficiency, positioning it as a practical and scalable solution for real-time plant disease detection in the field.

7. Conclusion and Future Work

This paper advances code-mixed language translation through a comprehensive framework for Banglish-to-English translation. Our contributions introduce the BanglishTrans dataset with 14,000 meticulously curated parallel sentences, implement an effective dictionary-based word mapping preprocessing technique, develop the N-gram Overlap Maximization for Multi-Translation Selection (NOMS) algorithm, and create an integrated pipeline combining these components. Through extensive experimentation with state-of-the-art transformer-based models, our BanglaT5_nmt_bn_en model, integrated with the preprocessing approach and NOMS algorithm, achieves a BLEU score of 39.83, demonstrating substantial improvement over existing approaches and highlighting its potential impact on cross-cultural communication. This technical advancement directly addresses crucial challenges in digital information exchange by providing an effective solution for bridging communication gaps between Bengali and English-speaking communities, particularly in professional and academic contexts where precise translation is paramount. The system's ability to maintain high performance at a 70% sampling rate while significantly reducing computational overhead demonstrates its viability for real-world deployment, especially in resource-constrained environments.

Looking toward future developments, we have identified several promising directions for research and implementation. First, we plan to expand the BanglishTrans dataset to encompass a broader range of domains

and code-mixing patterns, enhancing the model's ability to handle diverse linguistic variations. Second, we aim to develop specialized modules for context-aware translation, particularly focusing on improving the system's capability to maintain semantic integrity across different communication contexts. Third, we will explore the integration of advanced verification mechanisms to enhance the framework's reliability in processing and translating sensitive information. These enhancements will focus on strengthening the system's role in facilitating accurate cross-lingual communication while maintaining contextual authenticity. Through these improvements, we anticipate expanding the practical applications of our method across various domains, contributing to more effective and reliable cross-cultural communication in our increasingly interconnected world.

Ethical Statement

This research adheres to ethical guidelines concerning data collection and model development. No personally identifiable information (PII) or sensitive content were included in the datasets, and we thoroughly reviewed them for potential biases. Our models are designed to minimize bias and promote fairness. All datasets and models will be made publicly available (where permissible) to ensure transparency and reproducibility. We use Generative AI (ChatGPT) to correct grammatical errors and improve the writing quality.

Acknowledgments

This work has been supported by the Independent University, Bangladesh (IUB), Dhaka, Bangladesh.

References

[1] M. Brahimi, K. Boukhalfa, A. Moussaoui, Deep learning for tomato diseases: classification and symptoms visualization, Applied Artificial Intelligence 31 (2017) 299–315.

[2] P. Chowdappa, B. Nirmal Kumar, S. Madhura, S. Mohan Kumar, K. Myers, W. Fry, D. Cooke, Severe outbreaks of late blight on potato and tomato in south india caused by recent changes in the phytophthora infestans population, Plant Pathology 64 (2015) 191–199. doi:10.1111/ppa.12231.

[3] G. Hu, X. Yang, Y. Zhang, M. Wan, Identification of tea leaf diseases by using an improved deep convolutional neural network, Sustainable Computing: Informatics and Systems 24 (2019) 100353.

[4] . Çuu, E. ener, . Erciyes, B. Balc, E. Akn, I. Önal, A. Akyüz, Treelogy: A novel tree classifier utilizing deep and hand-crafted representations, arXiv preprint arXiv:1701.08291 (2017).

[5] S. Mohanty, D. Hughes, M. Salathé, Using deep learning for image-based plant disease detection, Frontiers in Plant Science 7 (2016) 1419. doi:10.3389/fpls.2016.01419.

[6] Y. Huang, T. Wang, H. Basanta, Using fuzzy mask r-cnn model to automatically identify tomato ripeness, IEEE Access 8 (2020) 207672–207682.

[7] J. Liu, X. Wang, Tomato diseases and pests detection based on improved yolo v3 convolutional neural network, Frontiers in Plant Science 11 (2020) 898. doi:10.3389/fpls.2020.00898.

[8] J. Qi, X. Liu, K. Liu, F. Xu, H. Guo, X. Tian, M. Li, Z. Bao, Y. Li, An improved yolov5 model based on visual attention mechanism: Application to recognition of tomato virus disease, Computers and Electronics in Agriculture 194 (2022) 106780.

[9] M. Soeb, M. Jubayer, T. Tarin, M. Al Mamun, F. Ruhad, A. Parven, N. Mubarak, S. Karri, I. Meftaul, Tea leaf disease detection and identification based on yolov7 (yolo-t), Scientific Reports 13 (2023) 6078.

[10] G. Agrios, Plant Pathology, Elsevier, 2005.

[11] G. Vallad, G. Messelink, H. Smith, Crop protection: Pest and disease management, in: E. Euvelink (Ed.), Tomatoes, 2 ed., CAB International, Boston, MA, USA, 2018, pp. 207–257.

[12] M. Agarwal, A. Singh, S. Arjaria, A. Sinha, S. Gupta, Toled: Tomato leaf disease detection using convolution neural network, in: Procedia Computer Science, volume 167, 2020, pp. 293–301.

[13] K. Pohronezny, R. Volin, The effect of bacterial spot on yield and quality of fresh market tomatoes, HortScience 18 (1983) 69–70.

[14] C. Xie, Y. Shao, X. Li, Y. He, Detection of early blight and late blight diseases on tomato leaves using hyperspectral imaging, Scientific Reports 5 (2015) 16564.

[15] S. Rivas, C. Thomas, Molecular interactions between tomato and the leaf mold pathogen cladosporium fulvum, Annual Review of Phytopathology 43 (2005) 395–436.

[16] S. Parker, F. Nutter Jr, M. Gleason, Directional spread of septoria leaf spot in tomato rows, Plant Disease 81 (1997) 272–276.

[17] J. Abdulridha, Y. Ampatzidis, J. Qureshi, P. Roberts, Laboratory and uav-based identification and classification of tomato yellow leaf curl, bacterial spot, and target spot diseases in tomato utilizing hyperspectral imaging and machine learning, Remote Sensing 12 (2020) 2732.

[18] M. Kant, K. Ament, M. Sabelis, M. Haring, R. Schuurink, Differential timing of spider mite-induced direct and indirect defenses in tomato plants, Plant Physiology 135 (2004) 483–495.

[19] S. Matsuura, S. Ishikura, Suppression of tomato mosaic virus disease in tomato plants by deep ultraviolet irradiation using lightemitting diodes, Letters in Applied Microbiology 59 (2014) 457–463.

[20] J. Polston, R. McGovern, L. Brown, Introduction of tomato yellow leaf curl virus in florida and implications for the spread of this and other geminiviruses of tomato, Plant Disease 83 (1999) 984–988.

[21] F. Prinzi, M. Insalaco, A. Orlando, S. Gaglio, S. Vitabile, A yolo-based model for breast cancer detection in mammograms, Cognitive Computation 16 (2024) 107–120. doi:10.1007/s12559-023-10225-z.

[22] J. Fragoso, C. Silva, T. Paixão, A. Alvarez, O. Júnior, R. Florez, F. Palomino-Quispe, L. Savian, P. Trazzi, Coffee-leaf diseases and pests detection based on yolo models, Applied Sciences 15 (2025) 5040. doi:10.3390/app15095040.

[23] Y. Wang, P. Zhang, S. Tian, Tomato leaf disease detection based on attention mechanism and multi-scale feature fusion, Frontiers in Plant Science 15 (2024) 1382802. doi:10.3389/fpls.2024.1382802.

[24] J. L. Fleiss, Measuring nominal scale agreement among many raters, Psychological Bulletin 76 (1971) 378.

[25] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).

[26] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[27] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell, S. Xie, A convnet for the 2020s, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 11976–11986.

[28] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, An image is worth 16x16 words: Transformers for image recognition at scale, arXiv preprint arXiv:2010.11929 (2020).

[29] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, S. Zagoruyko, End-to-end object detection with transformers, in: European Conference on Computer Vision, Springer, 2020, pp. 213–229.

[30] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, Advances in Neural Information Processing Systems 28 (2015).

[31] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2961–2969.

[32] M. Tan, R. Pang, Q. Le, Efficientdet: Scalable and efficient object detection, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 10781–10790.

[33] T. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2980–2988.

[34] R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: Visual explanations from deep networks via gradient-based localization, in: Proceedings of the IEEE international conference on computer vision, ????