Jordan Journal of Mathematics and Statistics.
Yarmouk University

DOI:https://doi.org/10.47013/19.1.4

A Comparative Investigation of Disease Detection in Jamun (Syzygium cumini): A Study on the YOLOv3 and Gaussian YOLOv3Models

M. Haripriya[1], *A. Radhika*[2,*], *J. Jeslin*

Research Scholar, Department of Statistics, Periyar University, Salem, Tamil Nadu, India
*Assistant Professor, Department of Statistics, Periyar University, Salem, Tamil Nadu, India
Research Scholar, Department of Statistics, Periyar University, Salem, Tamil Nadu, India

Received: Aug. 8, 2024 Accepted: Dec. 7, 2025

Abstract: Leaf disease detection is a critical task in precision agriculture, aiming to monitor and control the spread of plant diseases for sustainable crop management. Object detection models have shown promise in accurately identifying and localizing diseases on plant leaves in recent years. This paper explores the effectiveness of YOLOv3 (You Only Look Once) and a variant known as Gaussian YOLOv3 in the context of leaf disease detection. YOLOv3 is known for its real-time object detection capabilities and high accuracy. However, it may face challenges in accurately localizing subtle disease patterns and handling uncertainties in complex leaf images. To address these challenges, Gaussian YOLOv3 incorporates Gaussian components to model uncertainty and improves localization accuracy. The comparative analysis involves evaluating the performance of YOLOv3 and Gaussian YOLOv3 in terms of localization accuracy, speed, adaptability to diverse conditions, and training requirements. Experiments are conducted using a dataset comprising various leaf diseases under different environmental conditions. They enable timely interventions and agricultural decision-making, reducing crop losses and ensuring effective disease management.

Keywords: Leaf disease detection, YOLOv3, multi-scale prediction, K-means cluster, Gaussian YOLOv3, Uncertainty.

2010 Mathematics Subject Classification. 26A25; 26A35.

1 Introduction

Jamun (Syzygium cumini), or Java plum or Indian blackberry, is a fruit-bearing tree native to the Indian subcontinent. While jamun trees are generally hardy, they can be susceptible to various diseases that affect the leaves. Common diseases impacting jamun leaves include fungal infections, bacterial diseases, and viral infections. These diseases manifest as spots, discoloration, wilting, or unusual leaf growth patterns. Monitoring and early detection of these diseases are crucial for effective disease management in jamun orchards. Its leaves are often used for various purposes, including culinary, medicinal, and cultural applications. Some studies suggest that jamun leaves help lower blood sugar levels by enhancing insulin sensitivity [1]. Indian jamun leaves are rich in antioxidants, notably polyphenols. These antioxidants play a role in neutralizing harmful free radicals in the body, potentially contributing to overall health and well-being. Hence, identifying and controlling diseases affecting jamun (Syzygium cumini) leaves are crucial for various reasons, encompassing agricultural and environmental considerations. Nevertheless, further research is required to establish these effects definitively. However, more research is needed to establish these effects conclusively. Understanding and detecting diseases in Jamun leaves are essential for ongoing research into disease resistance. Accurate disease detection is essential for identifying diseaseresistant varieties and formulating effective management strategies. With the advancement of technology, automated disease detection using machine vision can be seen as analogous to how computer vision systems recognize patterns and objects in images. Challenges arise with manual identification due to the laborious process of inspecting the entire farm, leading to compromised accuracy in predictions. Additionally, the process of identifying diseased leaves is prolonged. This paper describes a YOLOv3 and Gaussian

* Corresponding author e-mail: m_ *priyaprakash*11597*@gmail.comandaradhika@periyaruniversity.ac.in*

YOLOv3 model that can help identify Jamun tree disease. It's a task within computer vision where the goal is to recognize the presence of objects and determine their precise positions in the given space.
Using image processing techniques with diverse classification methods for identifying and diagnosing plant diseases is a valuable approach in agriculture [2]. Automated and precise disease detection in crops, facilitating timely intervention and mitigating yield losses. Techniques like Image Processing, Computer Vision, Machine Learning, and Deep Learning deal with complex and intricate real-world problems for which finding a solution by traditional methods is difficult [3]. In recent studies, the deep learning system outperformed the existing machine learning techniques [4]. Deep CNN's ability to automatically learn relevant features from uncontrolled image data has contributed to its success in various domains. Input to a CNN-based classifier is a raw image, and output is the probability that the image belongs to the considered classes. A disease detection system for cucumber leaves, employing a compact CNN, demonstrated an average accuracy of 82.3% through a 4 -fold cross-validation strategy[5]. Transfer learning has been used quite effectively to adapt deep convolutional neural networks to classify tomato leaves into six classes of diseased and healthy conditions [6] and to distinguish between healthy and unhealthy Cassava leaves [7]. Recently, a YOLO paper presented a groundbreaking approach to real-time object detection, simplifying the process into a single neural network, and its concepts continue to influence the development of subsequent object detection models [8] It has been addressing several safety-critical applications, agriculture, autonomous vehicles [9], unmanned aerial vehicles [10], real-world noise, and varied environments [10]. In that way, [11] this paper focuses on the specific application of detecting traffic signs. It proposes using image degradation models based on the YOLO network, augmented with traditional image processing methods.[12] The proposed smart farming technique for diagnosing and classifying external diseases within fruits aims to address the increasing demand for adequate growth and improved yield in the agricultural industry. The system employs image processing techniques, leveraging the Open CV library for implementation. [13] The paper addresses a pertinent issue in agriculture and proposes a unique solution using genetic algorithms for image segmentation. Effectively delineates the problem of plant diseases and emphasizes their potential repercussions on product quality, quantity, and overall productivity. The main innovations of this paper are summarized as follows: 1) To evaluate the impact of uncertainty modeling on the accuracy and reliability of bounding box predictions for leaf diseases. 2) To compare the performance of YOLOv3 and Gaussian YOLOv3 in the context of leaf disease detection.

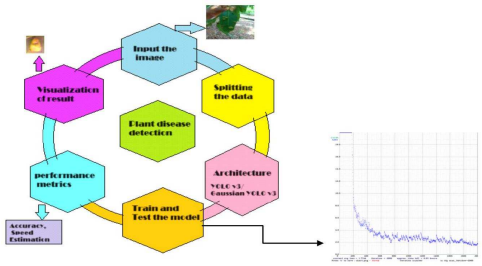

Fig. 1: Workflow Outline.

2 Methodology

This study compared YOLOv3 and Gaussian YOLOv3 models to determine the most suitable algorithm for detecting the diseased part of the jamun tree (Syzygium cumini) flow chart is represented in Fig. 1. YOLOv3 is versatile and ideal for a wide range of object detection tasks. Gaussian YOLOv3 is designed to estimate uncertainty in object detection. In the context of disease detection, uncertainty can arise from various factors such as variations in appearance, lighting conditions, and similar-looking symptoms. The model's ability to estimate uncertainty provides a measure of confidence in its predictions.

2.1 YOLOv3 Architecture

2.1.1 Training a custom YOLOv3 model

YOLOv3 utilizes DarkNet-53 as its underlying network architecture for object detection. This results in a fully convolutional architecture with 106 layers, as outlined by [14]. The architecture consists of blocks containing three

consecutive convolution layers with 3×3 and 1×1 filter sizes. These blocks are interconnected using residual connections, contributing to the network's architecture, and incorporate three distinct scales for predicting objects, aligning with the diverse feature maps generated by its architecture. Every scale is linked to a specific level of detail and accommodates objects of different sizes found in the input image. By adopting this multi-scale strategy, YOLOv3 effectively identifies objects across various scales and aspect ratios, improving its accuracy and resilience in object detection tasks. The last layer predicts a 3-D tensor encoding bounding box, objectness, and class predictions $\mathrm{S} \times \mathrm{S} \times (\mathrm{B} \times (4+1+\mathrm{C}))$ using logistic regression. Where, S is a grid size, B is the number of bounding boxes per grid cell and C is the number of object classes. YOLOv3 makes predictions for $\mathrm{B} = 3$ boxes at three distinct scales by extracting features from various scales, similar to the approach used in feature pyramid networks. Feature maps at these scales are derived from layers with strides of 32,16 and 8, respectively. For an input size of 416×416, detections are performed on scales of $13 \times 13, 26 \times 26$, and 52×52. The last two layers of these additional scales contribute features that are upsampled twice and combined with the base network's feature map. The upsampled features from the former provide semantic information, while those from the backbone offer finergrained details about the objects.

2.1.2 K-means Clustering

K-means clustering aims to identify the most representative bounding box shapes that cover the range of object sizes and aspect ratios present in the dataset. The number of clusters (k) equals the desired number of anchor boxes. During k-means clustering, the algorithm assigns each ground truth bounding box to the cluster (anchor box) that is the most similar in size and aspect ratio. The centroid of each cluster (average bounding box shape) becomes an anchor box. Subsequently, the data is classified into the nearest cluster, and the cluster center is adjusted iteratively until convergence. In k-means clustering, the dissimilarity or separation between data points and cluster centroids is measured by distance.

Distance=$\sqrt{(w-c_w)^2+(h-c_h)^2}$ Where, w and h are the width and height of a bounding box in the dataset, C_w and C_h are the width and height of a cluster centroid (representing an anchor box).

2.1.3 Predictions and Confidence Scores

Class prediction involves assigning probabilities to different classes to determine the most likely category of an object detected within an image. For each bounding box, the algorithm predicts four coordinates: x, y, w, h. where, x : The x -coordinate of the center of the bounding box (relative to the grid cell), y : The y -coordinate of the center of the bounding box (relative to the grid cell), w : The width of the bounding box (relative to the whole image or grid cell, depending on implementation) and h : The height of the bounding box (relative to the whole image or grid cell, depending on implementation). These coordinates are often predicted as offsets from the grid cell in which the bounding box is located. The predicted bounding box coordinates b_x, b_y, b_w, b_h can be obtained using the sigmoid function to squash the predicted values between 0 and 1 .

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h}
\end{aligned}
$$

Where, σ is the sigmoid function, t_x, t_y, t_w, t_h is the predicted values, c_x, c_y is the coordinates of the top-left corner of the grid cell. p_w, p_h are the dimensions of the anchor box.

The confidence score (Conf) measures the model's confidence that an object is in a bounding box. It is calculated using the sigmoid activation function on the raw network output.

$$\mathrm{Conf} = \sigma(t_{\mathrm{Conf}}) = P_r(\text{ object }) \times \text{ IoU (pred , truth)}$$

Where, t_{Conf} is the raw network output for confidence. The final confidence score is obtained by multiplying the objectness score (Conf) with the class-specific score (Class). The confidence score measures how certain the model is about object presence in the predicted box.

$$
\begin{aligned}
\mathrm{NMS}(B, \mathrm{Conf}_{\mathrm{threshold}}, \mathrm{IoU}_{\mathrm{threshold}}) = \{ & b_i \in B \mid \mathrm{conf}(b_i) > \mathrm{Conf}_{\mathrm{threshold}} \\
& \wedge \forall b_j \in B, i \neq j, \mathrm{IoU}(b_i, b_j) \leq \mathrm{IoU}_{\mathrm{threshold}} \}
\end{aligned}
$$

Where, B is the bounding box prediction, b_i is the i^{th} predicted bounding box, b_j is the j^{th} predicted bounding box, conf(b_i) is the confidence score of the bounding box b_i, $IoU\,(b_i, b_j)$ is the Intersection over Union between bounding boxes b_i and, b_j, Conf_threshold is a confidence threshold that determines which predictions to keep, IoU_thresholdis an IoU threshold that determines how much overlap is tolerated before discarding a bounding box. Non-maximum suppression (NMS) helps eliminate duplicate and overlapping predictions, ensuring that each object is detected only once and assigned to the most probable class.

$$C_i^j = \text{Conf} * \text{Class, class} \in \{0,1\}$$
$$IoU = \frac{\text{Areaof overlap}}{\text{Areaof union}} = \frac{b_i \cap b_j}{b_i \cup b_j} = \frac{I}{U}$$

Where, $b_j = (x, y, w, h)$ represent the position of ground –truth and $b_i = (t_x, t_y, t_w, t_h)$ represent the position of prediction. Therefore, the IoU loss function is suggested to be adapted for the IoU metric. The IoU loss is typically computed
as $L_{\text{IoU}} = 1 - IoU$. The goal during training is to minimize the IoU loss, encouraging the model to generate predictions with high IoU values, indicating better alignment with the ground truth. Ground truth bounding boxes, class labels, and object presence information are provided during training. The model is trained using a loss function that combines Localization Loss (Coordination Loss, (LL)), Confidence Loss(CL), and Class Probability Loss(CPL).

$$LL = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(t_{x_i} - x_i)^2 + (t_{y_i} - y_i)^2 \right] + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(\sqrt{t_{w_i}} - \sqrt{w_i} \right)^2 + \left(\sqrt{t_{h_i}} - \sqrt{h_i} \right)^2 \right]$$
$$CL = \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(\text{Con}\, f_i - \hat{Conf}_i \right)^2 + \lambda_{\text{noobj}}\, 1_{ij}^{\text{noobj}} \left(\text{Conf}\, f_i - \hat{Conff}_i \right)^2 \right]$$
$$CPL = \sum_{i=0}^{S^2} 1_i^{obj} \sum_{c=0}^{C} (\ \text{Class}_{\ i,c} - \ \text{Class}_{\ i,c})^2$$

TotalLoss = Localization Loss + ConfidenceLoss + ClassProbabilityLoss The goal is to minimize the total loss across the dataset to improve detection accuracy. It quantifies the statistical differences between model predictions and actual data, guiding the model's learning process.

2.2 Gaussian Yolo V3

Gaussian YOLOv3 shares the same underlying architecture as YOLOv3 for object detection, but the key difference lies in how they model the uncertainty associated with bounding box predictions. The algorithm aims to improve the accuracy and robustness of object detection, especially in cases where the objects have irregular shapes or significant variations. Instead of encoding bounding box coordinates directly, Gaussian YOLOv3 utilizes Gaussian distributions to represent the uncertainty linked to localization. Each bounding box coordinate (x, y, w, h) is modeled as a Gaussian distribution with mean (μ) and standard deviation (σ). The parameters of the Gaussian distribution are predicted by the network for each bounding box. These parameters encompass the mean (μ) and standard deviation (σ) for the x, y, width (w), and height (h) coordinates. This Gaussian distribution can be described by its mean μ_{t_x} and variance Σ_{t_x}. So, the output (μ_{t_x}, Σ_{tx}) is treated as a complete Gaussian curve. More naturally, we can also treat the mean value μ_{t_x} as the prediction t_x and the variance Σ_{t_x} as its uncertainty. The variance Σ_{t_x} can be interpreted as uncertainty because the value range t_x is between 0 and 1 . In a practical context, one can perceive the uncertainty as a learned weighting parameter that impacts the loss and functions as a regularizer. The loss is based on the negative log-likelihood of Gaussian distributions for bounding box coordinates, and the confidence loss is derived from the negative log-likelihood of a Gaussian distribution for the confidence score.

$$L_{conf} = \sum_{i,j,k} \gamma_{ijk} \left[\frac{1}{2} \log \left| 2\pi \left(\sum_{ijk} \right) \right| + \frac{1}{2} \left(x_{ijk} - \mu_{ijk} \right)^T \left(\sum_{ijk} \right)^{-1} \left(x_{ijk} - \mu_{ijk} \right) \right]$$

Where x_{ijk} is the ground truth confidence score for the box, and μ_{ijk} and $\sum_{ijk}$ are the predicted mean and predicted covariance matrix for that box. γ_{ijk} indicator function (1 if object is present in cell (i,j,k), 0 otherwise), $|\Sigma|$ is a determinant of the covariance matrix and Σ^{-1} is a inverse of the covariance matrix.

Architecture	Training Samples	Testing Samples	Total Samples	Average loss
YOLOv3	1600	400	2000	1.7743
Gaussian YOLOv3	1600	400	2000	1.5189

Table 1: Number of images for training and testing

This formulation penalizes both inaccurate predictions and excessive uncertainty. The Mahalanobis term penalizes deviation from the ground truth, while the log-determinant term discourages unrealistically large covariance (uncertainty). The weighting factor $\gamma_{(ijk)}$ scales the loss but is not placed inside the logarithm, since $\log(\gamma N() + .\varepsilon) \neq \gamma \log(N()$. To ensure numerical stability during training, especially when computing Σ^{-1}, a small constant ε is added to the diagonal of the covariance matrix: $\Sigma_{ijk} \leftarrow \Sigma_{ijk} + \varepsilon \mathbf{I}$ This regularization ensures that Σ remains positive definite and invertible. The addition of ε is purely a computational safeguard applied during implementation and does not modify the theoretical model. The model predicts both μ and σ for each coordinate. The loss is backpropagated using the negative log-likelihood, which naturally weights predictions by their uncertainty. The confidence score is derived from the likelihood of the predicted bounding box
matching the ground truth, incorporating both location and scale uncertainty. The total training objective for Gaussian YOLOv3 combines localization, confidence, and classification losses:

$$L_{\text{Total}} = L_{\text{loc}} + L_{\text{conf}} + L_{\text{class}}$$

This approach allows Gaussian YOLOv3 to produce more reliable bounding boxes, especially in cases of ambiguous or noisy input data, such as leaf images with subtle disease symptoms.

3 Experimental Analysis

3.1 Data set collection and labeling

To train and test our model, we collected a dataset of images of various trees from various places and their corresponding diseases Table 1. Our team collected the dataset in real-life scenarios under plant pathologists' supervision, ensuring it reflected the diverse range of conditions Fig. 2. The dataset consists of images of leaves affected by various diseases, including the White fly (Dialeurodes eugenia), the leaf-eating caterpillar (Carea subtillis), and the fungal anthracnose.
The images were captured using high-quality cameras with different angles and lighting conditions to make the dataset

Fig. 2: (a) Non diseased (b) Diseased.

diverse and challenging. To label the image, we used the labeling tool. To install 'labelImg,' we used 'pip install labelImg' in Python. Once the installation was completed successfully, we labeled the images by drawing bounding boxes and assigning class labels. After successfully labeling the image, we get the output as a text file. The text file consists of 5 decimal values; the first value represents the bounding box class, next is center x , then center y, followed by the width and the height [15].

	YOLOv3	Gaussian YOLOv3
mAP@ IoU	49.08%	50.62%
Precision	68%	73%
Recall	4%	61%
Training Time (Min)	69.38563 min	98.9456 min
Batch	1	1
Inference time	99s	2682.740000 milli-seconds
F1	0.5	0.66

Table 2: Comparison results of YOLOv3 and Gaussian YOLOv3

3.2 Performance evaluation metrics

3.2.1 The method for calculating precision

Typically, network performance is assessed using precision and recall as evaluation metrics. Precision measures the accuracy of the optimistic predictions made by the model. Recall measures the ability of the model to capture all the relevant instances of the positive class.

$$\text{Precision } = \frac{TP}{TP+FP}, \quad \text{Re call } = \frac{TP}{TP+FN}, \quad F1 = 2 * \frac{\text{Pr ecision } * \text{ Re call}}{\text{Pr ecision } + \text{ Re call}}$$

Where TP represents the true positive, FP represents the false positive, and FN represents the false negative. The F1 score, the harmonic mean of precision and recall, is another metric that combines these two measures into a single value.

3.2.2 Mean Average Precision (mAP)

mAP is frequently employed in object detection tasks to assess the precision of bounding box predictions. It considers precision at different recall levels, and the average precision is computed across multiple categories of diseases. $mAP = \frac{1}{N}\sum_{i=1}^{N} AP$

Where, AP is a Average Precision, Recall Levels is the Different thresholds (e.g., 0.1 to 1.0) used to measure precision at varying recall levels and N is the total number of predictions evaluated.

4 Result

4.1 Accuracy and Inference time

The model is performed on a GPU (Tesla T4) with CUDA version 11.080 and cuDNN version 8.9.0. \Open CV version is 4.5.4. The YOLOv3 starts with a convolutional layer (conv) with 32 filters and a 3×3 kernel, processing a $416 \times 416 \times 3$ image. It follows with several convolutional layers, down sampling the image and increasing the filters. Shortcut layers (Skip connections) are used to jump over some layers and help with the flow of information. The network has three YOLO detection layers (Yolo) at 82, 94, and 106 scales. Layer 82 makes detections on a 13×13 feature map. Layer 94 makes detections on a 26×26 feature map. Layer 106 makes detections on a 52×52 feature map. The Gaussian YOLOv3 network starts with an input image size of 512×512 pixels and three color channels (RGB). Upsampling layers (like "upsample $2x$ ") are used to increase the spatial resolution of feature maps. The network has three detection layers like YOLOv3 but of type 29. These layers predict bounding boxes and class probabilities for detected objects. The detection result is shown in Table 2. The mean average precision of YOLOv3 is reported as 49.08%. Similarly, mAP of Gaussian YOLOv3 is reported as 51.72%. Other metrics, such as precision, recall, and F1-score, are also provided for a confidence threshold 0.25 . A total of 4489 detections are made. There are 858 ground truth instances in the dataset. The detection statistics for the class include True Positives (TP = 339) and False Positives (FP = 161). The average Intersection over Union (IoU) is 47.49%. Similarly, the detection statistics for Gaussian YOLOv3 include True Positives (TP = 369) and False Positives (FP = 131). The average Intersection over Union (IoU) is 49.74%. In the context of Total BFLOPS (Billion Floating Point Operations Per Second), 65.304 is relatively low. BFLOPS measures the computational complexity of a model, indicating how many billion floating-point operations the model can perform in one second. YOLOv3 BFLOPS value of 65.304 might be considered moderate or efficient,

especially if the model is designed to be lightweight and suitable for real-time applications or deployment on resource-constrained devices. The Gaussian YOLOv3 model performs approximately 98.798 billion floating-point operations per second during inference. Higher BFLOPS values are often associated with faster inference speeds, which achieve 2682.740000 milliseconds. Also, the model can process input data more quickly. This is important for real-time applications where low latency is crucial. Since more efficient models can handle larger amounts of data or more complex computations within a given time frame, Gaussian YOLOv3 achieves that effect. Fig. 3 and Fig. 4 are the detection result of the YOLOv3 algorithm and Gaussian YOLOv3 algorithm. From the detection results, Gaussian YOLOv3 has an excellent adaptation to the complex illumination variation, and it can detect the object much more accurately than the YOLOv3 in real-time detection.

Fig. 3: YOLOv3 predictions

Fig. 4: Gaussian YOLOv3 predictions

After training and testing the dataset, the experiment results show that the mAP of object detection of Gaussian YOLOv3 is 95.56%. It's 34.09% better than the YOLOv3. Visualization of bounding boxes with uncertainty allows a better understanding of the model's confidence in the predicted disease location. Instead of presenting a fixed bounding box, the uncertainty can be represented as a probability distribution, showing the range where the disease might be present Fig. 5. Therefore, the lower uncertainty allows for a broader acceptance of predictions.

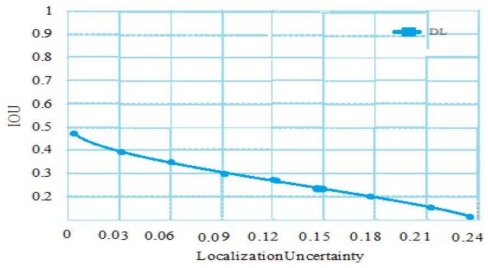

Fig. 5: IOU Vs LU

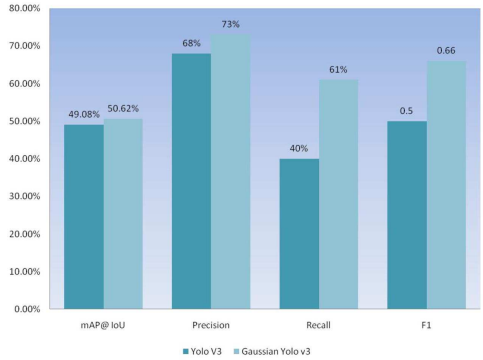

Fig. 6: Comparison chart of detection result of jamun leaf (mAP, Precision, Recall, F1

4.2 Limitations of the modeling approach

Gaussian YOLOv3 requires tuning hyper parameters specific to Gaussian distributions, which may increase computational resource requirements for training and inference compared to traditional YOLOv3. Optimizing the hyper parameters and exploring model compression techniques could alleviate this limitation. The experiment results demonstrate the effectiveness of Gaussian YOLOv3 for leaf disease detection in jamun trees. However, the model's scalability to other tree species or crops needs further investigation. Different crops may exhibit unique disease patterns and environmental conditions affecting the model's performance. The training time for Gaussian YOLOv3 is relatively longer compared to traditional YOLOv3. This could be a limitation in scenarios where computational resources are limited or real-time detection is required. Investigating methods to optimize training time without compromising performance would be beneficial.

5 Conclusion

Based on the experimental analysis, it is clear that both YOLOv3 and Gaussian YOLOv3 have proven to be effective in detecting diseases on jamun tree leaves. However, Gaussian YOLOv3 outperforms traditional YOLOv3 in several ways. Firstly, Gaussian YOLOv3's ability to model uncertainty gives it an advantage in accurately pinpointing subtle disease patterns and handling uncertainties in complex leaf images. This capability allows for more precise predictions of the bounding box, particularly when dealing with objects that have irregular shapes or significant variations. The incorporation of Gaussian components improves the accuracy of localization and provides a level of confidence in the model's predictions. Additionally, Gaussian YOLOv3 achieves a higher mean average precision (mAP) compared to YOLOv3, indicating its superior ability to detect diseased areas on jamun tree leaves. This increased precision leads to better precision, recall, and F1-score performance, demonstrating the effectiveness of Gaussian YOLOv3 in accurately identifying and localizing leaf diseases. In addition, despite the slightly extended duration of training for Gaussian YOLOv3, the model maintains a reasonable inference time, making it well-suited for real-time applications. The capacity to swiftly handle input data is essential for timely interventions and effective agricultural decision-making, leading to reduced crop losses and improved disease management.

In conclusion, Gaussian YOLOv3 emerges as a promising solution for leaf disease detection in jamun trees, offering improved accuracy, robustness, and adaptability to diverse conditions compared to traditional YOLOv3. By leveraging uncertainty modeling, Gaussian YOLOv3 enhances the reliability of bounding box predictions, ultimately facilitating more effective disease management in precision agriculture.

Future work: Extending the research to include other crop species and types of diseases can broaden the applicability of the developed models. Different crops may present unique challenges and characteristics that require specific adaptations of the detection algorithms.

6 Declarations

Competing interests: The author declare no competing of interest
Authors' contributions: M. Haripriya A. Radhika J. Jeslin conceived of the presented idea. M. Haripriya developed the theory and performed the computations. A. Radhika verified the analytical methods. All the authors discussed the results and contributed to the final manuscript.
Funding: No Funding
Availability of data and materials: The data are available from the corresponding author, upon reasonable request.
Acknowledgments: The authors sincerely thank the referees for their valuable suggestions.

References

[1] M. K. Rizvi, R. Rabail, S. Munir, M. Inam-Ur-Raheem, M. M. N. Qayyum, M. Kieliszek, A. Hassoun, R. M. Aadi, Astounding Health Benefits of Jamun (Syzygium cumini) toward Metabolic Syndrome, *M*olecules, **27**(2022), 7184.

[2] V. Singh, N. Sharma and S. Singh, A review of imaging techniques for plant disease detection,*A*rtificial Intelligence in Agriculture, **4** (2020), 229-42.

[3] K. N. Das, J. C. Bansal, K. Deep, A. K. Nagar, P. Pathipooranam and R. C. Naidu , *S*oft Computing for Problem Solving: SocProS 2018, **2**(2020), 1057.

[4] S. Khan, M. Tufail, M. Khan, Z. Ahmad and S. Anwar, Deep learning-based identification system of weeds and crops in strawberry and pea fields for a precision agriculture sprayer,*P*recision Agriculture, **22**(2021), 1-17.

[5] E. Fujita, Y. Kawasaki, H. Uga, S. Kagiwada and H. Iyatomi, Basic Investigation on a Robust and Practical Plant Diagnostic System 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA) (Anaheim, CA, USA: IEEE), 2016, pp 989-92.

[6] P. Bhatt, S. Sarangi and S. Pappula, Comparison of CNN models for application in crop health assessment with participatory sensing 2017 IEEE Global Humanitarian Technology Conference (GHTC) 2017 IEEE Global Humanitarian Technology Conference (GHTC) (San Jose, CA: IEEE), 2017, pp 1-7.

[7] A. Ramcharan, K. Baranowski, P. McCloskey, B. Ahmed, J. Legg and D. P. Hughes, Deep Learning for Image-Based Cassava Disease Detection*F*ront. Plant Sci. **8**(2017), 1852.

[8] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection,*a*rXiv 2016.

[9] Z. Sultan, M. U. Farooq and R. H. Raza, Improved Vehicle Logo Detection and Recognition for Complex Traffic Environments Using Deep Learning Based Unwarping of Extracted Logo Regions in Varying Angles,*D*igital Interaction and Machine Intelligence, 2023, pp 12-25

[10] K. C. Saranya, A. Thangavelu, A. Chidambaram, S. Arumugam and S. Govindraj, Cyclist Detection Using Tiny YOLO v2,*S*oft Computing for Problem Solving, 2020, pp 969-79.

[11] C. Liu, Y. Tao, J. Liang, K. Li and Y. Chen, Object Detection Based on YOLO Network, *I*EEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC, 2018, pp 799-803.

[12] A. Awate, D. Deshmankar, G. Amrutkar, U. Bagul and S. Sonavane, Fruit disease detection using color, texture analysis and ANN, *I*nternational Conference on Green Computing and Internet of Things (ICGCIoT) (Greater Noida, Delhi, India: IEEE), 2015, pp 970-5

[13] V. Singh and A. K. Misra, Detection of plant leaf diseases using image segmentation and soft computing techniques *I*nformation Processing in Agriculture **4**(2017), pp 41-9.

[14] P. V. Bhatt, S. Sarangi and S. Pappula, Detection of diseases and pests on images captured in uncontrolled conditions from tea plantations, *A*utonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping, vol 11008 (SPIE), 2019, pp 73-82

[15] M. Mathew and T. Y. Mahesh, Leaf-based disease detection in bell pepper plant using YOLO v5,*S*ignal, Image and Video Processing **16**(2022).