A Novel Image Segmentation Technique for Improving Plant Disease Classification with Deep Learning Models

ABSTRACT

The Image Segmentation for Plant Disease project focuses on developing a deep learning-based solution to segment leaf regions in images for identifying plant diseases. Utilizing the Leaf Disease Segmentation Dataset from Kaggle (https://www.kaggle.com/datasets/fakhrealam9537/leaf-disease-segmentation-dataset), the project implements three segmentation models: UNet++, DeepLabV3, both with ResNet34 encoders pretrained on ImageNet, and a custom Swin Transformer-based model with a convolutional decoder. The models are trained to generate binary masks, identifying leaf pixels labeled as 38 in grayscale masks, using data preprocessing with Albumentations for resizing, normalization, and augmentation. Training involves 10 epochs with Adam optimizer and Dice Loss (UNet++, DeepLabV3) or BCEWithLogitsLoss (Swin Transformer). Performance is evaluated using Dice Score and IoU metrics, with UNet++ achieving the highest scores (Dice: 0.8401, IoU: 0.7423), followed by DeepLabV3 (Dice: 0.7693, IoU: 0.6603), and Swin Transformer (Dice: 0.2461, IoU: 0.1554). The UNet++ model is deployed in a Flask web application, enabling user registration, login, and image upload for segmentation predictions, supported by a MySQL database for user management. This project demonstrates an effective approach to plant disease segmentation with practical deployment for real-world use.

Motivation:

Plant diseases significantly impact agricultural productivity, leading to reduced crop yields and economic losses. Early and accurate detection of diseased leaf regions is crucial for timely

intervention and disease management. Manual inspection of plants is labor-intensive and prone to errors, making automated solutions essential. Advances in deep learning, particularly in image segmentation, offer promising techniques for identifying affected leaf areas with high precision. The availability of the Leaf Disease Segmentation Dataset on Kaggle provides a valuable resource to develop and test such models. This project is motivated by the need to leverage deep learning to automate plant disease detection, enabling farmers and researchers to diagnose and address plant health issues efficiently.

Problem Statement:

The identification of diseased leaf regions in plant images is a challenging task due to variations in leaf appearance, lighting conditions, and disease patterns. Traditional methods rely on manual observation, which is time-consuming and subjective. The Leaf Disease Segmentation Dataset provides images and corresponding masks where leaf pixels are labeled as 38 in grayscale, requiring a robust segmentation model to generate accurate binary masks. The challenge is to develop deep learning models that can effectively segment leaf regions, compare their performance, and deploy a practical solution for end users to upload images and obtain segmentation results.

Objective of the Project:

Develop a deep learning-based system to segment leaf regions in images for plant disease detection using UNet++, DeepLabV3, and Swin Transformer, deployed via a Flask web application.

Scope:

The project focuses on the following components:

- Dataset: Utilize the Leaf Disease Segmentation Dataset from Kaggle, consisting of leaf images and corresponding masks with leaf pixels labeled as 38.
- Models: Implement and compare three segmentation models: UNet++, DeepLabV3 (both using ResNet34 encoders pretrained on ImageNet), and a custom Swin Transformer-based model with a convolutional decoder.
- Training: Train models for 10 epochs using the Adam optimizer, with Dice Loss for UNet++ and DeepLabV3, and BCEWithLogitsLoss for Swin Transformer.
- Evaluation: Assess model performance on the training dataset using Dice Score and IoU metrics.
- Deployment: Develop a Flask web application with routes for user registration, login, and image upload for segmentation using the UNet++ model, integrated with a MySQL database for user data storage.
- Limitations: The project evaluates models on the training set without a separate validation or test split and uses limited data augmentation (horizontal flip).

Project Introduction:

The Image Segmentation for Plant Disease project aims to develop a deep learning-based system to segment leaf regions in images for detecting plant diseases. The project leverages the Leaf Disease Segmentation Dataset from Kaggle

(https://www.kaggle.com/datasets/fakhrealam9537/leaf-disease-segmentation-dataset), which includes images of leaves and corresponding masks where leaf pixels are labeled as 38 in grayscale. Three segmentation models are implemented: UNet++ and DeepLabV3, both utilizing ResNet34 encoders pretrained on ImageNet, and a custom Swin Transformer-based model with a convolutional decoder. The models are trained to generate binary masks, with performance evaluated using Dice Score and IoU metrics. The UNet++ model, which achieved the highest performance (Dice: 0.8401, IoU: 0.7423), is deployed in a Flask web application. The application supports user registration, login, and image upload for segmentation predictions, with user data stored in a MySQL database. This project provides a practical solution for automated plant disease detection through image segmentation.

SYSTEM ANALYSIS

Existing System

The existing system for plant disease detection, as implied by the motivation of the Image Segmentation for Plant Disease project, relies on manual inspection or traditional image processing techniques to identify diseased leaf regions. Manual inspection involves agricultural experts visually examining plant leaves to detect disease symptoms, which is labor-intensive and subjective. Traditional image processing methods, such as thresholding or edge detection, may be used to segment leaf regions but struggle with variations in lighting, leaf texture, and disease patterns. These methods often require hand-crafted features and lack the robustness needed for complex scenes, such as those in the Leaf Disease Segmentation Dataset, where leaf pixels are labeled as 38 in grayscale masks. The project's focus on deep learning-based segmentation using

UNet++, DeepLabV3, and Swin Transformer suggests that existing systems are limited in accuracy and scalability, prompting the need for automated, data-driven approaches.

Disadvantages of Existing Methods

- Labor-Intensive: Manual inspection requires significant time and effort from agricultural experts, making it impractical for large-scale applications.
- Subjectivity: Human-based assessments are prone to errors and inconsistencies due to subjective interpretation of disease symptoms.
- Limited Robustness: Traditional image processing techniques, such as thresholding, fail to handle variations in lighting, leaf texture, and disease patterns.
- Hand-Crafted Features: Existing methods often rely on manually designed features, which are less adaptable to diverse datasets like the Leaf Disease Segmentation Dataset.
- Scalability Issues: Manual and traditional methods are not scalable for processing large volumes of images, limiting their use in modern agriculture.

Proposed Method

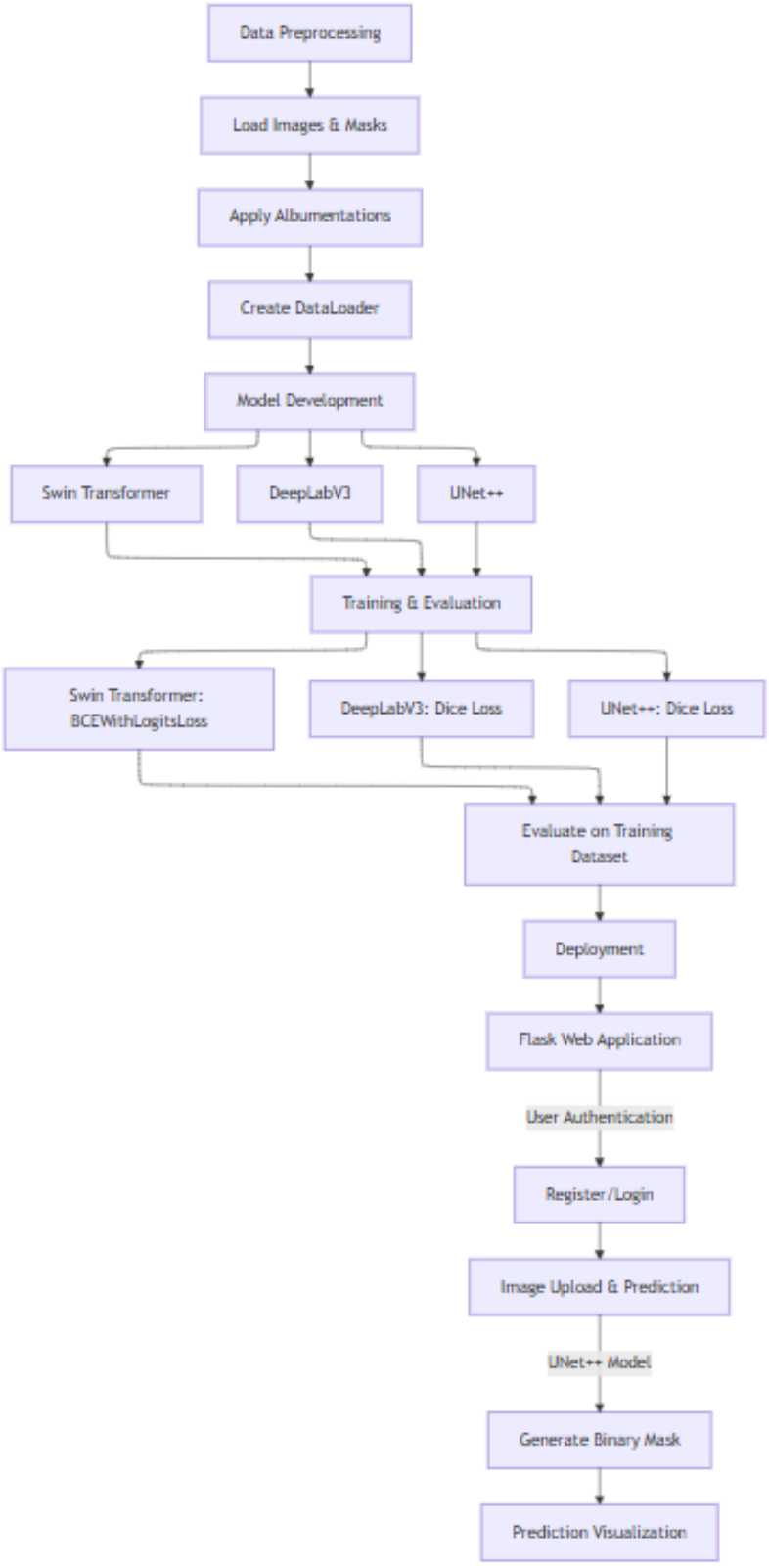
The proposed method in the Image Segmentation for Plant Disease project employs deep learning-based image segmentation to automate the detection of diseased leaf regions using the Leaf Disease Segmentation Dataset from Kaggle. Three models are implemented: UNet++ and DeepLabV3, both with ResNet34 encoders pretrained on ImageNet, and a custom Swin Transformer-based model with a convolutional decoder. The models are trained on leaf images and grayscale masks

(label 38 for leaf pixels, converted to binary) using Albumentations for preprocessing (resizing to 256x256, normalization, and horizontal flip augmentation). Training involves 10 epochs with the Adam optimizer, using Dice Loss for UNet++ and DeepLabV3, and BCEWithLogitsLoss for Swin Transformer. Performance is evaluated using Dice Score and IoU metrics, with UNet++ achieving the highest scores (Dice: 0.8401, IoU: 0.7423). The UNet++ model is deployed in a Flask web application, allowing users to register, log in, and upload images for segmentation, with results displayed as binary masks. A MySQL database manages user data (name, email, password), enabling secure user authentication.

Advantages of Proposed Method

- Automation: Deep learning models (UNet++, DeepLabV3, Swin Transformer) automate leaf segmentation, reducing the need for manual inspection.
- High Accuracy: UNet++ achieves a Dice Score of 0.8401 and IoU of 0.7423, indicating precise segmentation of leaf regions.
- Robust Preprocessing: Albumentations ensures consistent resizing, normalization, and augmentation, improving model performance on diverse images.
- Practical Deployment: The Flask web application enables user-friendly image uploads and segmentation, supported by MySQL for user management.
- Comparative Analysis: Evaluating three models allows selection of the best-performing model (UNet++) for deployment, ensuring optimal results.

Project flow diagram



REQUIREMENT ANALYSIS

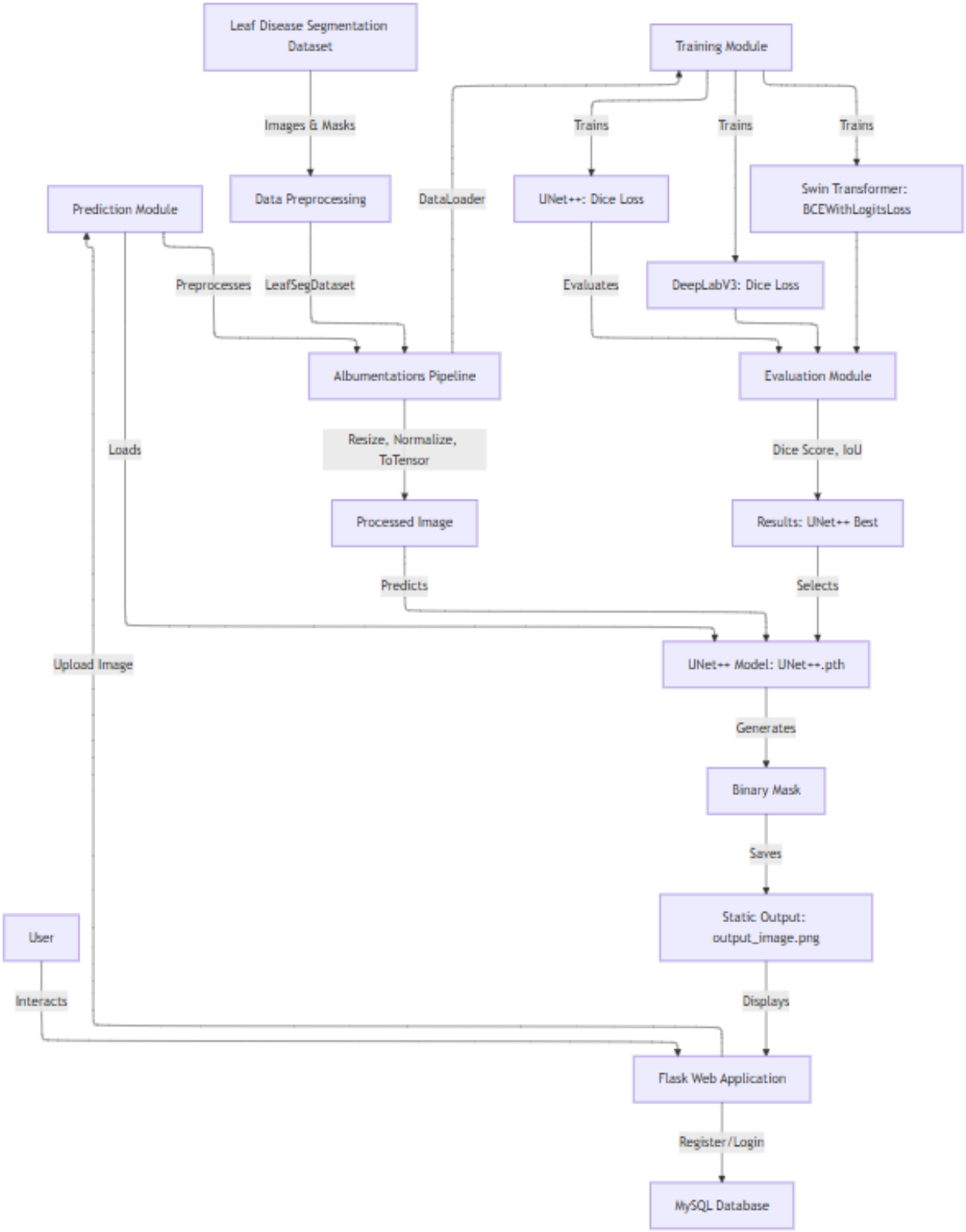
Hardware Requirements

Processor	- I7/Intel Processor
Hard Disk	- 160GB
Key Board	- Standard Windows Keyboard
Mouse	- Two or Three Button Mouse
Monitor	- SVGA
RAM	- 8GB

Software Requirements:

Operating System	: Windows 11
Server side Script	: HTML, CSS, Bootstrap & JS
Programming Language	: Python
Libraries	: Pandas, NumPy, tensorflow, , scikit-learn
IDE/Workbench	: VSCode
Technology	: Python 3.10.8
Server Deployment	: Xampp Server
Database	: MySQL

Architecture



MODULES

1) Data Preprocessing Module

This module handles the loading and preprocessing of the Leaf Disease Segmentation Dataset from Kaggle. It uses a custom LeafSegDataset class to load images and grayscale masks (label 38 for leaf pixels, converted to binary) from the dataset\images and dataset\masks directories. The Albumentations library is employed for preprocessing, which includes:

- Resizing images and masks to 256x256 pixels.
- Applying horizontal flip augmentation with a probability of 0.5.
- Normalizing pixel values (mean=0, std=1).
- Converting images and masks to PyTorch tensors using ToTensorV2. The dataset is loaded into a DataLoader with a batch size of 8 for UNet++ and DeepLabV3, and 4 for Swin Transformer, with shuffling enabled to ensure randomized batch processing during training.

2) Model Development Module

This module encompasses the implementation of three segmentation models: UNet++, DeepLabV3, and a custom Swin Transformer-based model.

- UNet++: Implemented using the segmentation_models_pytorch library with a ResNet34 encoder pretrained on ImageNet, configured for 3 input channels (RGB) and 1 output channel (binary mask) without activation.

- DeepLabV3: Also implemented using segmentation_models_pytorch with a ResNet34 encoder pretrained on ImageNet, configured similarly for 3 input channels and 1 output channel.
- Swin Transformer: A custom model using a swin_t backbone (pretrained on ImageNet) from torchvision, with features extracted from the last layer (features.7, 768 channels). A convolutional decoder (sequential layers: 768→384→128→64→32→1) with ReLU, batch normalization, and bilinear upsampling is used to produce a binary mask, interpolated to 256x256.

  Each model is designed to generate binary masks identifying leaf regions, leveraging pretrained weights for enhanced feature extraction.

3) Training and Evaluation Module

This module handles the training and evaluation of the three models. Training is conducted for 10 epochs on a CUDA device (if available, else CPU) using the Adam optimizer with a learning rate of 1e-4. UNet++ and DeepLabV3 use Dice Loss (binary mode), while Swin Transformer uses BCEWithLogitsLoss. The training loop involves:

- Forward pass to generate predictions.
- Loss computation between predictions and ground truth masks.
- Backpropagation and weight updates using the optimizer.
- Printing epoch losses for monitoring.

  Models are saved as UNet++.pth, DeepLabV3.pth, and SwinSegModel.pth. Evaluation is performed on the training dataset, computing Dice Score and IoU metrics. Predictions are thresholded at 0.5 after sigmoid activation, and metrics are calculated per image by

comparing predicted and ground truth masks. The evaluation loop operates in no-gradient mode for efficiency.

4) Web Application Deployment Module

This module implements a Flask web application for deploying the UNet++ model, enabling user interaction for segmentation predictions. The application includes:

- User Authentication: Routes for registration (/register) and login (/login), storing user data (name, email, password) in a MySQL database (plant database, users table). Registration checks for duplicate emails and password confirmation, while login verifies credentials.
- Prediction: A /prediction route allows users to upload images, which are saved in static/saved_images. The UNet++ model, loaded from UNet++.pth, processes the image with the same Albumentations preprocessing (resize, normalize, tensor conversion). The predicted binary mask is generated and saved as static/output_image.png for display.
- Database Management: MySQL handles user data storage with queries for inserting and retrieving user information, using parameterized queries to prevent SQL injection. The application uses session management to track user logins and renders HTML templates (index.html, about.html, register.html, login.html, home.html, prediction.html) for a user-friendly interface.

5) Prediction Visualization Module

This module facilitates the visualization of segmentation results for single images. A predict function processes an input image by applying the same Albumentations preprocessing (resize to 256x256, normalization, tensor conversion) and passes it through the trained model (UNet++ for

deployment, also implemented for DeepLabV3 and Swin Transformer). The model's output is processed with a sigmoid activation and thresholded at 0.5 to generate a binary mask. The show_prediction function displays the original image (resized to 256x256) and the predicted mask side-by-side using Matplotlib. For the Flask application, the predicted mask is saved as a PNG file in the static directory for web display. This module ensures users can visually interpret the segmentation results, critical for validating leaf region detection.