

Design and Evaluation of a Deep Convolutional Neural Network for Semantic Segmentation of Tomato Leaf Diseases

A. Hemalatha¹, J. Vijayakumar², M. Mervin Paul Raj³

^{1,2,3} Department of Electronics and Instrumentation, Bharathiar University, Coimbatore, India

Abstract

This research introduces a unique method for detecting and segmenting diseases in tomato leaves, utilizing sophisticated computer vision and deep learning techniques. The study employs a deep convolutional neural network (DCNN) for image segmentation, particularly targeting tomato leaf diseases. The network architecture comprises an encoder (ResNet) and a decoder (DeepLabv3), with specific elements such as ResBlocks and Atrous Spatial Pyramid Pooling (ASPP). This approach yields promising results in achieving precise pixel-wise classification of tomato leaves, a critical aspect of disease detection and management. The proposed network's performance is compared with other segmentation algorithms, specifically U-Net and Segnet, and assessed using various metrics, including Accuracy, Intersection over Union (IoU), and the Jaccard and Dice similarity coefficients.

Keywords: Deep Convolutional Neural Networks, Semantic Segmentation, Tomato Leaf Diseases, Image Analysis, Agricultural Machine Vision

1. Introduction

Tomato leaf diseases significantly impact agriculture, affecting both plant health and crop yield. Accurate detection and segmentation of these diseases are vital for effective disease management. In recent years, advanced computer vision and deep learning techniques have been increasingly used to tackle this issue. Researchers have emphasized the importance of precise disease classification in tomato cultivation through effective image-based segmentation methods [1]. Early detection is particularly important as diseases like Septoria leaf spot can lead to considerable yield losses, especially during warm and wet seasons [2].

Deep learning approaches have demonstrated potential in efficiently detecting tomato plant diseases [3]. Moreover, discussions about the potential of CNN-based models for identifying and categorizing tomato leaf diseases [4] and the importance of accurate detection and classification for effective disease management [5] have influenced the field.

Semantic segmentation, a critical image analysis task, assigns class labels to each pixel, facilitating a detailed understanding of image content. Segmenting tomato leaves into healthy and diseased regions is crucial for disease control [6]. Precise segmentation allows for targeted treatment and control measures [1]. Semantic segmentation overcomes limitations in disease detection by providing pixel-level classification, aiding in assessing disease severity based on the affected leaf area [7]. Models have been enhanced with attention mechanisms and dilated convolution feature extractors [8, 9]. Integrating semantic segmentation with architectures like MC-UNet and EfficientNet has improved disease detection accuracy. The significance of semantic segmentation in plant disease detection has been emphasized in various studies [6, 10].

2. Related works

Numerous studies have delved into the segmentation of tomato leaf diseases using various techniques, including clustering, deep learning, feature extraction, and segmentation algorithms. A key aspect of tomato leaf disease detection based on computer vision is the accurate segmentation of healthy and diseased areas of the tomato leaf [11]. The Tomato Leaf Disease Detection (ToLeD) model, which is based on convolutional neural networks (CNNs), has achieved high accuracy rates of 91.2% in classifying various tomato diseases from leaf images [10]. For disease detection, texture-based feature extraction using Gabor filters has been applied through Otsu thresholding segmentation [12].

Deep learning models, especially CNNs, have been widely explored for segmenting and classifying tomato leaf diseases, surpassing traditional methods [13]. Comparative studies have shown the superiority of CNNs over support vector machines and k-nearest neighbor methods [14]. Innovative algorithms, such as Leaf Segmentation Fuzzy CNN (LSFCNN) and ant colony-based Mask RCNN, have been developed for detecting tomato leaf diseases in complex backgrounds [15]. Hybrid deep learning approaches, which combine attention-based dilated convolution feature extractors with logistic regression, have also been used for accurate disease detection [8]. Various CNN architectures have enhanced tomato leaf disease classification and segmentation, including VGG Nets, Lenets, ResNet, R-CNN, Mask R-CNN, FCN, and SSD [16]. Furthermore, restructured deep residual dense networks have been used to identify specific tomato leaf diseases, such as spot blight, late blight, and yellow leaf curl [17].

Semantic segmentation, a crucial task in computer vision, has evolved through deep learning models like DeepLab v3+ and ResNet. DeepLab v3+ is commonly used for semantic segmentation tasks, and its effectiveness is enhanced by leveraging pre-trained models like ResNet-18 and ResNet-50 [18-20]. ResNet, known for its deep architecture with residual connections, has also played a significant role in semantic segmentation tasks, combining with DeepLab v3+ for various applications [20-22]. The literature emphasizes the exploration of different architectures and techniques to enhance semantic segmentation. Novel approaches like SegNet, RCC-Net, and PP-NAS have been proposed to address challenges in semantic segmentation tasks [23-25].

3. Methodology

3.1. Dataset Preparation and Pre-processing

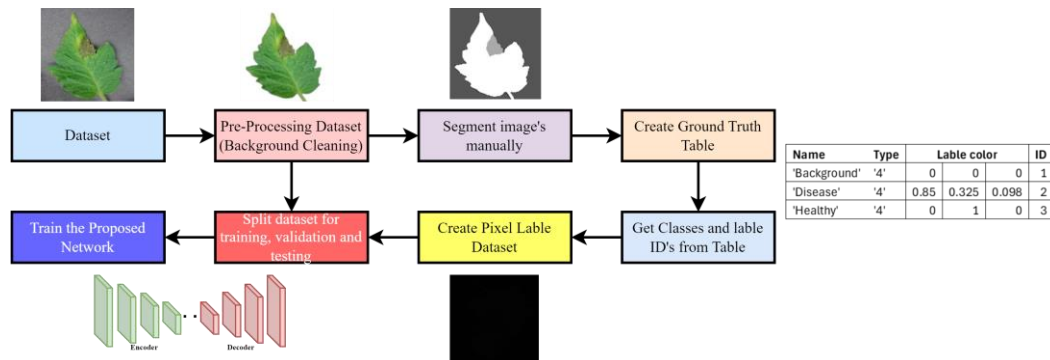


Figure 1. Pre-processing the Dataset for Training the Network

This study assembled an initial dataset of images depicting tomato leaf diseases from the Plant Village dataset. After pre-processing to eliminate background noise, we manually segmented the images to create accurate pixel labels. This step involves

identifying regions of interest (ROIs) and labeling each pixel within those ROIs. Each pixel in an image should be assigned a class label (e.g., foreground, background, or specific object classes). In our case, we labeled it as Background, Diseased, and Healthy. Using this labeled pixel dataset and pre-processed dataset, we divided it into training, validation, and testing subsets. Subsequently, we trained the proposed network for image segmentation. The pre-processing steps are illustrated in Figure 1.

3.2. Semantic Segmentation Network Architecture

Semantic segmentation is a critical task in computer vision, aiming to assign class labels to each pixel in an image. This methodology outlines the design and evaluation of a semantic segmentation network. The objective is to achieve accurate pixel-wise classification for tomato leaves. The network comprises two main parts: an encoder and a decoder. Here, ResNet serves as the backbone of the encoder with a DeepLabv3 decoder. The basic architecture is depicted in Figure 2.

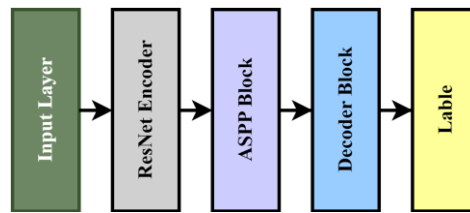


Figure 2. Block diagram of Semantic Segmentation Network Architecture

3.3. Encoder (ResNet)

The encoder extracts feature from the input image. The code uses pre-trained layers from a ResNet architecture. Here, specific layers like convolutional layers, batch normalization layers, and ReLU activation layers are used for feature extraction.

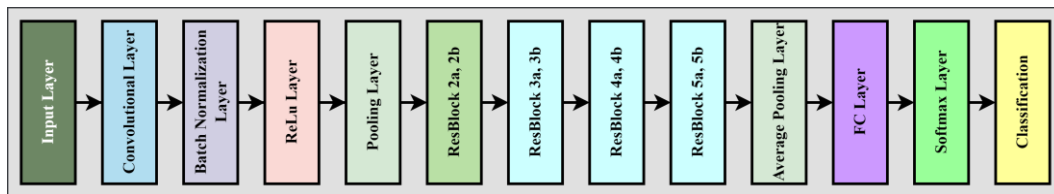


Figure 3. Resnet Architecture

The block diagram in Figure 3 represents a deep-learning model used for image recognition and classification. The process begins with the input layer receiving input data as 256×256 . This is followed by a convolutional layer applying filters to create a feature map. A batch normalization layer then normalizes this output, which is then passed through a Rectified Linear Unit (ReLU) layer to introduce non-linearity. The pooling layer reduces the spatial dimensions while preserving important information. Residual blocks (ResBlocks) containing multiple convolutional layers and shortcut connections follow, helping the model learn complex features and alleviate the problem of vanishing gradients in deep networks. An average pooling layer further reduces the spatial dimensions, preparing the data for the final classification stage. A fully connected (FC) layer then flattens the output. It connects every neuron to every neuron in the next layer, allowing the model to learn global patterns. A softmax layer outputs a probability distribution over the classes, indicating the model's confidence for each class. Finally, the classification layer provides the model's output, which is the class prediction for the input image.

The layers up to the average pooling layer will act as the encoder part for the segmentation network as they allow the capture of the hierarchical feature representations of the input image. The encoder will progressively reduce the spatial dimensions of the

input while increasing the depth, capturing high-level semantic information. The output of the average pooling layer will then serve as the input to the decoder part of the network, which will upsample the feature maps back to the original input size, producing a pixel-wise classification.

3.3.1. ResBlocks: The ResBlocks 2a 2b, shown in Figure 4, consist of two primary blocks: Block A and Block B. Each block represents a series of layers that data passes through in the network. Block A includes convolutional, batch normalization, and ReLU activation layers. The output from the ReLU layer in Block A is then added to the input of the next block, Block B, which has a similar structure to Block A. Adding the input from the previous layer is a form of skip connection or shortcut, which helps mitigate the problem of vanishing gradients in deep neural networks.

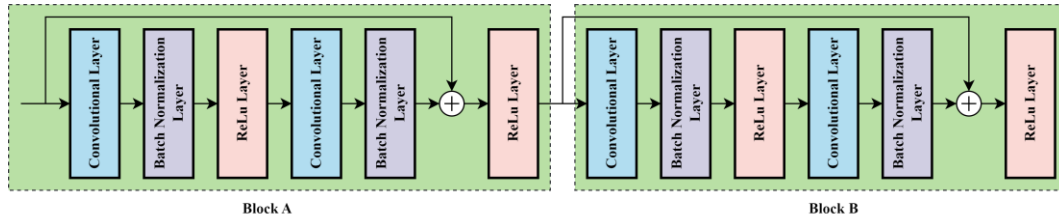


Figure 4. ResBlocks 2a 2b

The ResBlocks from 3a, 3b to 5a, 5b, shown in Figure 5, are similar to ResBlocks 2a, 2b. The only change is that it has another convolutional layer, and the batch normalization layer follows. The output from this second batch normalization layer is added to the output from the first ReLU layer. This addition operation forms a shortcut or skip connection. The result then passes through another ReLU Layer.

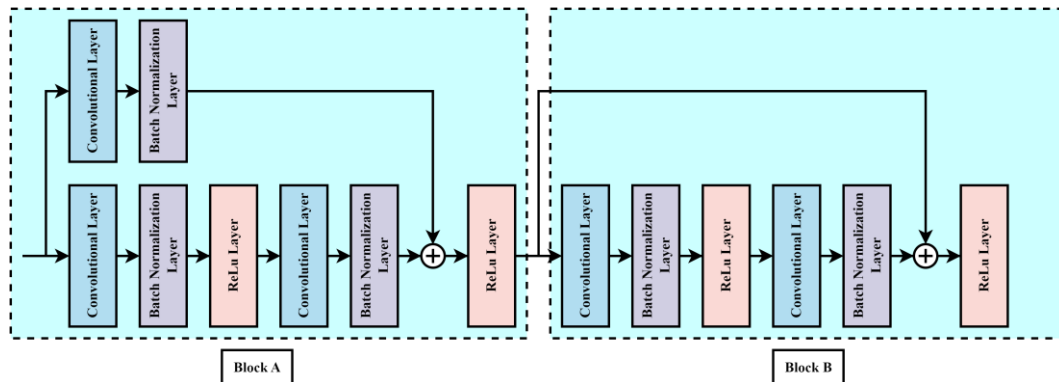


Figure 5. ResBlocks from 3a, 3b to 5a, 5b

3.4. Decoder (DeepLabv3)

The decoder part refines the extracted features and predicts the pixel-wise class probabilities. It utilizes techniques such as Atrous Convolution, which applies convolution with dilated kernels to capture long-range dependencies in the features. ASPP (Atrous Spatial Pyramid Pooling) aggregates features from different scales using atrous convolutions with varying dilation rates. Lastly, Upsampling increases the resolution of the feature maps to match the input image size.

3.4.1. Atrous Spatial Pyramid Pooling (ASPP) and Decoder: Figure 6 represents a neural network architecture specifically designed for tasks that require capturing multi-scale context, such as image segmentation. This architecture utilizes Atrous Spatial Pyramid Pooling (ASPP), a technique that has been proven effective in capturing multi-scale context. The architecture begins with a ResBlock 5a, 5b layer. The output from this

layer is then fed into four parallel streams. Each stream consists of three layers: an ASPP Convolutional Layer, a Normalization Batch Layer, and an ASPP ReLu Layer.

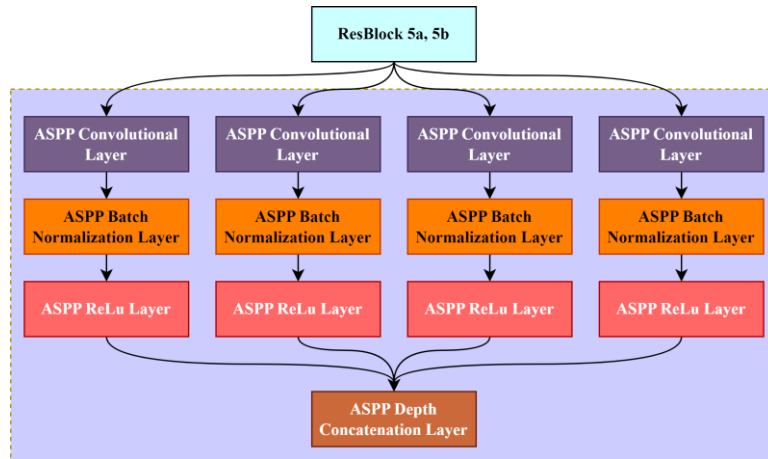


Figure 6. Atrous Spatial Pyramid Pooling Network Structure

The ASPP Convolutional Layer applies a set of convolutional operations to the input data. These operations are designed to extract features from the input at multiple scales, which is crucial for tasks like image segmentation. Following the ASPP Convolutional Layer is the Normalization Batch Layer. This layer normalizes the output from the convolutional layer, which can help improve the stability and performance of the neural network. The ASPP ReLu Layer applies the ReLu (Rectified Linear Unit) activation function to the output from the normalization batch layer. This introduces non-linearity into the model, allowing it to learn more complex patterns in the data. Finally, the outputs from the four parallel streams are combined in the ASPP Depth Concatenation Layer. This layer concatenates the outputs from the four streams along the depth dimension, effectively combining the multi-scale features extracted by each stream into a single output. This architecture is designed to be highly flexible and adaptable, making it suitable for a wide range of tasks that require capturing multi-scale context.

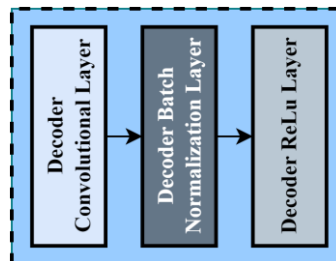


Figure 7. Decoder Block

The decoder architecture consists of Convolution, batch Normalization, and Relu layers, as shown in Figure 7. This sequence of layers is typical in the decoding phase of a convolutional neural network, where the goal is to upsample the feature maps to generate an output of the same size as the original input.

3.5. Proposed Network

Our methodology utilizes a convolutional neural network (CNN) architecture for image segmentation, as shown in Figure 8. The network follows a DeepLabv3+ inspired structure, incorporating residual learning blocks for improved depth, atrous spatial pyramid pooling for multi-scale information capture, and decoder blocks for upsampling feature maps. The network processes the input image through residual stages, extracts feature at multiple scales through the ASPP block, and progressively upsamples and

refines these features in the decoder stages. Finally, the output layer predicts the probability distribution of each pixel belonging to specific classes.

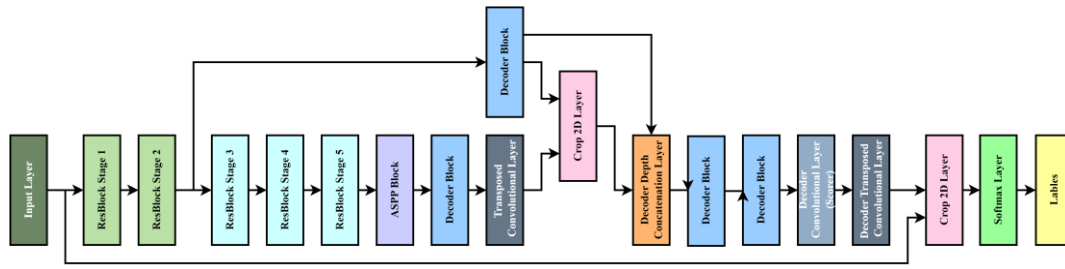


Figure 8. Proposed Network Architecture

3.6. Evaluation Metrics

An evaluation metric quantifies the performance of the predictive model. The Jaccard and Dice similarity coefficients are used to evaluate segmentation models. Several metrics are used to accurately measure the performance of segmentation models, including Accuracy, Precision, Recall, F1-score, the Dice coefficient, and Mean IoU. The Jaccard similarity coefficient, also known as the Jaccard Index [26], is returned as a numeric scalar or a numeric vector ranging from [0, 1]. The segmentations in the two images (segmented images using the network and hand-drawn segmentation) are matched if the similarity value is 1. The Jaccard similarity coefficient is defined as follows:

$$J(P^*, P) = \frac{\sum_{i,j} \binom{N_{ij}}{2}}{\sum_i \binom{N_i}{2} + \sum_j \binom{N_j}{2} - \sum_i \binom{N_{ij}}{2}} \quad (1)$$

Where P^* and P are the two sets for comparing, N_{ij} represents the number of elements common to both sets (the intersection). N_i and N_j represent the total number of elements in each set.

The Dice similarity coefficient value is a straightforward and practical summary metric of spatial overlap that can be used to investigate the reliability and accuracy of image segmentation. This metric can be modified for tasks related to validation. A DSC's value ranges from 0, which denotes complete spatial overlap between two sets of results from binary segmentation, to 1, which denotes total spatial overlap [27]. The most popular evaluation index in semantic segmentation is the Dice coefficient. The greater the similarity between the two samples, the higher the Dice coefficient.

$$DC(A, B) = \frac{2TP}{2TP + FP + FN} = \frac{2|A \cap B|}{|A| + |B|} \quad (2)$$

Where A is the predicted segmentation, and B is the ground truth. TP stands for True Positives, FP for False Positives, TN for True Negative and FN for False Negatives.

$$IoU(A, B) = \frac{TP}{TP + FP + FN} = \frac{|A \cap B|}{|A \cup B|} \quad (3)$$

$IoU(A, B)$ represents the Intersection over Union between two sets A and B . $|A \cap B|$ represents the intersection of sets A and B . $|A \cup B|$ represents the union of sets A and B . Mean-IoU is a metric that takes the IoU over all of the classes and takes the mean of them:

$$MIoU = \frac{1}{k+1} \sum_{i=0}^k \frac{TP}{FN + FP + TP} \quad (4)$$

Where, MIoU stands for Mean Intersection over Union and k is the number of classes in segmentation task.

Indeed, a confusion matrix is a powerful tool for evaluating the performance of a classification model, where N represents the total number of target classes. This $N \times N$ matrix contrasts the actual target values with the predictions made by the machine learning model. The four fundamental elements of a confusion matrix are:

True Positives (TP): These are cases in which the model predicted the positive class correctly.

True Negatives (TN): These are cases in which the model predicted the negative class correctly.

False Positives (FP): These are cases in which the model incorrectly predicted the positive class.

False Negatives (FN): These are cases in which the model incorrectly predicted the negative class.

With these values, we can calculate several performance metrics to assess the effectiveness of a classification model [28]:

Accuracy (ACC): This is the proportion of true results (both true positives and true negatives) among the total number of cases examined. It is calculated as follows:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

Precision: This is the proportion of true positive results among the total predicted positive (both true and false positives). It is calculated as follows:

$$Precision = \frac{TP}{TP + FP} \quad (6)$$

Recall: This is the proportion of true positive results among the total actual positives (both true positives and false negatives). It is also known as sensitivity, hit rate, or true positive rate (TPR). It is calculated as follows:

$$Recall = \frac{TP}{TP + FN} \quad (7)$$

F1-score (F1): This is the harmonic mean of precision and recall, and it provides a balance between them. An F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0. It is calculated as follows:

$$Fscore = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (8)$$

Specificity: This is the proportion of true negative results out of all actual negative results in the population.

$$Specificity = \frac{TN}{TN + FP} \quad (9)$$

These metrics provide a comprehensive view of the model's performance and can help identify the areas where the model needs improvement.

4. Result and Discussion

This experiment was conducted using the MATLAB 2022b environment on a Windows 11, 64-bit operating system. The hardware setup comprises an AMD Ryzen Threadripper processor and an NVIDIA RTX A5000 GPU. The proposed network was trained using Stochastic Gradient Descent with Momentum (SGDM) optimization. The learning rate, initially set at 0.01, was reduced by a factor of 10 every ten epochs. The network underwent training for 100 epochs, with the training speed enhanced by setting a high learning rate.

4.1. Comparing the trained network with other networks

The performance of the trained network was benchmarked against other segmentation algorithms, such as U-Net and Segnet. After numerous trials and adjustments with various epochs and batch sizes, we discovered that superior accuracy was achieved with the following hyper-tuned settings.

4.1.1. Impact of Batch sizes

To optimize the network, we set the learning rate at 0.01 and the number of epochs at 100. The results show that our Proposed Network surpasses both U-Net and Segnet in terms of Batch Accuracy. Notably, the Proposed Network achieved a Batch Accuracy of 97.74% with a Batch Size of 8, significantly outperforming the maximum Batch Accuracy of U-Net (88.72%) and Segnet (92.61%) as shown in Table 1.

Table 1. Comparison of Network by Different Batch Sizes

S. No	Network	Batch Size	Batch Accuracy	Validation Accuracy	Graphical Accuracy
1	U-Net	16	88.72	88.5	88.65
2	U-Net	8	88.46	87.11	87.31
3	Segnet	16	91.76	91.94	92.39
4	Segnet	8	92.61	91.6	91.49
5	Proposed Network	16	97.44	89.59	89.58
6	Proposed Network	8	97.74	92.63	92.61

Interestingly, despite the high batch accuracy, the proposed network's validation and graphic accuracy do not match its batch accuracy. This discrepancy may be attributed to overfitting, where the model excels on the training data but underperforms on unseen data. However, it is noteworthy that the Proposed Network, with a Batch Size of 8, achieved the highest Validation Accuracy and Graphical Accuracy among all models, indicating its superior performance.

Regarding the impact of Batch Size, it seems that a smaller Batch Size generally improves performance across all networks. This could be due to the regularizing effect of smaller batches, which can help prevent overfitting. Future studies could further investigate the influence of other hyperparameters, such as the number of epochs, on the performance of the Proposed Network.

4.1.2. Impact of Epochs in training network

The previous Table 1 reveals that a batch size of 8 yields optimal results. With this batch size, we evaluated the network across different epoch rates. The Proposed Network consistently outperforms both U-Net and Segnet in terms of Batch Accuracy across all epochs. Specifically, with 100 epochs, the Proposed Network achieved a Batch Accuracy of 97.74%, significantly surpassing the maximum Batch Accuracy of U-Net (90.2%) and Segnet (93.88%) as shown in Table 2.

Interestingly, despite the high batch accuracy, the proposed network's validation and graphic accuracy do not match its batch accuracy. This discrepancy may be due to

overfitting, where the model performs well on the training data but underperforms on unseen data. However, it is noteworthy that the Proposed Network, with 50 epochs, achieved the highest Validation Accuracy and Graphical Accuracy among all models, indicating its superior performance.

Table 2. Comparison of Networks by Different Epochs

S. No	Network	Epochs	Batch Accuracy	Validation Accuracy	Graphical Accuracy
1	U-Net	100	88.46	87.11	87.31
2	U-Net	50	87.44	86.8	86.73
3	U-Net	25	87.7	87.66	88.03
4	U-Net	10	90.2	88.96	89.02
5	Segnet	100	92.61	91.6	91.49
6	Segnet	50	93.88	91.59	92.27
7	Segnet	25	91.82	92.38	93.19
8	Segnet	10	92.68	88.37	89.44
9	Proposed Network	100	97.74	92.63	92.61
10	Proposed Network	50	97.31	93.05	94.3
11	Proposed Network	25	97.35	90.16	92.65
12	Proposed Network	10	95.16	92.74	93.26

Regarding the impact of epochs, it appears that a moderate number of epochs (50 for the Proposed Network) generally leads to better performance across all networks. This could be because a larger number of epochs may lead to overfitting. In comparison, a smaller number of epochs may lead to underfitting. These findings suggest that the Proposed Network, particularly with 50 epochs, provides promising results for segmenting tomato leaf diseases.

4.1.3. Evaluation of networks Accuracy based on individual classes.

Table 3 evaluates the accuracy of different trained networks based on individual classes. The Proposed Network boasts the highest mean accuracy of 0.9493. It excels in identifying the background with an accuracy of 0.99243. Also, it demonstrates high accuracy in identifying disease and healthy classes, with accuracies of 0.90267 and 0.9528, respectively. Segnet has a mean accuracy of 0.89999 and a high background accuracy of 0.97851. The accuracies for disease and healthy classes are 0.87239 and 0.84906, respectively. U-Net, with the lowest mean accuracy of 0.78789 among the three, has a high background accuracy of 0.96969. However, its accuracy for disease and healthy classes is significantly lower, with values of 0.64562 and 0.74837, respectively.

Table 3: Comparison of Networks based on accuracy of individual classes

S. No	Trained Network	Mean Accuracy	Background Accuracy	Disease Accuracy	Healthy Accuracy
1	Proposed Network	0.9493	0.99243	0.90267	0.9528
2	Segnet	0.89999	0.97851	0.87239	0.84906
3	U-Net	0.78789	0.96969	0.64562	0.74837

The Proposed Network outperforms Segnet and U-Net in terms of mean accuracy in identifying disease and healthy classes. However, all three networks demonstrate high accuracy in identifying the background class.

4.1.4. Evaluation of networks IoU based on individual classes.

Table 4 evaluates the Intersection over Union (IoU) of different trained networks based on individual classes. The Proposed Network has the highest mean IoU of 0.90413 and the highest weighted IoU of 0.93678. It excels in identifying the background with an

IoU of 0.98259. It also demonstrates high IoUs for disease and healthy classes, with values of 0.87414 and 0.85566, respectively.

Table 4. Comparison of Various Networks IoU based on individual classes













S. No	Trained Network	Mean IoU	Weighted IoU	Background IoU	Disease IoU	Healthy IoU
1	Proposed Network	0.90413	0.93678	0.98259	0.87414	0.85566
2	Segnet	0.82504	0.87865	0.95486	0.76077	0.75948
3	U-Net	0.67604	0.75171	0.8914	0.49993	0.6368

Segnet has a mean IoU of 0.82504 and a weighted IoU of 0.87865. It has a high background IoU of 0.95486. The IoUs for disease and healthy classes are 0.76077 and 0.75948, respectively. U-Net, with the lowest mean IoU of 0.67604 and the lowest weighted IoU of 0.75171 among the three, has a high background IoU of 0.8914. However, its IoUs for disease and healthy classes are significantly lower, with values of 0.49993 and 0.6368, respectively. The Proposed Network outperforms Segnet and U-Net in terms of mean IoU, weighted IoU, and IoUs for disease and healthy classes. However, all three networks demonstrate high IoUs for the background class.

4.2. Segmentation results of proposed network vs. Manual segmentation

The comparison in Table 5 offers a valuable visual representation of the segmentation results from the proposed network against manual segmentation for four different samples. Each sample has an input image, a segmented image, and a hand-drawn image.

Table 5. Segmentation Results of The Proposed Algorithm

S. No	Image Name	Input Image	Segmented image using Proposed network	Hand drawn Image
1	4			
2	14			
3	20			
4	37			

The Input Image presents the original image of a leaf against a plain background. The Segmented Image displays the outcome of the automated segmentation process executed

by the proposed network. In this image, the diseased region is highlighted in grey, the healthy areas in white, and the background in black, indicating the areas identified by the network. The hand-drawn image appears to be manually segmented images of leaves in white against a black background. This comparison visually illustrates how the automated segmentation network can emulate manual segmentation. This is particularly relevant in fields such as computer vision and machine learning, where such networks are utilized for image analysis. However, the performance may vary depending on the images' characteristics and the complexity of the segmentation task.

4.3. Evaluation based on the Jaccard and Dice similarity Coefficient

We employed the Jaccard index and the Dice Coefficient method to assess our segmentation model. This approach involves overlapping one image over another to verify the model's segmentation similarity. In our case, we overlapped network-segmented images on human-segmented images. According to the evaluation metric, the value should range from 0 to 1, where 1 signifies exact overlap, and 0 indicates a mismatch.

For the proposed model, we segmented tomato leaves into the background, disease, and healthy areas. The Jaccard index and Dice coefficient were calculated for these three classes, as shown in Table 6. This table reveals a higher similarity in segmenting the background and healthy regions by both Jaccard and Dice. However, there is a loss in the diseased area, which might be due to manual segmentation.

Manual segmentation is prone to errors compared to neural network-based segmentation. Upon zooming in on the image, it is evident that the network-based segmentation precisely segments the affected region, while the human-segmented region includes areas slightly affected by diseases. This observation underscores the accuracy and precision of network-based segmentation.

Table 6. Jaccard and Dice Similarity of Proposed Network vs. Drawn Segmentation

S: no	Image Name	Jaccard background	Jaccard Disease	Jaccard Healthy	Dice Background	Dice Disease	Dice Healthy
1	4	0.9838	0.6437	0.89622	0.99183	0.7833	0.94527
2	14	0.98586	0.77728	0.96731	0.99288	0.87468	0.98339
3	20	0.98234	0.83314	0.9145	0.99109	0.90898	0.95534
4	37	0.98685	0.97332	0.80714	0.99338	0.98648	0.89328

The network was assessed using various random images, and the results were quantified using the Jaccard index and Dice coefficients for the background, disease, and healthy classes. The network consistently demonstrated high performance in identifying the background class, as evidenced by high Jaccard and Dice coefficients. The performance of disease and healthy classes varied across images, with some images yielding higher coefficients than others.

4.4. Evaluation based on other metrics

Table 7 evaluates the network based on various metrics for different images. The network was tested on images 14, 20, 37, and 4. The Intersection over Union (IoU) for the background (BG) class was consistently high across all images, ranging from 0.98326 to 0.98736. The IoU for the disease class showed more variation, ranging from 0.64637 to 0.97549. The IoU for the healthy class also varied, ranging from 0.86334 to 0.96665.

The Accuracy (Acc), precision (Prec), recall (Rec1), F-score (Fscore) and Specificity (Spe) were all high across the images, indicating the network's overall strong performance. The specificity, a measure of the true negative rate, was also high across all images, ranging from 0.9801 to 0.9954. The network demonstrated robust performance across different images and classes, as indicated by the high values of the various metrics.

Table 7. Other Evaluation Metrics for Proposed Segmentation Network

Input Image	BG IoU	Disease IoU	Healthy IoU	Acc	Prec	Recl	Fscore	Spe
14	0.98736	0.79989	0.96665	0.9934	0.9902	0.9902	0.9902	0.9951
20	0.98364	0.815	0.91244	0.9806	0.9709	0.9709	0.9709	0.9854
37	0.98326	0.97549	0.86334	0.9939	0.9909	0.9909	0.9909	0.9954
4	0.98376	0.64637	0.90003	0.9734	0.9602	0.9602	0.9602	0.9801

However, the performance on the disease class, as measured by the IoU, showed more variation across images. This suggests that the network's ability to identify the disease class may depend on specific characteristics of the images.

5. Conclusion

The proposed network demonstrated robust performance across different images and classes, as indicated by the high values of various metrics. It achieved a high mean accuracy and Intersection over Union (IoU) for the background, disease, and healthy classes. However, the performance of the disease class showed more variation across images, suggesting that the network's ability to identify the disease class may depend on the specific characteristics of the images. Despite this, the proposed network, particularly with a batch size of 8 and 50 epochs, offers promising results for segmenting tomato leaf diseases. Future research may focus on improving the validation and graphical accuracy of the proposed network, possibly through techniques such as regularization, early stopping, or data augmentation. Additionally, the impact of other hyperparameters, such as learning rate and number of epochs, on the performance of the proposed network could be explored.

REFERENCES

- [1] Y. Deng, H. Xi, G. Zhou, A. Chen, Y. Wang, L. Li, et al., "An effective image-based tomato leaf disease segmentation method using mc-unet", *Plant Phenomics*, vol. 5, (2023).
- [2] H. David, K. Ramalakshmi, R. Venkatesan, & G. Hemalatha, "Tomato leaf disease detection using hybrid cnn-rnn model", (2021).
- [3] S. Islam, "Enhanced deep learning architecture for rapid and accurate tomato plant disease diagnosis", *Agriengineering*, vol. 6, no. 1, (2024), pp. 375-395.
- [4] L. Saxena, "Evaluation of enhanced resnet-50 based deep learning classifier for tomato leaf disease detection and classification", *Journal of Electrical Systems*, vol. 20, no. 3s, (2024), pp. 2270-2282.
- [5] H. Vo, "Tomato disease recognition: advancing accuracy through xception and bilinear pooling fusion", *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 8, (2023).
- [6] S. Anam, I. Yanti, Z. Fitriah, & M. Assidiq, "Tomato leaf disease segmentation using clustering method based on fatps with multi features", *International Journal of Emerging Technology and Advanced Engineering*, vol. 13, no. 1, (2023), pp. 28-36.
- [7] L. Li, B. Wang, Y. Li, & H. Yang, "Diagnosis and mobile application of apple leaf disease degree based on a small-sample dataset", *Plants*, vol. 12, no. 4, (2023), p. 786.
- [8] S. Islam, S. Sultana, F. Farid, N. Islam, M. Rashid, B. Bari et al., "Multimodal hybrid deep learning approach to detect tomato leaf disease using attention based dilated convolution feature extractor with logistic regression classification", *Sensors*, vol. 22, no. 16, (2022), p. 6079.
- [9] R. Yang, "Semantic segmentation of cucumber leaf disease spots based on eca-segformer", *Agriculture*, vol. 13, no. 8, (2023), p. 1513.
- [10] M. Chowdhury, T. Rahman, A. Khandakar, M. Ayari, A. Khan, M. Khan et al., "Automatic and reliable leaf disease detection using deep learning techniques", *Agriengineering*, vol. 3, no. 2, (2021), pp. 294-312.

- [11] S. Anam, I. Yanti, Z. Fitriah, & M. Assidiq, "Tomato leaf disease segmentation using clustering method based on fatpso with multi features", *International Journal of Emerging Technology and Advanced Engineering*, vol. 13, no. 1, (2023), pp. 28-36.
- [12] W. Wiharto, F. Nashrullah, E. Suryani, U. Salamah, N. Prakisy, & S. Setyawan, "Texture-based feature extraction using gabor filters to detect diseases of tomato leaves", *Revue D Intelligence Artificielle*, vol. 35, no. 4, (2021), pp. 331-339.
- [13] M. Bhandari, T. Shahi, A. Neupane, & K. Walsh, "Botanicx-ai: identification of tomato leaf diseases using an explanation-driven deep-learning model", *Journal of Imaging*, vol. 9, no. 2, (2023), p. 53.
- [14] P. Zer, F. Tambunan, R. Rosnelly, & W. Wanayumini, "Comparison of tomato leaf disease classification accuracy using support vector machine and k-nearest neighbor methods", *Sinkron*, vol. 8, no. 2, (2023), pp. 939-947.
- [15] R. Kumar, J. Athimoolam, A. Appathurai, & S. Rajendiran, "Novel segmentation and classification algorithm for detection of tomato leaf disease", *Concurrency and Computation Practice and Experience*, vol. 35, no. 12, (2023).
- [16] C. Vengaiyah, "A comparative study of convolutional neural network architectures for enhanced tomato leaf disease classification using refined statistical features", *Traitement Du Signal*, vol. 41, no. 1, (2024), pp. 201-212.
- [17] C. Zhou, S. Zhou, J. Xing, & J. Song, "Tomato leaf disease identification by restructured deep residual dense network", *Ieee Access*, vol. 9, (2021), pp. 28822-28831.
- [18] T. Inan and U. Kacar, "Ear semantic segmentation in natural images with tversky loss function supported deeplabv3+ convolutional neural network", *Balkan Journal of Electrical and Computer Engineering*, vol. 10, no. 3, (2022), pp. 337-346.
- [19] Z. Wang, C. Wang, Y. Qiu, Y. Liu, & J. Wang, "Real-time identification of cyanobacteria blooms in lakeshore zone using camera and semantic segmentation: a case study of lake chaohu (eastern china)", *Sustainability*, vol. 15, no. 2, (2023), p. 1215.
- [20] B. Sayraci, M. Ağrali, & V. Kilic, "Artificial intelligence based instance-aware semantic lobe segmentation on chest computed tomography images", *European Journal of Science and Technology*, (2022).
- [21] C. Karabağ, M. Jones, C. Peddie, A. Weston, L. Collinson, & C. Reyes-Aldasoro, "Semantic segmentation of hela cells: an objective comparison between one traditional algorithm and four deep-learning architectures", *Plos One*, vol. 15, no. 10, (2020), p. e0230605.
- [22] J. Zhang, "Afc-resnet18: a novel real-time image semantic segmentation network for orchard scene understanding", *Journal of the Asabe*, vol. 67, no. 2, (2024), pp. 493-500.
- [23] V. Badrinarayanan and R. Cipolla, "Segnet: a deep convolutional encoder-decoder architecture for image segmentation", *Ieee Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, (2017), pp. 2481-2495.
- [24] H. Keles and L. Lim, "Learning dense contextual features for semantic segmentation", *Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering*, vol. 62, no. 1, (2020), pp. 26-34.
- [25] A. Xiao, B. Shen, J. Tian, & Z. Hu, "Pp-nas: searching for plug-and-play blocks on convolutional neural networks", *Ieee Transactions on Neural Networks and Learning Systems*, (2024), pp. 1-13.
- [26] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "Clustering validity checking methods," *ACM SIGMOD Record*, vol. 31, no. 3, (2002), pp. 19-27.
- [27] K. H. Zou et al., "Statistical validation of image segmentation quality based on a spatial overlap index1," *Academic Radiology*, vol. 11, no. 2, (2004), pp. 178-189.
- [28] P. Shi, M. Duan, L. Yang, W. Feng, L. Ding, and L. Jiang, "An Improved U-Net Image Segmentation Method and Its Application for Metallic Grain Size Statistics," *Materials*, vol. 15, no. 13, (2022), p. 4417. Detection and Classification - ScienceDirect")