CNN-Based Tomato Leaf Disease Detection Using Deep Learning

Daniel Matias Suryadi Putra

1. Summary

This project focuses on developing a practical tool for identifying diseases in tomato leaves using computer vision techniques. The system is capable of recognizing common tomato leaf diseases such as Early Blight, Late Blight, and Leaf Mold, as well as differentiating them from healthy leaves. The tool utilizes a comprehensive dataset of labeled tomato leaf images to train a model that can detect patterns and features indicative of each disease. The model is trained over several epochs to ensure it learns the necessary patterns effectively, optimizing its accuracy and reliability.

The model is integrated into a web application built with React, providing an intuitive interface where users can upload images of tomato leaves for analysis. The backend of the application is powered by FastAPI, which processes the uploaded images and returns predictions on the disease type, along with a confidence percentage indicating the model's certainty. This setup allows users to quickly assess the health of their crops and make informed decisions about disease management.

The entire system is deployed on Google Cloud Platform (GCP), ensuring scalability and efficient handling of multiple user requests. By providing a fast and reliable method for disease detection, this tool aims to assist farmers in improving crop management and reducing losses due to disease, ultimately benefiting the agricultural sector by enhancing productivity and sustainability.

2. Introduction

In the agricultural sector, early detection and management of crop diseases are crucial for ensuring high yields and minimizing losses. This project aims to address the challenge of detecting tomato leaf diseases using advanced computer vision techniques. We focus on identifying diseases such as Early Blight, Late Blight, and Leaf Mold, as well as recognizing healthy leaves. These diseases, if not detected and managed promptly, can lead to significant economic losses for farmers and impact food supply chains.

The system leverages a Convolutional Neural Network (CNN) model trained on a diverse dataset of labeled tomato leaf images. The model is designed to learn intricate patterns and features associated with each

disease type through multiple training epochs. This allows the system to provide quick and precise disease identification.

The project integrates the CNN model into a user-friendly web application built with React. This application allows users to easily upload images of tomato leaves for analysis. The backend, powered by FastAPI, efficiently processes the images and returns predictions along with a confidence percentage.

Deployment on Google Cloud Platform (GCP) ensures that the system is scalable and capable of handling numerous requests simultaneously, making it accessible to users worldwide. By automating the disease detection process, this tool provides a valuable resource for farmers and agricultural professionals, enabling them to make informed decisions and improve crop management practices. The combination of computer vision, web technologies, and cloud deployment highlights the potential of modern technology to revolutionize agricultural practices and contribute to sustainable food production.

3. Data Preparation

Effective data preparation is crucial for developing a reliable machine learning model, especially in the domain of computer vision. For this project, we utilized a comprehensive dataset of tomato leaf images, which included images categorized into four classes: Early Blight, Late Blight, Leaf Mold, and Healthy. The dataset was sourced from publicly available agricultural image repositories and datasets.

Data Collection

- **Dataset Composition**: The dataset consists of around 3 thousand images captured under varying lighting conditions and angles to simulate real-world scenarios. This diversity helps the model generalize well across different environments and conditions.

Data Preprocessing

1. **Image Resizing and Normalization**:
    - All images were resized to a standard dimension (e.g., 256x256 pixels) to ensure uniformity and reduce computational overhead during training.
    - Pixel values were normalized to a range of [0, 1] to enhance the model's ability to learn from the data by ensuring that all features contribute equally to the learning process.

2. **Data Augmentation**:
    - To improve the model's generalization capabilities, data augmentation techniques were applied. This included random horizontal and vertical flips and random rotations up to 20 degrees. Data augmentation increases the dataset's diversity and helps prevent overfitting by exposing the model to a wider range of potential variations.
3. **Dataset Splitting**:
    - The dataset was split into training, validation, and test sets using an 80-10-10 ratio. This split ensures that the model is trained on a substantial portion of the data while also being evaluated on unseen images to assess its performance and generalization capabilities.
    - TensorFlow's image_dataset_from_directory function was used to efficiently load and split the dataset, leveraging its built-in support for data augmentation and batching.
4. **Caching and Prefetching**:
    - To optimize training performance, caching and prefetching strategies were applied to the dataset. Caching allows the dataset to be stored in memory, reducing data loading times during training epochs. Prefetching overlaps data preprocessing and model execution, further speeding up the training process.

By carefully preparing and augmenting the dataset, the model is better equipped to learn the distinguishing features of each disease category, ultimately leading to more accurate and robust predictions. This comprehensive data preparation pipeline forms the foundation for the subsequent model training and deployment phases.

4. Model Architecture

The model for classifying tomato leaf diseases is built using a Convolutional Neural Network (CNN), which is effective for image classification tasks. The architecture consists of several layers designed to extract and learn features from the images:

1. Input Layer:

- The model accepts input images of size 256x256 pixels with three color channels (RGB). This standardized input size helps the model process images consistently.

2. **Convolutional Layers:**

- The model consists of multiple convolutional layers that apply filters to the input images. These filters detect features such as edges, textures, and patterns that are crucial for identifying diseases.
- ReLU Activation: The Rectified Linear Unit (ReLU) activation function is used in these layers. ReLU is chosen because it introduces non-linearity to the model and helps prevent issues like the vanishing gradient problem, allowing the model to learn complex patterns more effectively.

3. **Max Pooling Layers:**

- Max pooling layers follow each convolutional layer to reduce the spatial dimensions of the feature maps. This operation retains the most important features while reducing computational complexity, making the model more efficient.

4. **Flatten Layer:**

- The flatten layer converts the 2D feature maps from the convolutional layers into a 1D vector. This transformation allows the output to be fed into the dense layers for classification.

5. **Dense Layers:**

- The dense layers are fully connected layers that aggregate features extracted by the convolutional layers. They help the model learn relationships between features and make predictions.
- The first dense layer uses ReLU activation to introduce non-linearity and capture complex patterns.

6. **Output Layer:**

- The final dense layer has four units, corresponding to the four classes: Early Blight, Late Blight, Leaf Mold, and Healthy.
- Softmax Activation: The softmax activation function is used in the output layer to convert the output into a probability distribution over the four classes. Softmax ensures that the sum of the output probabilities equals 1, making it suitable for multi-class classification problems. It helps the model output a confidence score for each class, indicating how likely it is that a given image belongs to each category.

The architecture is designed to efficiently learn from the dataset and accurately classify tomato leaf images into their respective categories

5. Training Process

The training process involves teaching the Convolutional Neural Network (CNN) model to accurately classify tomato leaf images into their respective disease categories. The following steps outline how the training was conducted:

1. **Model Compilation**:
    - The model is compiled using the **Adam optimizer**, known for its efficiency and effectiveness in training deep learning models. Adam adapts the learning rate during training, which helps in converging quickly to an optimal solution.
    - The loss function used is **sparse_categorical_crossentropy**, which is appropriate for multi-class classification tasks with integer labels. This function measures the discrepancy between the predicted class probabilities and the true class labels.
    - **Accuracy** is used as the evaluation metric to monitor the model's performance during training and validation.
2. **Training the Model**:

- The model is trained over a specified number of epochs, allowing it to learn from the training dataset. During each epoch, the model processes the entire training dataset and updates its weights to minimize the loss function.
- The training dataset is augmented with techniques such as random flips and rotations to enhance the model's ability to generalize to unseen data.

3. **Validation**:

- A portion of the dataset is set aside for validation to evaluate the model's performance on unseen data. This helps in monitoring overfitting, where the model performs well on training data but poorly on new data.
- Validation metrics, including accuracy and loss, are calculated at the end of each epoch to provide insights into the model's progress and to guide adjustments if necessary.

4. **Learning Curves**:

- The training and validation accuracy and loss are plotted over epochs to visualize the model's learning process. These learning curves help identify trends, such as overfitting or underfitting, and assess whether the model is improving with additional training.

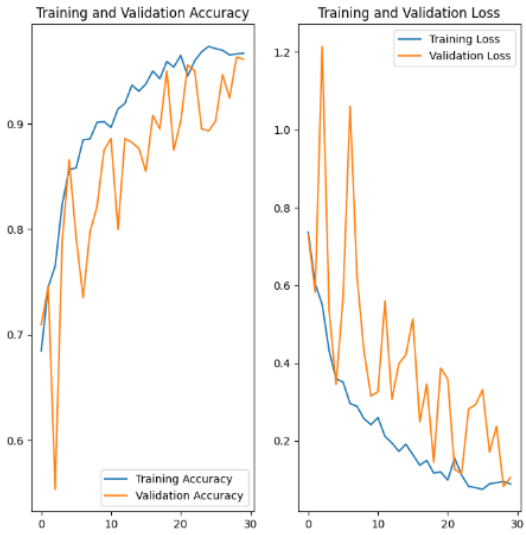
6. Results

6.1 Accuracy and Loss Metrics

The Convolutional Neural Network (CNN) model was evaluated over 30 training epochs, with the following key performance metrics:

- **Final Training Accuracy**: 97.65%
- **Final Training Loss**: 0.0718

These metrics indicate that the model learned the training data well, achieving high accuracy and low loss, which is crucial for effective classification.



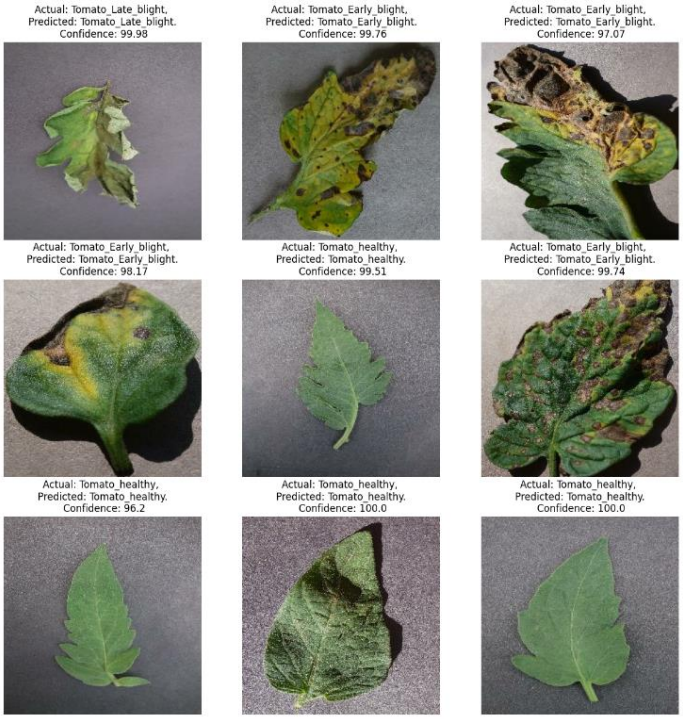
The learning curves for both accuracy and loss are shown in the Figure above:

- **Training Accuracy and Loss**: Throughout the training process, the model consistently improved, with accuracy increasing and loss decreasing. This trend indicates effective learning and adaptation to the data.
- **Validation Accuracy and Loss**: Validation metrics mirrored the training performance, with validation accuracy closely tracking training accuracy. The consistent decrease in validation loss suggests that the model generalizes well to new data.

6.2 Prediction Results

Sample predictions made by the model on the validation set are shown in the Figure below. These results highlight the model's effectiveness:

- **Correctly Classified Examples**: The model successfully identified various diseases, such as Early Blight and healthy leaves, with confidence scores often exceeding 96%. This level of accuracy and confidence underscores the model's reliability for practical applications.
- **Confidence Scores**: The model's predictions consistently demonstrated high confidence, reinforcing its robustness in real-world scenarios.



6.3 Training Configuration

The model was trained with the following configuration:

- **Epochs**: 30
- **Steps per Epoch**: 136, based on dividing the total number of images by the batch size. This calculation ensures that each image is processed once per epoch, optimizing the training process.
- **Verbose Mode**: Auto

This configuration was chosen to optimize learning while ensuring efficient use of computational resources

7. Web application integration

To make the tomato leaf disease classification model accessible and user-friendly, we developed a web application using React for the frontend and FastAPI for the backend.

Frontend: React

The frontend is built with React, offering a simple and responsive interface. Users can upload images of tomato leaves directly on the website. Once uploaded, the application displays the predicted disease category and confidence level for each image, allowing users to quickly assess the health of their crops.

Backend: FastAPI

The backend is powered by FastAPI, which efficiently handles requests and processes images through the trained CNN model. FastAPI is chosen for its speed and capability to manage real-time data processing. When an image is uploaded, the backend preprocesses it, runs it through the model, and sends back the prediction results to the frontend.

TensorFlow Serving is used to deploy the trained CNN model. It offers efficient inference with low latency, making it ideal for real-time applications where quick prediction responses are critical. Additionally, TensorFlow Serving supports model versioning, allowing for seamless updates and management of different model versions as the application evolves.

Deployment on Google Cloud Platform

The application is deployed on Google Cloud Platform (GCP), ensuring it is scalable and always accessible. GCP provides the necessary infrastructure to handle multiple users and requests, making the tool reliable and efficient for farmers and agricultural professionals.

8. Conclusion

In this project, we demonstrated the effectiveness of using a Convolutional Neural Network (CNN) for classifying tomato leaf diseases. The model achieved high accuracy and reliability in distinguishing between Early Blight, Late Blight, Leaf Mold, and healthy leaves. The CNN's performance, as indicated by the training and validation results, shows minimal signs of overfitting, with the model maintaining consistent accuracy across different datasets.

While CNN proved to be a powerful tool for this classification task, it is important to consider multiple algorithms and evaluate their performance on specific datasets. Different models may offer unique advantages depending on the characteristics of the data. Therefore, exploring other machine learning techniques or combining models through ensemble methods could further enhance the system's accuracy and robustness.

For future work, it would be beneficial to expand the model to cover additional plant diseases and optimize its efficiency for faster processing. Investigating the features that contribute most significantly to accurate predictions could also provide insights into improving the model further. This project highlights the potential of leveraging modern AI and web technologies to tackle real-world challenges in agriculture