ISSN: 2584-0495 Vol. 2, Issue 12, pp. 1394-1400

International Journal of Microsystems and IoT

ISSN: (Online) Journal homepage: https://www.ijmit.org

Analysis of Tomato Leaf Diseases using a Deep Learning Model

Deepa G, Jaya Kishna V S, Kavya T and Kamalesh R

Cite as: Deepa, G., Jaya Kishna, V. S., Kavya, T., & Kamalesh, R. (2024). Analysis of Tomato Leaf Diseases using a Deep Learning Model. International Journal of Microsystems and IoT, 2(12), 1394–1140. https://doi.org/10.5281/zenodo.15076203

Published online: 24 December 2024

DOI: https://doi.org/10.5281/zenodo.15076203

International Journal of Microsystems and IoT
Vol.2, Issue 12, pp.1394-1400 ; DOI: https://doi.org/10.5281/zenodo.15076203

Analysis of Tomato Leaf Diseases using a Deep Learning Model

Deepa G, Jaya Kishna V S, Kavya T and Kamalesh R

Department of Electronics and Communication Engineering, *Kongu Engineering College, Erode, India*

ABSTRACT

Plant leaf diseases cause substantial annual production losses for farmers, impacting their primary food source. Minimizing these losses requires early detection of diseases. A deep learning approach is used to address the early identification of tomato leaf diseases, enabling farmers to take preventive measures and reduce production loss. A Customized six-layer Convolutional Neural Network (CNN) is suggested for the identification of diseases in tomato plant leaves, aiming to mitigate annual production losses for farmers. The CNN utilizes automatic feature extraction, requiring no explicit feature engineering, enabling finer disease classification. Employing 11,100 leaf images from the Plant Village dataset, the model classifies ten classes, including nine distinct diseases and one healthy class. With an 80/20 dataset split, 30 epochs, and a 0.001 learning rate, the CNN achieves a overall accuracy of 91.3%. The focus on computational efficiency addresses the critical issue of early and accurate disease detection, potentially boosting agricultural productivity and affordability for consumers. Comparative analysis with VGG-16 and VGG-19 using transfer learning reveals the superior performance of the proposed model. Its simplicity not only reduces parameters but also facilitates deployment on lightweight devices, significantly reducing training time and the simulation utilized Google Colab. It also emphasizes the effectiveness of a simplified approach in addressing crucial challenges in tomato disease identification, with potential applicability to other crops and plants.

KEYWORDS
Deep learning,
Customized CNN,
Transfer learning,
Leaf disease classification,
hyperparameters.

I. INTRODUCTION

Global agriculture, a major source of vegetables crucial for economic growth and food security [11-13], [19], [20], heavily relies on tomato production. However, the existence of diseases represents a potential threat, leading to substantial yield losses. Tomatoes are vulnerable to various diseases caused by bacteria, fungi, and viruses. Tomato leaf diseases encompass Early Blight, Leaf mold, late blight, Bacterial Spot, Septoria Leaf Spot, Target Spot, Yellow Leaf Curl Virus, and Mosaic Virus [14-16]. Various tomato leaf diseases demand specific management strategies to safeguard crop health. Images of infected tomato leaves and a healthy leaf are shown in Figure 1.

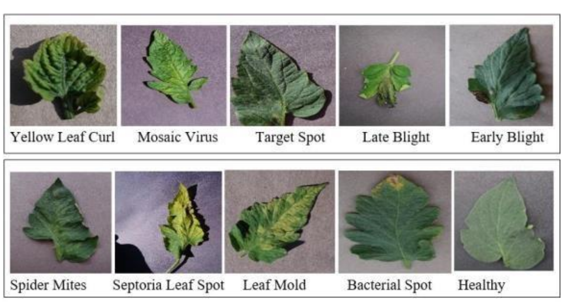


Fig. 1. Random visualization of tomato leaf images

II. LITERATURE REVIEW

Recent advancements in tomato disease identification have seen significant progress through the application of advanced techniques to improve accuracy and practicality. Convolutional Neural Networks (CNNs) have played a central role, as high- lighted in recent studies. One notable research effort employed various models, including InceptionV3, ResNet152, VGG19, and EfficientNetB0-B4, achieving impressive accuracy rates ranging from 91.07% to an exceptional 99.23%. The proposed deep learning architecture, incorporating transfer learning with VGGNet and an enhanced categorical cross-entropy loss func- tion, stands out as a crucial tool for farmers, ensuring precise detection and protection of tomato crops [1].

Another study underscores the significance of deep learning in agricultural disease identification, emphasizing the role of data augmentation for improved model generalization. Uti- lizing GANs like CycleGAN and LeafGAN, Attaining

94.33 percent, the top-1 average identification accuracy reflects the model's strong performance, demonstrating its capability to accurately classify items in the given dataset comprising 1500 tomato leaf images categorized into five classes [2]. A modified RDN model has been designed for the identification of diseases in tomato leaves. This hybrid deep learning approach showcases effectiveness in reducing training process parameters and enhancing calculation accuracy [3].

By combining leaf photos, (CNNs), and a chatbot controller, an integrated system is implemented to identify eight categories of pests and illnesses affecting tomatoes. This method ensures a holistic and interactive solution for managing plant health. The study contributes to the intersection of agriculture, computer vision, and conversational AI for efficient plant disease identification [4]. Addressing the challenge of low classification accuracy due to limited samples in tomato disease classification, a proposed method, MMDGAN, achieving a 97.129% accuracy and an FI (Fowlkes-Mallows Index) value of 97.78% on open datasets, such as Plant Village [5], establishes notable performance. Particularly in the domain of Sectioning and identifying tomato plant leaves autonomously, these metrics signify advanced capabilities in precision and effectiveness, Mask R-CNN exhibits robust performance [6]. Innovation continues with the introduction of LMBRNet, an advanced method for tomato leaf disease identification that surpasses existing models with fewer parameters. This method showcases effectiveness with 80 images for training and validation and 20 for testing [7].

A study using Xception achieved 99.45% accuracy in tomato leaf disease detection, highlighting timely identification's importance with the Plant Village dataset [9]. A pre- trained MobileNetV2 architecture demonstrates remarkable efficiency with a 99.30% accuracy, this approach emphasizes a small model size and low computational cost, contributing to its lightweight nature [10].

III. MATERIALS AND METHODS

The technical workflow involves data acquisition, pre-

A. Process pipeline

processing, network training, and testing and evaluation [1]. The Plant Village dataset is where the input images are obtained [8]. The dataset is subsequently partitioned into training, validation, and testing sets for comprehensive model development and assessment. A constant 224 x 224-pixel size is applied to the images. Data augmentation is performed to increase the training accuracy [5]. The compact convolutional networks and VGG architectures are trained using the dataset [6], [7]. Finally, the classes of tomato leaf are predicted using test dataset. The schematic representation delineates the proposed system architecture for the classification of tomato leaf diseases, shown in Figure 2.

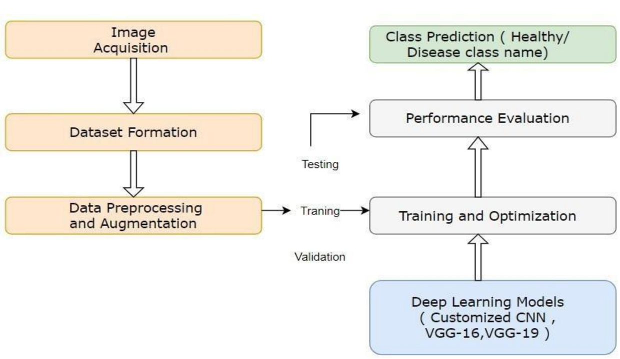


Fig. 2. Proposed workflow for classification of tomato plant leaf

B. Image acquisition and dataset

Create an experimental dataset that will be used for train- ing, validation, and performance evaluation of the proposed architecture. The public dataset [8] consists of 11,000 images that formed the basis of the dataset. The classification sys- tem encompasses various disease categories in tomato leaves, including leaf mold, tomato yellow leaf curl virus, bacterial spot, mosaic virus, target spot, late blight, early late blight, two-spotted spider mite, and Septoria spot. Additionally, one category is dedicated to the classification of healthy leaves. These images are pre-processed with a deep learning input dimensional model (specifically 224*224*3) and divided into 10 classes.

C. Data pre-processing and augmentation

The raw input images undergo standard transformations to create an augmented dataset [13]. Necessary libraries for training, testing, and graphical output are imported. Due to variations in image sizes from different sources, images are resized during preprocessing to adhere to a standard dimension model (specifically 224x224) [15] and saved in JPEG format. Augmentation aims to maximize the training dataset by introducing slight distortions [19], minimizing overfitting. To enhance testing performance, image transformations are applied to various rotated images during testing. Images are augmented using an arbitrary combination of different image augmentation methods shown in Figure 3.

```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1/255.0,
                                  shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   validation_split=0.3)
```

Fig. 3. Sample Augmentation code

Using the above augmented code technique, an image is created that is different from the original image. During the conversion, the image is not saved to disk and requires no storage space, because the converted image is created at runtime and is computationally efficient. Augmentation techniques can be used to address the problem of overfitting, while improving test performance [17]. The sample augmented images are represented in Figure 4.

Fig. 4. Augmented images using the original image

D. Customized CNN Architecture

In the proposed work on customized CNN architecture, six convolutional layers followed by six max-pooling layers, one flattening and three dense layers are used. In this work, Keras Sequential API is used to build a CNN model. Sequential mode instances are created using the Sequential class. To get a snapshot of the entire CNN architecture considering all parameters, the summary method is used, as shown in Figure 5. convolutional layers in a CNN extract features, preserving local context [22]. Pooling layers reduce pixel dimensions, parameters, and remove noise. The final dense layer assigns class labels, driven by an activation function, concluding image classification.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_12 (Conv2D)          (None, 222, 222, 32)      896

 max_pooling2d_12 (MaxPoolin  (None, 111, 111, 32)     0
 g2D)

 conv2d_13 (Conv2D)          (None, 109, 109, 64)      18496

 max_pooling2d_13 (MaxPoolin  (None, 54, 54, 64)       0
 g2D)

 conv2d_14 (Conv2D)          (None, 52, 52, 64)        36928

 max_pooling2d_14 (MaxPoolin  (None, 26, 26, 64)       0
 g2D)

 conv2d_15 (Conv2D)          (None, 24, 24, 64)        36928

 max_pooling2d_15 (MaxPoolin  (None, 12, 12, 64)       0
 g2D)

 conv2d_16 (Conv2D)          (None, 10, 10, 64)        36928

 max_pooling2d_16 (MaxPoolin  (None, 5, 5, 64)         0
 g2D)

 conv2d_17 (Conv2D)          (None, 3, 3, 64)          36928

 max_pooling2d_17 (MaxPoolin  (None, 1, 1, 64)         0
 g2D)

 flatten_2 (Flatten)         (None, 64)                0

 dense_5 (Dense)             (None, 64)                4160

 dense_6 (Dense)             (None, 32)                2080

 dense_7 (Dense)             (None, 10)                330

=================================================================
Total params: 173,674
Trainable params: 173,674
Non-trainable params: 0
_________________________________________________________________
```

Fig. 5. Customized CNN model

E. Transfer learning

Transfer learning is a technique in the field of machine learning whereby a pre-existing model, which has been initially trained for a specific task, is leveraged as a foundation to construct a novel model for a different task.VGG-16 and VGG- 19 model are taken as transfer learning framework which is trained on ImageNet datasets [21]. The proposed methodology incorporates distinct training and testing phases. In the training phase, Pretrained VGG-16 and VGG-19 models are taken as the base model then the Flatten Layer, Dense layers, and dropout layer are added to the pre-trained VGG models. The output layer, tailored to the number 10 classes, uses softmax activation to compute class probabilities. Once the model has converged on new data, the model undergoes evaluation using previously unseen data. during the phase of testing, the input that has undergone pre-processing data is employed for assessing the model's performance and the results of the classification are acquired by means of the Softmax layer. [18].

F. CNN Model

a) VGG-16: VGG-16 architecture uses 3x3 convolutional filters and 2x2 max-pooling layers as shown in Figure 6 , which allows for a deep network without excessive complexity. VGG-16, part of the VGG model family, developed at the University of Oxford by the Visual Geometry Group. excels in image classification due to its uniform design and pre-training on the extensive ImageNet dataset [18]. Widely used, its 16- layer depth makes it computationally demanding. The final fully connected layers serve as a classifier, making VGG-16 suitable for diverse image recognition tasks, with the flatten layer converting its output into a flat vector for further processing. The custom added layers include two densely connected layers with rectified linear unit (ReLU) activation. functions (4096 and 1072 neurons) to learn high-level features. The Dropout Layer helps to prevent overfitting by randomly deactivating some neurons (20% dropout rate). The output layer with several neurons equal to the classification classes which is 10, uses softmax activation to produce class proba- bilities.

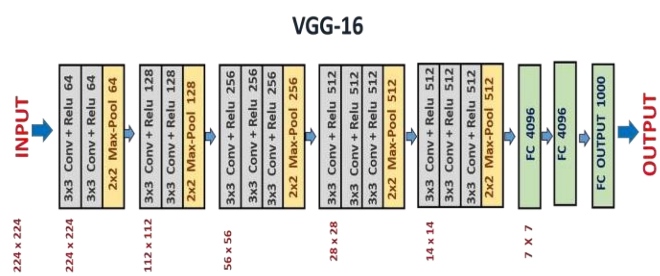


Fig. 6. VGG-16 Model Architecture

VGG-19: VGG-19 is a deep a CNN that uses 3x3 convolutional filters and 2x2 max-pooling layers as shown in Figure 7. It consists of 19 layers, making it deeper than VGG-16. VGG-19 is well-regarded for its image classification capabilities and is part of the VGG family of models developed at the University of Oxford by the Visual Geometry Group. The model's depth allows it to capture intricate visual features. VGG-19's final layers serve as a classifier, making it suit- able for various image recognition tasks. However, its depth also makes it computationally demanding during training and deployment.VGG-19 is incorporated with the same layers as VGG-16, including the Flatten Layer, two Dense layers (4096 and 1072 ReLU-activated neurons) functions, and a Layer of Dropouts with a rate of twenty percent.

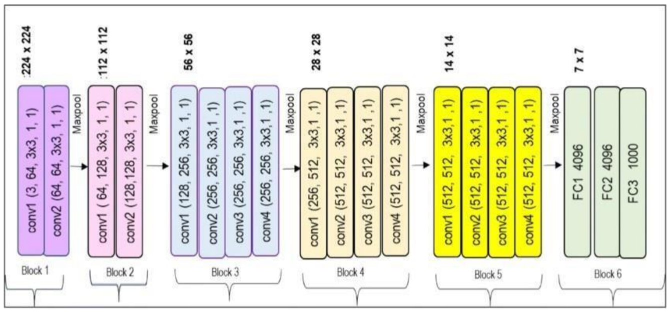


Fig. 7. VGG-19 Model Architecture

IV. RESULTS AND DISCUSSION

Table I provides the detailed overview of the hyperparameter employed in the proposed model.

TABLE I
HYPERPARAMETERS FOR THE PROPOSED CUSTOMIZED CNN

Hyperparameters	Description
No. of convolution layers	6
No. of max pooling layers	6
Activation function	Relu (Rectified Linear Unit)
Learning rate	0.001
Number of epochs	30
Batch size	32

The comparative analysis is conducted by comparing the proposed network against the transfer learning approach such as VGG-16 and VGG-19 using the same dataset. Following training, the proposed model achieves 92.73% validation accuracy and a 95.84% training accuracy. The accuracy progression across the validation and training epochs is shown in the accuracy graph in Figure 8. With increasing epochs, validation accuracy as well as training accuracy both gradually increases. The associated shift in the model's loss function parameters over the training and validation phases is displayed in the loss function graph in Figure 9.

Table II implies a comparison between the Customized CNN, VGG-16 and VGG-19 architectures.

TABLE II

COMPARISON TABLE OF CUSTOMIZED CNN AND VGG ARCHITECTURES

Model	Train Acc.	Train Loss	Val Acc.	Val Loss
VGG-16	96.77%	0.100	88.93%	0.465
VGG-19	97.79%	0.068	88.16%	0.446
Customized CNN	95.84%	0.112	92.73%	0.211

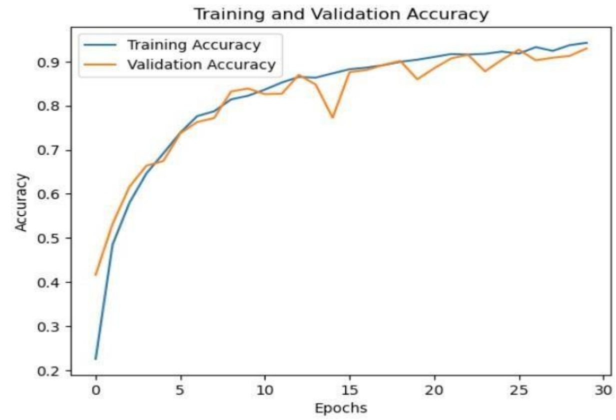


Fig. 8. Accuracy graph for training and validation of Customized CNN

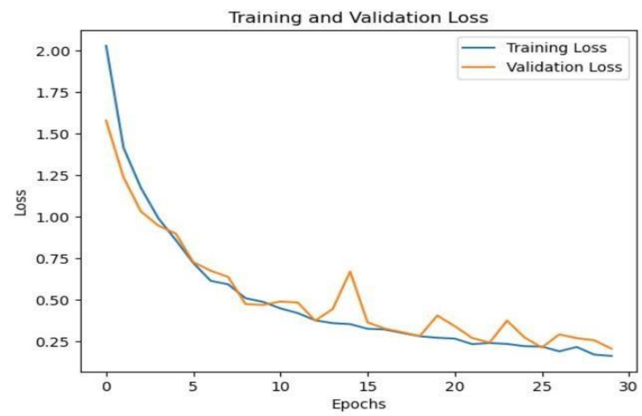


Fig. 9. Loss graph for training and validation of Customized CNN

A. Performance Evaluation

Analysis was conducted with three distinct neural network structures, including VGG-16, VGG-19, and a customized CNN architecture with reduced parameters. The cells on the right and left of Figure 10 emphasize this disparity.

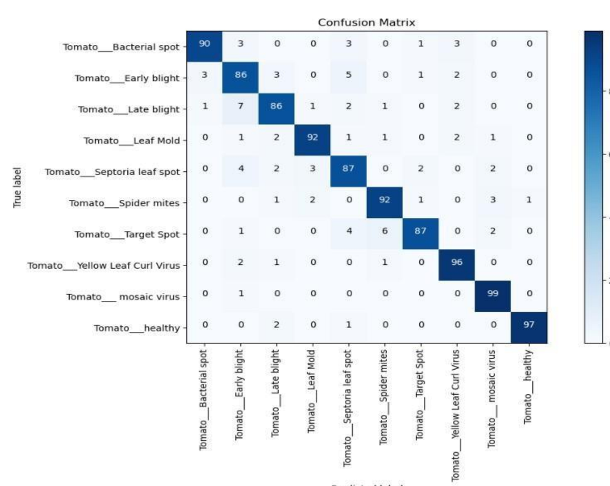
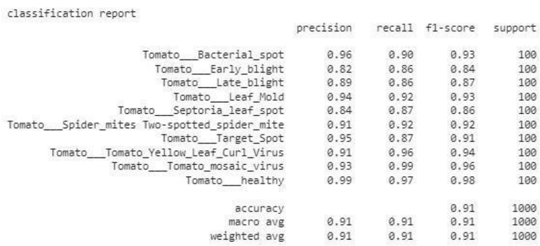


Fig. 10. Confusion Matrix for Proposed Customized CNN

The Customized CNN architecture outperforms both the VGG-19 and VGG-16 models in terms of disease classification. It consistently achieves higher precision, recall, and F1-score values across all classes as shown in Figure 11.


classification report

	precision	recall	f1-score	support
Tomato___Bacterial_spot	0.96	0.90	0.93	100
Tomato___Early_blight	0.82	0.86	0.84	100
Tomato___Late_blight	0.89	0.86	0.87	100
Tomato___Leaf_Mold	0.94	0.92	0.93	100
Tomato___Septoria_leaf_spot	0.84	0.87	0.86	100
Tomato___Spider_mites Two-spotted_spider_mite	0.91	0.92	0.92	100
Tomato___Target_Spot	0.95	0.87	0.91	100
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.91	0.96	0.94	100
Tomato___Tomato_mosaic_virus	0.93	0.99	0.96	100
Tomato___healthy	0.99	0.97	0.98	100
accuracy			0.91	1000
macro avg	0.91	0.91	0.91	1000
weighted avg	0.91	0.91	0.91	1000

Fig. 11. Classification Report for Proposed Customized CNN

The bar chart shown in Figure 12 visually represents the outcomes of testing CNN and VGG models for disease classification. The CNN model demonstrates higher accuracy and fewer misclassifications compared to the VGG model across various tomato diseases. The Customized CNN model is the recommended option for precise disease prediction.

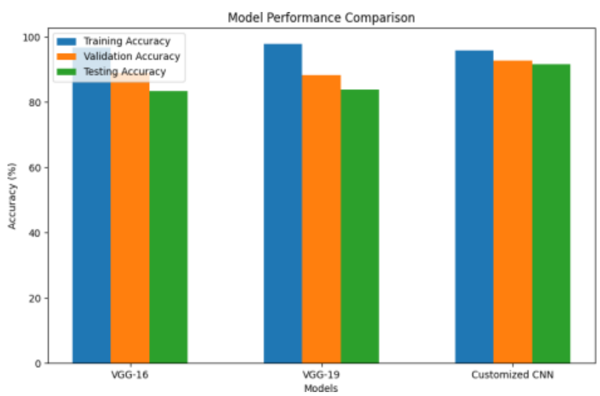


Fig. 12. Testing results comparison between the Customized CNN and VGG architecture

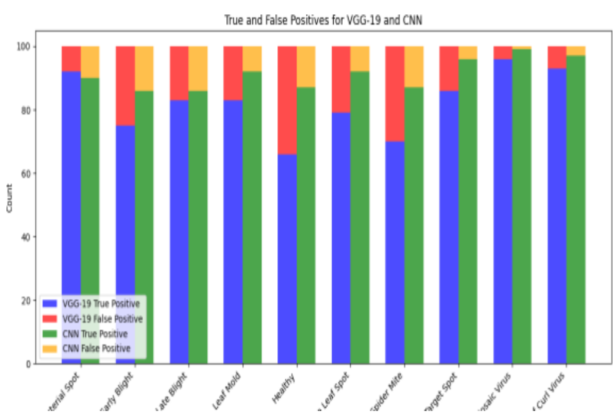


Fig.13. Comparison of classification results

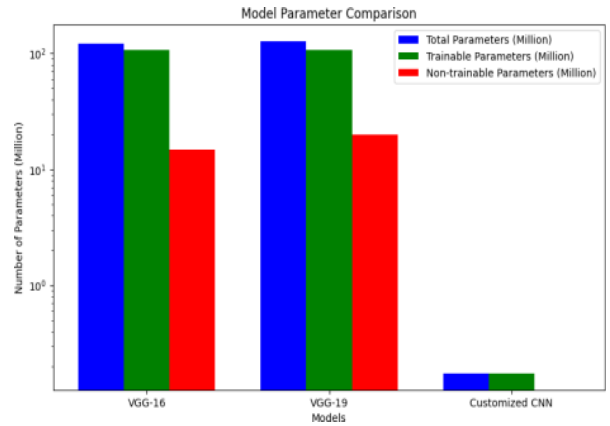


Fig. 14. Comparison of parameters

sentation includes a side-by-side comparison of real images and the model's predictions, illustrating how well the custom CNN model performs in accurately classifying and predicting the content of these images. The bar chart shown in Figure 13 illustrates the classifi- cation results for VGG-16, VGG-19, and a Customized CNN model. The Customized CNN model exhibited a slightly lower training accuracy of 95.84%. However, in terms of validation and testing accuracy, the Customized CNN outperformed the VGG models, achieving a validation accuracy of 92.73% and a testing accuracy of 91.50%. This suggests that the Customized CNN demonstrated superior generalization to unseen data, highlighting its efficacy in real-world applications.

The bar chart shown in Figure 14 provides a visual comparison of the total, trainable, and non-trainable parameters for VGG-16, VGG-19, and a Customized CNN model. The VGG models have significantly more total parameters, mainly due to their deep architecture. However, the Customized CNN model is lightweight, with all parameters being trainable and no non-trainable parameters. This highlights the efficiency of the Customized CNN in terms of parameter count, making it a more streamlined option for scenarios with limited computational resources compared to the VGG models[23-24].

In Figure 15 actual and predicted images during testing are shown using a Customized CNN model. This visual representation

Fig. 15. Prediction output

Early disease detection is vital for improving tomato production, efficiency, and product quality. To identify tomato leaf diseases, an effective CNN architecture was presented. This model prioritizes computational efficiency by avoiding overly complex layers and parameters. Using a ten-class dataset containing nine disease categories and healthy leaves, this method outperforms pre-trained networks. This emphasizes that superior disease identification is attainable without intricate architecture, making this model practical and efficient for deployment. The results exhibit the efficacy of the Customized CNN architecture in classifying tomato leaf diseases. With a training accuracy of 95.84%, the model exhibits strong learning capabilities during the training phase. This indicates that it has successfully learned the features and patterns associated with the different classes. The high validation accuracy of 92.73% further validates the model's performance. This suggests that the Customized CNN generalizes well to unseen data, which is a crucial aspect for real-world applications. The impressive testing accuracy of 91.5% solidifies the model's proficiency in accurately identifying tomato diseases in practical scenarios. This shows the effectivenes of the Customized CNN in making accurate predictions on new, unseen data.

V. CONCLUSIONS

Overall, the Customized CNN architecture showcases robust performance across the board, demonstrating its potential as a powerful tool for automating the identification and categorization of leaf diseases in tomatoes. The combination of high accuracy rates in training, validation, and testing phases affirms the model's effectiveness and suitability for practical deployment in agricultural settings. For future improvements, the application can be integrated into web services for wider network use. Expanding the dataset with diverse disease images will boost model performance. Real-time monitoring systems can enable instant disease detection in agricultural fields through live video analysis. Enhancements can also include providing disease localization and severity details. Extending the model's applicability to various crops will further contribute to agricultural and food security efforts.

REFERENCES

[1] Tabakis, I. M. et.al. (2023). A Robust Hybrid Deep Convolutional Neural Network for COVID-19 Disease Identification from Chest X-ray Images. Information 14(6), 310.

[2] Wang, T., et.al. (2023). RiceNet: A two-stage machine learning method for rice disease identification. Biosystems Engineering 225(1), 25-40.

[3] Zhou, S., et.al. (2021). Tomato leaf disease identification by restructured deep residual dense network. 9(1), 28822-28831.

[4] Hemalatha, A., et.al. (2021). Automatic tomato leaf diseases classification and recognition using transfer learning model with image processing techniques. In 2021 Smart Technologies, Communication and Robotics (STCR), pp. 1-5.

[5] Liangji, Z., et.al. (2022). MMDGAN: A fusion data augmentation method for tomato-leaf disease identification. Applied Soft Computing 123 (1), 108969.

[6] Prabhjot, K., et.al. (2022). An approach for characterization of infected area in tomato leaf disease based on deep learning and object detection technique. Engineering Applications of Artificial Intelligence. 115 (1),105210.

[7] Chen, A., et.al. (2023). Identification of tomato leaf diseases based on LMBRNet. Engineering Applications of Artificial Intelligence. 123(1), 106195.

[8] Tomato Leaf Disease Detection. Available online: https://www.kaggle.com/datasets/kaustubhb999/tomatoleaf.

[9] Anandhakrishnan, T., et.al. (2020). Identification of tomato leaf disease detection using pretrained deep convolutional neural network models. Scalable Computing: 21(4), 625-635.

[10] Ahmed, T., et.al. (2022). Less is more: Lighter and faster deep neural architecture for tomato leaf disease classification.10 (1), 68868-68884.

[11] Agarwal, M., et.al. (2020). TOLED: Tomato Leaf Disease Detection using convolution neural network 167(1), 293–301.

[12] Ahmad, I., et.al. (2020). Optimizing pretrained convolutional neural networks for tomato leaf disease detection, Complexity, 20(1), 1–6.

[13] Aishwarya, N., et.al. (2022). Smart farming for detection and identification of tomato plant diseases using light weight deep neural network , Multimedia Tools and Applications 82(18),18799–18810.

[14] Bhandari, M., et.al. (2023). Botanicx-ai: Identification of tomato leaf diseases using an explanation- driven deep-learning model 9(2), 53.

[15] Deshpande, R., et.al. (2023). Detection of leaf disease in tomato plants using a lightweight parallel deep convolutional neural network, Archives of Phytopathology and Plant Protection, 11(1), 707–720.

[16] Dhingra, G., et.al. (2018). Study of digital image processing techniques for leaf disease detection and classification, Multimedia Tools and Applications 77(1), 19951-20000.

[17] Kokate, J. K., et.al. (2023). Classification of Tomato Leaf Disease using a Custom Convolutional Neural Network , Current Agriculture Research Journal 11(1),315-325.

[18] Kumar, A., et.al. (2023). Automatic recognition and classification of Tomato leaf diseases using transfer learning model, Future Farming: Advancing Agriculture with Artificial Intelligence 13(1), 23–40.

[19] Rubanga, D., et.al. (2020). Early identification of Tuta absoluta in tomato plants using Deep Learning, Scientific African 10(1), 1000590- 1000601.

[20] Patnayakuni, S.P., et.al. (2022). Tomato: Different leaf disease detection using transfer learning based network, Journal of Mobile Multimedia 6(5), 1-11.

[21] Rajathi, N., et.al. (2022). Early stage prediction of plant leaf diseases using Deep Learning Models, Algorithms for Intelligent Systems 2(1), 245–260.

[22] Sakkarvarthi, G., et.al. (2022). Detection and classification of tomato crop disease using convolutional neural network 11(1),3618-3632.

[23] Sarkar S., Kerketta A., **Nath V.** (2021) Kisaan Seva—A Web site for Serving the Farmers. In: Nath V., Mandal J. (eds) Nanoelectronics, Circuits and Communication Systems. Lecture Notes in Electrical Engineering, 692. Springer, Singapore. https://doi.org/10.1007/978-981-15-7486-3_61

[24] Lakra,A.A., Murmu,K., Prasad, D. and **Nath, V.** (2019) Study and Development of Solar-Powered Water Pumping System. In: Nath V., Mandal J. (eds) Proceedings of the Third International Conference on Microelectronics, Computing and Communication Systems, Lecture Notes in Electrical Engineering, 556(1), 655-660 . Springer, Singapore DOI: 10.1007/978-981-13-7091-5_56.