ToMate

Tomato Leaf Disease Prediction using Deep Learning

Hathik Ihthizam - COHNDDS24.1F-012

Nuhan Gunasekara - COHNDDS24.1F-013

Bachelor of Science (Hons) in Data Science

National Institute of Business Management

Colombo, Sri Lanka.

Table of Content

1. Introduction

ToMate is an AI-powered tool designed to help farmers quickly and accurately detect diseases in tomato plants. Tomato crops often suffer from various diseases that affect their yield and quality. Traditionally, farmers rely on manual inspection, which can be slow, labor-intensive, and prone to errors. ToMate solves this problem by using deep learning to analyze images of tomato leaves and detect diseases automatically.

The primary users of ToMate include farmers, agricultural researchers, agritech companies, and government agencies working in agriculture. This tool benefits both small and large-scale farms by providing a fast and efficient way to monitor plant health.

ToMate has a significant impact on the agricultural industry by reducing the time and effort required for disease detection. It helps prevent crop losses, reduces the need for excessive pesticide use, and improves overall farm productivity. By automating disease identification, ToMate enhances decision-making, leading to better crop management and increased profits for farmers.

Problem Statement

Tomato plants are susceptible to various diseases (e.g., bacterial spot, early blight), which significantly impact agricultural productivity. Manual diagnosis is time-consuming and error-prone. ToMate addresses the need for an automated, scalable solution to detect diseases accurately and efficiently, reducing crop losses and supporting sustainable farming.

Objectives

ToMate aims to develop a deep learning model to accurately detect diseases in tomato leaves using image data. The goal is to empower farmers and gardeners with an accessible tool for early diagnosis, enabling timely intervention to improve crop health and yield. The model will be integrated into a web application for user-friendly disease detection, with future to enhance real-time prediction using bounding box techniques.

Target Users

ToMate is designed for a wide range of users in the agricultural sector. Farmers, both small-scale and commercial, can use it to monitor crop health and detect diseases early, preventing major losses. Agricultural researchers and agritech companies can benefit from ToMate by integrating it into precision farming solutions, improving data-driven decision-making. Additionally, government agencies and agricultural organizations focused on food security can utilize ToMate to support farmers with advanced disease detection tools. By providing a fast, accurate, and scalable solution, ToMate serves as an asset for anyone involved in tomato cultivation and plant disease management.

Business Impact

ToMate has a significant real-world impact on the agricultural industry by improving efficiency, reducing costs, and enhancing crop management. By automating disease detection, it minimizes the need for manual inspection, saving farmers time and labor costs. Early detection helps prevent the spread of diseases, reducing crop losses and increasing overall yield. This also leads to optimized pesticide usage, lowering expenses and promoting sustainable farming practices. Additionally, ToMate's real-time analysis improves decision-making, allowing farmers to take immediate action and protect their crops. With its scalability, ToMate can be integrated into large farming operations, agri-tech solutions, and government initiatives, making agriculture more productive and profitable.

Project Workflow

The project follows a systematic process, starting with data collection, which involves gathering a diverse set of images representing different tomato diseases and healthy plants. Data preprocessing ensures the dataset is clean and augmented, preparing it for model training. A convolutional neural network (CNN) is trained on this dataset to learn distinguishing features for each class. The model's performance is evaluated using metrics like AUC, precision, recall, and F1-score, and ROC curves visualize the trade-offs between sensitivity and specificity. Finally, the trained model is deployed for practical use, potentially in an app that assists farmers with plant disease diagnosis.

2. Features and Requirements

Core Features

ToMate provides an interactive web application where users can easily upload images of tomato leaves to check for diseases. The platform is designed for simplicity, allowing farmers and agricultural professionals to detect plant diseases quickly. In addition to image uploads, ToMate also offers a real-time scanning feature that enables users to scan their tomato leaves directly using a camera. This ensures faster and more convenient disease detection without needing any specialized equipment.

The web application is built with a user-friendly interface to ensure accessibility for all users, regardless of technical expertise. Once an image is uploaded or scanned, the system processes it using a deep learning model and provides instant results. This feature helps farmers take timely action to protect their crops, preventing the spread of diseases and improving overall yield. ToMate's interactive and easy-to-use approach makes disease detection more accessible and efficient.

Technical Requirements

ToMate relies on several technical components to ensure smooth functionality. The model was trained using Google Colab Premium, utilizing a 35GB GPU to speed up the training process. The trained model is then stored on a local machine for deployment. ToMate's backend is powered by Flask API, which allows seamless communication between the model and the web application. The user interface (UI) is developed using HTML and styled with CSS, ensuring a responsive and visually appealing experience for users.

The system is designed for performance, scalability, and reliability. It runs on local servers, making it accessible even in areas with limited internet connectivity. The software dependencies include TensorFlow and Keras, which are essential for deep learning-based image analysis. The infrastructure ensures that ToMate can handle multiple image requests efficiently while maintaining high accuracy in disease

detection. These technical choices make ToMate a powerful and reliable tool for plant disease identification.

3. Data Collection and Preprocessing

Data Collection

For the "Tomato Leaf Disease Prediction" project, the primary data source is a public dataset specifically curated for plant disease detection. The dataset used is the PlantVillage Dataset, which is obtained from kaggle. This dataset contains thousands of labeled images of healthy and diseased tomato leaves, covering various diseases such as Early Blight, Late Blight, Leaf Mold, Septoria Leaf Spot, Spider Mites, Target Spot, Yellow Leaf Curl Virus, Bacterial Spot, Mosaic Virus including the healthy tomato leaf images as well. The dataset is available in the form of high-resolution images, which are categorized into different folders based on the disease type. This makes it easier to organize and preprocess the data for training deep learning models.

Data Preprocessing

Before training the deep learning model, the dataset underwent extensive preprocessing to improve performance and generalization. The dataset is organized into folders, each representing a specific disease class. To ensure the model is trained only on relevant data, only tomato-specific folders were extracted and copied to a new directory named tomato_data. This step ensures that the model learns exclusively from tomato leaf diseases without being influenced by irrelevant images.

To train and evaluate the model effectively, the dataset was split into three subsets: training (80%), validation (10%), and test (10%). The training set was used for learning patterns, the validation set was utilized for tuning hyperparameters and preventing overfitting, and the test set provided an unbiased evaluation of the final model. This split ensures that the model performs well on unseen data and generalizes beyond the training samples.

Image preprocessing techniques were applied to enhance model stability and generalization:

- **Rescaling:** Pixel values were normalized to the range [0, 1] to ensure stable training and faster convergence.
- **Augmentation:** Various transformations such as random rotations, shifts, flips, and brightness adjustments were applied to increase dataset diversity. This step helps reduce overfitting by simulating real-world variations in images.
- **Fill Mode:** Applied to prevent blank spaces from appearing during transformations, ensuring the integrity of images remains intact.

To handle class imbalance, class weights were computed dynamically. If a particular disease class had significantly fewer samples compared to others, a higher weight was assigned to it during training. This adjustment ensures the model does not become biased toward majority classes and can effectively recognize underrepresented disease types.

Transfer learning was further optimized using Gradual Unfreezing, a technique that fine-tunes a pretrained model by progressively unfreezing its layers. The Gradual Unfreezing callback was implemented to unfreeze 10 layers of the base model every 5 epochs (default: num_epochs_per_unfreeze=5). Initially, the pretrained model's layers remained frozen to retain high-level features, and as training progressed, deeper layers were gradually fine-tuned. This approach prevents overfitting, ensures stable training, and avoids catastrophic forgetting, making the model more adaptable to tomato leaf disease classification.

By integrating these preprocessing steps, the model was trained on high-quality, well-balanced data, ensuring robust and accurate disease classification.

4. Model Architecture, Training and Evaluation

Model Architecture

Base Model Initialization

The deep learning model for tomato leaf disease prediction is built upon ResNet50V2, a powerful convolutional neural network (CNN) pretrained on the ImageNet dataset. ResNet50V2 is known for its deep residual learning framework, which allows efficient gradient propagation through skip connections. To tailor the model for tomato disease classification, the default classification layer of ResNet50V2 is removed, enabling the addition of custom layers. The input shape is set to 224×224 pixels with 3 color channels (RGB), ensuring compatibility with the tomato leaf images used for training. By leveraging the pretrained ImageNet weights, the model starts with robust feature extraction capabilities rather than learning from scratch, significantly improving training efficiency.

Freezing Layers

At the beginning of training, all layers of the ResNet50V2 base model are frozen to retain the knowledge acquired during its pretraining on ImageNet. This prevents unnecessary modifications to low-level feature representations, such as edges, textures, and simple patterns, which are often useful across different datasets. By freezing these layers initially, the model focuses on learning dataset-specific features in the custom layers added on top of the base model. Later in the training process, selected layers are progressively unfrozen to allow fine-tuning, ensuring the model adapts to tomato disease classification while preserving valuable pretrained knowledge.

Custom Layers

Following feature extraction by the base model, additional custom layers are introduced to refine and enhance the extracted features. A Global Average Pooling layer is applied to reduce the spatial dimensions of the feature maps, converting them into a compact vector representation while retaining essential information. This is followed by a

Dropout layer (20%), which randomly deactivates neurons during training to prevent overfitting and enhance generalization.

To further improve feature learning, the model incorporates two residual blocks. Each residual block consists of Dense layers with ReLU activation, ensuring non-linearity and efficient learning of complex patterns. Batch Normalization is applied after the dense layers to stabilize the training process by normalizing activations. Additionally, Dropout layers (ranging from 10% to 15%) are included to further mitigate overfitting. A key feature of these blocks is the skip connection (Add layer), which allows the input to bypass intermediate layers and be added directly to the output. This technique prevents the vanishing gradient problem, improves gradient flow, and ensures stable training, particularly in deep networks.

Output Layer

After processing through the custom layers, the final Dense layer uses softmax activation to generate a probability distribution over the 10 tomato disease classes. Each output neuron corresponds to a specific disease class, and the softmax function ensures that the sum of all probabilities equals 1, enabling clear classification decisions. This architecture ensures that the model efficiently maps extracted features to disease categories.

```
In [ ]: model.summary()
```

Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 224, 224, 3)	0	-
resnet50v2 (Functional)	(None, 7, 7, 2048)	23,564,800	input_layer_1[0][0]
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0	resnet50v2[0][0]
dropout (Dropout)	(None, 2048)	0	global_average_poolin…
dense (Dense)	(None, 512)	1,049,088	dropout[0][0]
batch_normalization (BatchNormalization)	(None, 512)	2,048	dense[0][0]
dense_1 (Dense)	(None, 512)	1,049,088	dropout[0][0]
dropout_1 (Dropout)	(None, 512)	0	batch_normalization[0…
add (Add)	(None, 512)	0	dense_1[0][0], dropout_1[0][0]
activation (Activation)	(None, 512)	0	add[0][0]
dense_2 (Dense)	(None, 256)	131,328	activation[0][0]
dense_3 (Dense)	(None, 256)	131,328	activation[0][0]
dropout_2 (Dropout)	(None, 256)	0	dense_2[0][0]
add_1 (Add)	(None, 256)	0	dense_3[0][0], dropout_2[0][0]
activation_1 (Activation)	(None, 256)	0	add_1[0][0]
dense_4 (Dense)	(None, 10)	2,570	activation_1[0][0]

Total params: 25,930,250 (98.92 MB)
Trainable params: 2,364,426 (9.02 MB)
Non-trainable params: 23,565,824 (89.90 MB)

Model Summary

Training and Optimization

Model Compilation

Once the model architecture is defined, it is compiled for training using Adam W as the optimizer. Adam W, an improved version of Adam, integrates weight decay regularization, which helps prevent overfitting by penalizing large weight updates. The learning rate is set to 1e-4, ensuring stable convergence. To optimize classification performance, the loss function used is Categorical Cross entropy with Label Smoothing (0.1). Label smoothing reduces the model's confidence in predictions, preventing overfitting and improving generalization. The model is evaluated using multiple performance metrics, including Accuracy, AUC (Area Under the Curve), Precision, and Recall, to provide a comprehensive assessment of its classification effectiveness.

Learning Rate Scheduling

A warmup cosine decay learning rate schedule is implemented to optimize the training process. During the warmup phase (first five epochs), the learning rate gradually increases from 0 to 1e-4, ensuring stable training and preventing large gradient updates that could destabilize the model. After the warmup phase, the cosine decay schedule reduces the learning rate smoothly from 1e-4 to 0 over the remaining epochs. This approach enables controlled learning at the start while ensuring fine-tuned adjustments toward the end of training, leading to better convergence and improved model performance.

Callbacks Setup

To enhance and monitor training, several callbacks are incorporated. The Early Stopping callback monitors validation loss and stops training if no improvement is observed for 10 consecutive epochs, restoring the best-performing model weights to prevent overfitting. The Model Checkpoint callback saves the best model whenever validation accuracy improves, ensuring that the highest-performing model is retained for deployment. A Learning Rate Scheduler callback dynamically adjusts the learning rate according to the warmup cosine decay schedule, further optimizing convergence.

One of the key strategies used in this model is Gradual Unfreezing, where layers of the ResNet50V2 base model are progressively unfrozen during training. Every five epochs, ten additional layers are unfrozen, allowing the model to fine-tune deeper layers without disrupting previously learned features. This method ensures a balance between leveraging pretrained knowledge and adapting to the specific task of tomato disease classification.

To prevent training interruptions, the Backup and Restore callback saves progress periodically, allowing training to resume from the last saved state in case of unexpected failures. Finally, Tensor Board logging is enabled, providing real-time visualization of training metrics such as loss, accuracy, and learning rate progression. This facilitates better monitoring and debugging of the training process.

5. Model Evaluation

The deep learning model demonstrates strong performance in detecting tomato leaf diseases, achieving high accuracy and robust classification metrics. The evaluation was conducted on a test set of 1,606 images, covering ten different disease categories, including bacterial spot, early blight, late blight, and healthy tomato leaves. The results indicate that the model is highly reliable and capable of distinguishing between disease classes with high precision, recall, and AUC scores.

Test Performance Metrics

The model achieves a test accuracy of 93.09%, meaning that it correctly classifies 93.09% of the test images, confirming its effectiveness in disease detection. The test loss of 0.7588 is relatively low, indicating that the model's predictions closely match the actual labels. Additionally, the test AUC score of 0.9969 suggests excellent class separation, as a value close to 1.0 means the model is highly confident in its classifications. This ensures that the model is capable of differentiating between healthy and diseased leaves with minimal errors.

Precision and recall metrics further validate the model's effectiveness. The test precision of 0.9581 indicates that the model produces very few false positives, ensuring that when it predicts a diseased leaf, it is highly likely to be correct. Meanwhile, the test recall of 0.8823 suggests that the model successfully identifies most diseased cases, reducing the chances of missing infected plants. This is particularly important for real-world agricultural applications were failing to detect diseases could lead to crop losses.

Validation Performance

The model's validation accuracy of 91.63% and validation loss of 0.7818 show that it generalizes well to unseen data without overfitting. The validation loss being close to the training loss suggests that the model maintains a balanced learning process and does not memorize training data, further confirming its reliability.

Class-wise Performance

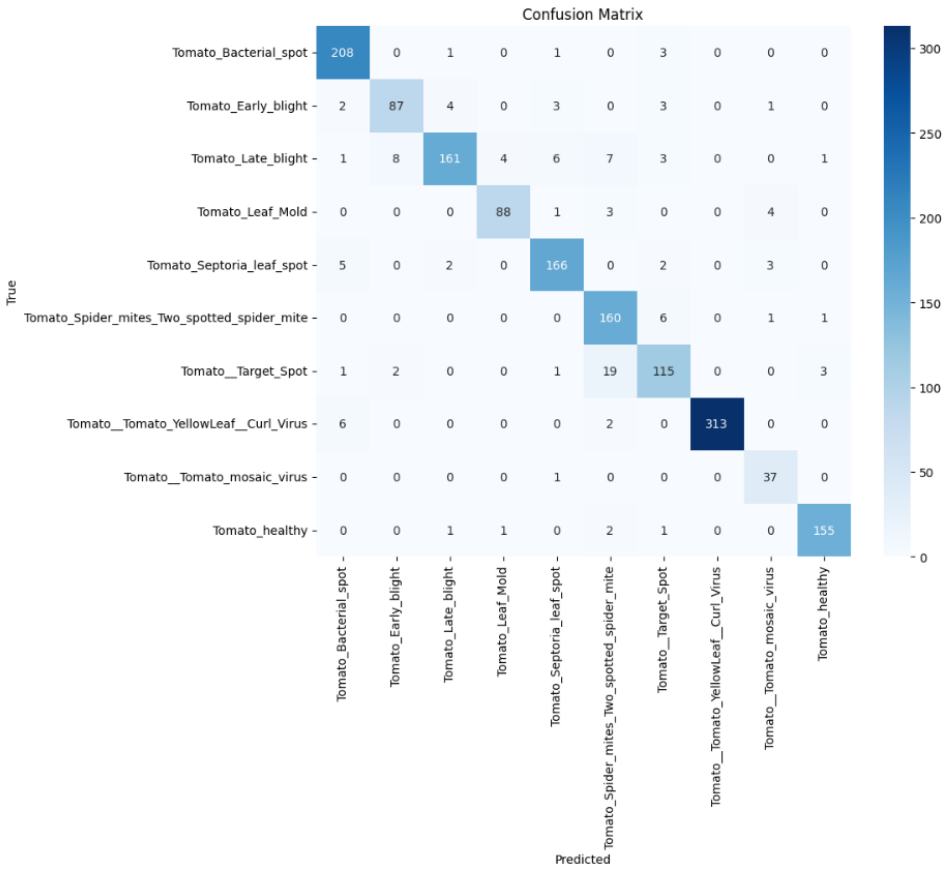
An in-depth classification report highlights the model’s per-class precision, recall, and F1-score. It excels in detecting diseases such as Tomato Yellow Leaf Curl Virus, with 100% precision and 98% recall, ensuring that nearly all cases of this disease are accurately identified. Similarly, Tomato Bacterial Spot achieves a 95% F1-score, demonstrating its ability to correctly classify this disease with minimal errors. The Tomato Mosaic Virus class, despite having a smaller number of samples, achieves 97% recall, showing the model's ability to recognize even rare disease instances.

Some disease classes, such as Tomato Target Spot, have slightly lower recall values (82%), indicating that some images in this category were misclassified. However, the overall performance across all classes remains consistent, with an average macro F1-score of 92%, confirming the model’s strong classification ability across different types of tomato leaf diseases.

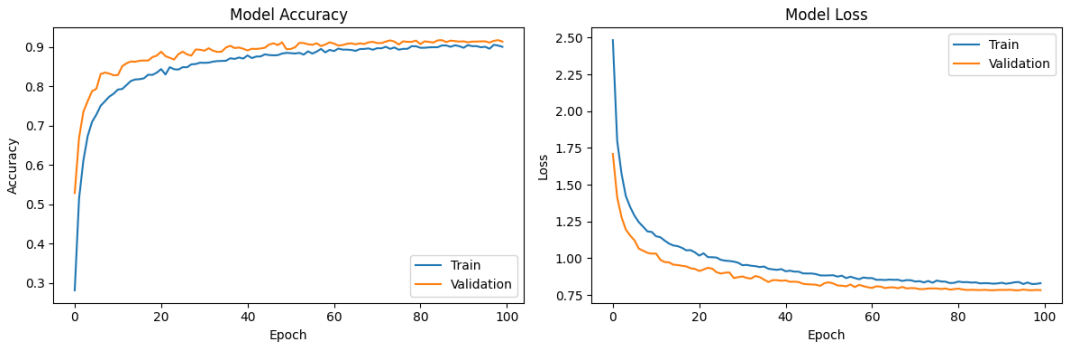
	precision	recall	f1-score	support
Tomato_Bacterial_spot	0.93	0.98	0.95	213
Tomato_Early_blight	0.90	0.87	0.88	100
Tomato_Late_blight	0.95	0.84	0.89	191
Tomato_Leaf_Mold	0.95	0.92	0.93	96
Tomato_Septoria_leaf_spot	0.93	0.93	0.93	178
Tomato_Spider_mites_Two_spotted_spider_mite	0.83	0.95	0.89	168
Tomato__Target_Spot	0.86	0.82	0.84	141
Tomato__Tomato_YellowLeaf__Curl_Virus	1.00	0.98	0.99	321
Tomato__Tomato_mosaic_virus	0.80	0.97	0.88	38
Tomato_healthy	0.97	0.97	0.97	160
accuracy			0.93	1606
macro avg	0.91	0.92	0.92	1606
weighted avg	0.93	0.93	0.93	1606

Classification Report

The classification report shows that the model performs exceptionally well across most tomato disease classes, achieving 93% accuracy with high precision, recall, and F1-scores. It correctly identifies diseases like Tomato_Bacterial_spot (F1: 0.95) and TomatoTomato_YellowLeafCurl_Virus (F1: 0.99) with near-perfect accuracy. Some classes, like TomatoTarget_Spot (F1: 0.84) and TomatoTomato_mosaic_virus (F1: 0.88), have slightly lower performance, suggesting room for improvement. The model maintains a good balance between precision and recall, making it reliable for real-world tomato disease detection.



The confusion matrix illustrates the model's performance in classifying different tomato leaf diseases. The diagonal values represent the correctly predicted instances for each class, showing strong performance, especially for Tomato Yellow Leaf Curl Virus (313 correct predictions) and Tomato Bacterial spot (208 correct predictions). Most misclassifications involve similar disease types, such as Tomato Late blight, which has some overlap with Tomato Early blight and Tomato Septoria leaf spot. Some minor confusion occurs in Tomato mosaic virus (37 correct predictions) and Tomato Target Spot (115 correct predictions), indicating potential challenges in differentiating these classes. Overall, the model exhibits high accuracy with minimal misclassification, confirming its strong performance in tomato disease detection.



The training plots show the model's accuracy and loss over 100 epochs. The accuracy curve indicates steady improvement, with both training and validation accuracy reaching over 90%, suggesting strong generalization. The loss curve shows a sharp decline initially, then stabilizes, with validation loss slightly lower than training loss. There are no major gaps between training and validation curves, meaning the model is not overfitting. If overfitting were present, the validation loss would increase while training loss continued decreasing. Similarly, no underfitting is observed, as both accuracies are high, and loss is consistently reducing. The model appears well-trained, with good generalization and minimal overfitting concerns.

6. Conclusion

Tomato crops are vulnerable to various diseases that can severely impact yield and agricultural productivity. Early and accurate detection is essential for farmers to take preventive measures, but manual identification is often slow, error-prone, and requires expert knowledge. ToMate addresses this challenge by providing an AI-powered, automated tomato leaf disease detection system that leverages deep learning to classify multiple diseases efficiently.

Built on a ResNet50V2-based architecture with residual connections, dropout regularization, and learning rate scheduling, ToMate extracts meaningful patterns from leaf images and delivers highly accurate predictions with minimal intervention. By integrating transfer learning and fine-tuning strategies, the system ensures scalability and adaptability for real-world agricultural use. Designed for deployment in farms, greenhouses, and monitoring systems, ToMate empowers farmers with a fast, reliable, and automated solution, promoting sustainable farming practices and reducing crop losses.

References

Agarwal, M. (2020). ToLeD: Tomato Leaf Disease Detection using Convolution Neural Network. *Science Direct*, 9.

Brahimi M., A. M. (2018). Deep learning for plant diseases: Detection and saliency map visualisation. *Human and Machine Learning, Springer*.

Brahimi M., B. K. (2017). Deep learning for tomato diseases: classification and symptoms visualization. *Applied Artificial Intelligence*, 299 - 315.

Emmanuel, T. O. (2019). *Kaggle*. Retrieved from kaggle.com: https://www.kaggle.com/datasets/emmarex/plantdisease/data

Kawasaki Y., U. H. (2015). Basic study of automated diagnosis of viral plant diseases using convolutional neural networks. *International Symposium on Visual Computing, Springer* , 938 - 645.

Sharma, R. (2022). Tomato Leaf Disease Detection Using Machine Learning. *CEUR-WS*, 6.