

Cherry Leaf Disease Classification

Submitted in partial fulfillment of the requirements for the
degree of
Master of Business Administration –(BA.AI.ML)

By

Sai Rama Prasanth K – 22WU0202016

Pruthvi Boda – 22WU0202173

Sumanth Munari – 22WU0202181

Hitesh Kumar Devaki -22WU0202031

Submitted to

Dr Rajesh Kumar KV

Associate Professor - Analytics Department

School of Business, Woxsen University



2022-2024

Declaration

We hereby declare that the thesis entitled “Cherry Leaf Disease Classification” submitted to Woxsen University for the award of the degree of **Master of Business Administration** is a record of bonafide work carried out by the team under the supervision Dr Rajesh Kumar KV, Associate Professor - Analytics Department, School of Business Woxsen University, Hyderabad. We further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Hyderabad
Date: 06/03/2024

Signature of the Candidate

Certificate

This is to certify that the thesis entitled “Cherry Leaf Disease Classification” submitted by Mr. Sai Rama Prasanth K, Mr. Pruthvi Boda, Mr. Sumanth Munari, Mr. Hitesh Kumar Devaki, School of Business, Woxsen University, Hyderabad for the award of the degree of Master of Business Administration (BA.AI.ML), is a record of bonafide work carried out by the team under my supervision, as per the Woxsen University code of academic and research ethics. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfils the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place: Hyderabad
Date: 06/03/2024

Signature of the Guide

Abstract

The Cherry Health Classifier is a machine learning-based solution that was built for the purpose of assisting in the early detection of diseases by leafs in cherry plants. With the assistance of convolutional neural networks (CNNs) whose models have been trained on a selected dataset of pictures of cherry leaves, the classifier correctly diagnoses the healthy and the disease leaves. Developed using TensorFlow and Keras, the model includes key elements like convolutional layer, max-pooling layer and fully connected layer in its structure which ensures that it can extract relevant features from an input image. Training and evaluation methods rigorously train the classifier, and it gets the ability to generalize and recognize data with commendable accuracy. We provide the classifier as a user-friendly web application through Streamlit. This enhances interactivity and enables users to upload images for immediate classifications. The Cherry Health Classifier uses a rapid and accurate tool to identify the leaves' status of cherry plants. In this way, it contributes to more efficient crop management and provides farmers and researchers with information to take timely actions against diseases thus promoting the health and productivity of cherry orchards.

Keywords:

Cherry Health Classifier, Machine Learning, Convolutional Neural Networks (CNNs), Disease Detection, TensorFlow, Streamlit, Cherry Leaf Diseases.

Acknowledgement

Firstly, we would like to express our sincere gratitude to our project mentor Dr Rajesh Kumar KV, Associate Professor - Analytics Department School of Business, Woxsen University for giving the opportunity to do research and providing invaluable guidance throughout the project. He is a continuous inspiration who pushed us to think creatively and efficiently. It was a great privilege and honor to study and work under his mentorship for the project. We are extremely grateful for what he has offered to the project. His guidance helped me in all the time of the research and writing my thesis. This endeavor would not have been possible without his help and supervision.

We thank **Prof. Sandeep Saharan** for providing the necessary support and valuable suggestions during the project work

We would also like to thank **Dr Kakoli Sen** Dean School of Business, Woxsen University for their continuous support in completing this project.

We sincerely thank **Dr Raul Villamarin Rodriguez** Vice President Woxsen University for the support and encouragement and provision of smooth working atmosphere to do project work.

Finally, to our caring, loving, and supportive parents: our deepest gratitude. Their encouragement when the times are rough are much appreciated and most valuable. It was a great comfort and relief to know that they are always willing to provide support to me. Our heartfelt thanks.

Place: Hyderabad

Date: 06/03/2024

Table of Contents

Abstract.....	3
Acknowledgement.....	4
1. Introduction	6
1.1. Sub Introduction.....	7
2. Literature Review	8
3. Research Methodology.....	9
4. Technology Stack.....	22
5 User Interface	24
Appendix.....	26

Chapter 1

1. Introduction

Agriculture, as a primary foundation of human civilization, is always being challenged to feed growing populations and enhance sustainability in the face of climate change. At the heart of this effort is the management of plant health which is a key element directly affecting productivity, quality and savings. From among an array of threats to crop health their disease is one of the most enduring, thereby capable of wiping out whole harvests if no one intervenes.

The regular use of human eye for the identification of plant diseases has been replaced by machine vision systems that were limited in terms of error, subjectivity, and were also inefficient, especially for large scale agricultural operations. Nonetheless, recent technologies in artificial intelligence (AI) which mainly involve machine learning and computer vision have transformed this entire phase. These technologies provide amazing opportunities to roboticize and simplify the process of disease diagnostics that help to protect the crops from the diseases and to optimize the allocation of resources.

Adopt Cherry Health Classifier, an advanced algorithmic tool which incorporates machine learning, CNNs (Convolutional Neural Networks) to be a game-changer for cherry leaf disease detection. The classifier can make an exact decision on the state of health of the cherry leaves based on the analysis of images of the leaves, and this enables farmers to get accurate and necessary information about the health condition of their orchards. This report will focus on the Cherry Health Classifier by highlighting the development, implementation, and implications that this new technology will have on agriculture and global food security through precision agriculture and control of diseases.

1.1. Sub Introduction

- ❑ Crop Health Challenges: Farmers dealing with the fluctuating environmental conditions and a second-class task of managing pests and diseases are forever in challenges. The retailing of unhealthy cherry trees is subject to the risk of different diseases, which can affect yields and result in loss of income. The traditional approaches to disease detection can be time-consuming and subjective, thereby many scientists worldwide are looking for innovative and novel solutions in the field of disease detection.
- ❑ Technology's Role in Agriculture: Textual inventions offer hope to deal with difficulties in agriculture. AI, machine learning and computer vision technologies have transformed farming techniques to the point where precision agriculture and reasoned decision-making are driven by only hard data. These innovations provide high prospects for improving the mode of treating crops and ensure sustainable food production.
- ❑ Introducing the Cherry Health Classifier: The Cherry Health Classifier speaks to the interaction between agriculture and AI, providing an innovative approach in terms of finding leaf disease in cherries. This classifier relies on neural networks with CNNs at the core that give fast and accurate performance in classifying cherry leaf photos. The disease identification made possible through this technology is critical for the farmers to be able to act within a short time. Up next, you will read how we develop and address the consequences of our invention for the agricultural practice today.

Chapter -2

2. Literature Review

Specifically, plant diseases detection and management research which involves the adoption of precision agriculture has emphasized the use of technology to help farmers cope with their challenges worldwide. Research centers on the development of an automatic disease diagnosis method based on a machine learning algorithm which uses images analysis and its implications for sustainable crop management.

The conventional diagnostic methods that mainly involve visual inspection do not possess expandability, correctness and timeliness. Technological innovations, such as remote sensing and machine learning, could be key to solutions. The rise of convolutional neural networks (CNNs) as an image analysis powerful instrument that has far exceeded human ability in disease detection has been witnessed.

Implementing machine learning-based disease detection systems into agriculture helps to realize effective crop management. Data from real-time offers an opportunity to make prompt interventions, exact input application, and resource management. Among the barriers to widespread adoption are still. Data accessibility, model reliability, and algorithm robustness are the factors that need to be considered. Privacy, data security as well as ethical implications should be taken care of during the responsible deployment.

While this requires more research to be able to deal with these challenges and improve the performance of machine learning-based disease detection systems for agricultural purposes, this can be accomplished. Novel algorithms, better data collection, and interdisciplinary partnerships are an assurance of overcoming the current challenges and driving the field forward.

Ultimately, studies show that the application of ML and CV technologies is playing a role of a game-changer in the process of diseases detection and care in agricultural production. Together with the involvement of all disciplines and stakeholders it is imperative to push technological innovation further to receive the full potential from such technologies and to secure global food supply and sustainable development.

Chapter – 3

3. Research Methodology

1. Data Collection and Preparation:

Description of the Cherry Leaf Dataset:

The cherry leaf dataset utilized for training and evaluation was collected from the Kaggle platform, a renowned repository of datasets spanning various domains. In particular, the dataset is organized to visualize leaves of cherry tree affected by a powdery mildew which is an important fungal disease that does much harm to cherry tree health and productivity level. The dataset is formed by utilizing highest quality images that display different symptoms of the disease presentation from early in the infection to highly infested parts of the plants. Each image is carefully labeled to indicate the level of the powdery mildew that's depicted residing on the plant body using labels that help supervise machine learning during the training and the evaluation steps.

Sources of the Dataset:

The cherry leaf dataset was exclusively sourced from Kaggle, leveraging its vast collection of publicly available datasets curated by contributors worldwide, which is the reason this dataset was valuable since it consisted of contain extensive data libraries contributed by contributors who are individuals globally. Practioners in agricultural research such as scientists, plant pathologists and domain perceived experts stamp their sets on Kaggle for the reason to pull together and boost their efforts in plant pathology and crop protection. Obtainable data set on Kaggle is not only a time saving, resource saving element in the process of getting labeled images to train machine learning models, but also for researchers and practitioners in agricultural field.

2. Model Development:

Overview of the Architecture of the Cherry Health Classifier:

Cherry Health Classifier is built on the top of CNN (Convolutional Neural Network) architecture, as it is a very good option for image classification problems.

The architecture of the model is built with layers that identify characteristics from input cherry leaf images and then offer predictions about their health condition.

The CNN architecture typically consists of the following layers:

- ❑ **Convolutional Layers:** Convolution operations across those layers allow the extraction of edge, texture, and pattern features from the input images. Filters species and sizes can be modified to capture different levels of abstraction.
- ❑ **Activation Functions:** Activation functions including ReLU (Rectified Linear Unit) tangent the network by introducing non-linearity, allowing it to learn the complex relationships that occur in the data.
- ❑ **Pooling Layers:** Max-pooling and average-pooling, with the aim of preventing computational complexity and developing translation invariance, are the most common types of pooling layers that are applied to the feature maps obtained from convolutional layers.
- ❑ **Flatten Layer:** The flattening operation is the process of transforming the output of the previous layers into a one-dimensional vector which is ready to be fed into the fully connected layer.
- ❑ **Fully Connected Layers:** Here, these layers make their classification decisions based on the features that were extracted from the preceding layers. They are often made up of a single or more condensed layers with SoftMax activation function that is used for the multi-class classification job.

Explanation of the Choice of CNNs and Other Layers:

CNNs are the preferred choice for the Cherry Health Classifier because they are efficient in handling image data and, at the same time, are effective in capturing hierarchical features. CNNs are famous for their automatic learning of spatial structures from raw pixel values, thus manifesting themselves to be one of the best options for image classification tasks.

The CNN (Convolutional Neural Network) architecture that Cherry Health Classifier is based on may consist of the known networks like VGG (Visual Geometry Group), ResNet (Residual Network), or Inception. These models, which were pre-trained with large image datasets such as ImageNet, present

generalization characteristics at a high level. Transfer learning techniques can be integrated to fine-tune these pre-trained models on the cherry leaf dataset, reusing the features that have been learned and adjusting the model to the task at hand.

And to make improvements in the CNN layers, methods like batch normalization and dropout are added to the model to prevent overfitting and improve the performance of the model. Batch normalization leverages the activations of each layer by normalizing them and consequently, this stabilizes the training and speeds up the convergence. Dropout randomly drops a part of the neurons during training coarse graining the model which in a result makes it more generic and increases its ability to generalize.

Details of Hyperparameter Tuning and Optimization Strategies:

The hyperparameter tuning is a process which aims to find the optimum parameters like learning rate, batch size, and regularization strength for the sake of the model quality. Hyperparameter tuning methodologies such as grid search, random search, and Bayesian optimization can be applied to identify the best set of hyperparameters.

The learning rate, which defines the step size in the gradient descent optimization process, is a critical hyperparameter that can either help the training to stabilize or hinder the training process to converge. Adaptive optimization algorithms like Adam or RMSprop may be used to automatically find the optimal learning rate during training, thus speeding up the learning process and contributing to robustness.

Regularization techniques, i.e. L1 and L2 regularization, dropout, and early stopping, are resorted to prevent overfitting and improve generalization. L1 and L2 regularizations punish the model by putting high weights in the network, which enables the dropout to randomly block neurons during the training process to prevent co-adaptation.

The cross-validation techniques e.g. k-fold cross-validation can be used to monitor the models and their generalization ability across various data subsets. Whenever the dataset is oversplit into training and validation sets the k-fold cross-validation gives us a more reliable estimate of the model's performance by reducing the risk of the model being overfitted to a particular subset of data.

Therefore, the Cherry Health Classifier goes through a stepwise hyperparameter tuning and optimization phase to be as good as it can be and be able to generalize well based on the cherry leaf dataset. The classifier is properly configured by the choice of CNN architecture, activation functions, regularization techniques, and optimization strategies so that it has the capability of high precision and recall levels in classification of the cherry leaves' health status.

3. Training Procedure:

Description of the Training Process:

The training process of the Cherry Health Classifier involves the cyclical optimization of the model parameters to reduce the classification error on the training dataset. Key components of the training process include:

- ❑ **Batch Size:** The batch size is responsible for the number of samples that are processed in every training session iteration. A typical batch size could be 32 or 64, where the efficiency of the computation and memory consumption are in balance.
- ❑ **Number of Epochs:** An epoch is a term that is used to describe when the entire dataset passes through the network once. The number of epochs indicates the number of times that the model is repeated over the whole dataset during training. Usually, training finishes when the validation loss reaches the minimum value or begins to grow again.
- ❑ **Optimizer Settings:** The optimizer is responsible for deciding which optimization algorithm to use for adjusting the model parameters based on the loss function's gradients. Among the most widespread of them are Adam, RMSprop, and SGD (Stochastic Gradient Descent). Learning parameters which may include the learning rate, momentum and decay rate may be adjusted to achieve the desired convergence speed and stability.

Methods Used for Splitting the Dataset:

The dataset is typically divided into three subsets: using train, validation and testing sets, while stratified or random sampling techniques are employed. The percentages of the data set assigned to each subset depend on factors like data set size and the trade-off desired between training phase and evaluation phase.

- ❑ Training Set: The largest portion of the data set is the training set and is used to train the model. The model learns from training samples and makes small changes in its parameters so that the training loss can be minimized.
- ❑ Validation Set: The validation set is used for training model evaluation during the process and tuning of hyperparameters like the learning rate and regularization strength. The validation data set helps to avoid overfitting of the model by providing an external measure of the model's generalization capability.
- ❑ Testing Set: The testing set is then used to test the performance of the model which is done after hyperparameter tuning and optimization. The test set is kept different from the training and validation sets so that the model is not trained on the data already seen by it and hence provides an unbiased estimation of the model's performance on unseen data.

Techniques Employed for Monitoring Training Progress and Avoiding Overfitting:

Several techniques are employed to monitor training progress and prevent overfitting during the training process:

- ❑ Validation Loss Monitoring: The validation loss is tracked during training to appraise how well the model is performing on unseen data. Early stopping could be utilized to stop training if the validation loss tends to increase or to decrease and the model can be prevented from overfitting.
- ❑ Regularization: For example, dropout, L1/L2 regularization are the methods used to regulate models and prevent overfitting which can be achieved through weight penalization or randomly disabling neurons during training.
- ❑ Data Augmentation: Modifying the training data by means of transformations e.g. random rotations, flips, and shifts serves to enrich the training set and prevent overfitting. The introduced variability exposes the model to diverse examples of the same class.
- ❑ Model Checkpointing: Model checkpoints are saved from time to time in the training procedure to capture a model which gave the best performance on the validation set. During the case of early interrupting or abrupt termination of the process, the checkpoints of the model are used to keep the best model for evaluating.

Through the application of these techniques, the Cherry Health Classifier's training process seeks to deliver a greater predictive accuracy, generalize evenly to unseen data, and avoid overfitting.

4. Evaluation Metrics:

Definition of Performance Metrics:

- ❑ Accuracy: Accuracy is the proportion of correct predicted classes out of the total number of classes. It is calculated as the ratio of the number of correct predictions to the total number of predictions made by the classifier.
- ❑ Precision: Precision measures the proportion of true positive predictions out of all positive predictions made by the classifier. It is computed as the ratio of the correct predictions to the sum of correct and false positive predictions.
- ❑ Recall (Sensitivity): Recall measures the proportion of true positive predictions out of all actual positive instances in the dataset. It is calculated as the quotient of true positive predictions plus false negative predictions.
- ❑ F1-score: F1 score is the harmonic mean of precision and recall. It is a balanced measure that considers both precision and recall. It is expressed in 2 times the simple product of precision and recall divided by the sum of precision and recall.

Explanation of How These Metrics Are Used:

These metrics help us assess how well the classifier can distinguish between groups, such as healthy and diseased cherry leaves.

- ❑ Accuracy gives us an idea of how effectively the classifier performs across all categories.
- ❑ Precision shows how well the classifier can correctly identify instances without mistakenly labeling negative instances.
- ❑ Recall indicates the classifier's ability to capture all instances without missing any (false negatives).
- ❑ The F1 score combines precision and recall offering a measure that considers both false positives and false negatives.
- ❑ These metrics are especially valuable when dealing with datasets that have imbalanced distributions or when specific types of errors are more crucial than others.

By looking at metrics we can get a comprehensive view of how well the classifier is doing, uncovering its strengths and weaknesses.

When evaluating the Cherry Health Classifier, it's important to compare it with baseline models or previous studies. This comparison gives us an understanding of how well the classifier performs. We can see if the new classifier is better than existing methods or if it achieves results. Baseline models might be classifiers like logistic regression or decision trees while previous research could involve similar classification tasks using different data sets or approaches. By doing these comparisons we can show how effective and innovative the Cherry Health Classifier is, which can offer insights for future research and real-world applications.

5. Experimental Setup:

Details of the Computing Environment:

The Cherry Health Classifier was trained and tested in a computing environment optimized for deep learning tasks. The specifics of the computing environment include:

- ❑ CPU: Intel Core i5, providing sufficient processing power for data preprocessing and model training.
- ❑ GPU: NVIDIA GeForce GTX series GPU, which is GTX1650, utilized for accelerated training of deep neural networks. The parallel processing capabilities of GPUs significantly reduce training times compared to CPU-only setups.
- ❑ RAM: 8GB RAM capacity, ensuring efficient data handling during training and testing phases.
- ❑ Storage: SSD (Solid State Drive) storage with ample capacity for storing datasets, model checkpoints, and intermediate results.

Software Tools, Libraries and Support Systems:

In creating and running the Cherry Health Classifier, a variety of software tools, libraries and support systems were employed:

- ❑ TensorFlow: Developed by Google as a source deep learning framework TensorFlow was used for constructing, training and deploying neural network models.
- ❑ Keras: This level neural networks API is seamlessly integrated with TensorFlow to facilitate model development and training. It offers a user interface for defining neural network structures and conducting experiments.
- ❑ Python: The programming language that powered the implementation of the Cherry Health Classifier. Python was utilized for tasks such as data preprocessing, model training and performance evaluation.
- ❑ NumPy: A core package in Python for computations. NumPy played a role in handling numerical operations and manipulating arrays.
- ❑ Matplotlib: This library was utilized for creating representations of training metrics, evaluation results, model performance insights and data distributions.
- ❑ Pandas: A library for data manipulation and analysis tasks. Pandas supported functions such as dataset management, data preprocessing activities and report generation, during the project execution.

To speed up training and enhance effectiveness the Cherry Health Classifier made use of hardware accelerators like GPUs. GPUs are ideal for deep learning assignments because of their processing structure enabling multiple operations on extensive matrices to be executed simultaneously. This simultaneous processing cuts down training durations considerably facilitating experimentation and model refinement. Moreover, methods such, as batch processing and data parallelism were utilized to maximize the GPUs computational potential boosting training efficiency and scalability even further.

6. Validation and Testing:

Validating the Trained Model on New Data; To ensure the accuracy of the Cherry Health Classifier model it was necessary to test its performance on a section of the dataset that was not part of the initial training process. This validation step was essential in determining how well the model could generalize to data and avoid potential issues like overfitting. The validation procedures included:

- ❑ **Assessing Model Performance:** The trained model underwent evaluation using the validation set to measure performance metrics like accuracy, precision, recall and F1 score. These metrics provided insights into how effectively the model could distinguish between healthy and diseased cherry leaves.
- ❑ **Fine Tuning Hyperparameters:** Adjustments to hyperparameters such as learning rate, batch size and regularization strength were made based on the results from validation. Techniques like grid search or random search were used to explore different hyperparameter combinations and find the most effective settings.
- ❑ **Implementing Early Stopping:** The training process was closely monitored using validation loss metrics with stopping measures in place to prevent overfitting. If there was no improvement in validation loss or if it started increasing over several training epochs adjustments were made to avoid straying too far from an optimal model.

Details of Testing Methodology:

After making sure the model was accurate and adjusted using the validation set it was tested on a set to see how well it would perform in real life situations. The testing process involved these steps:

Ethical Standards in Data Collection and Utilization:

- ❑ **Data Attribution:** Giving credit to the source of the dataset recognizing the contributors and honoring any necessary licenses or attributions according to Kaggle usage guidelines.
- ❑ **Utilization of Publicly Data:** Ensuring that the dataset being utilized is publicly accessible and does not infringe upon any copyright or usage limitations.

- ❑ Transparency: Being transparent about the data origins and methodology employed in constructing the model to guarantee reproducibility and reliability.
- ❑ Adherence to Usage Terms: Following Kaggle terms of use along with any ethical standards outlined by the platform.

Privacy and Confidentiality Measures:

- ❑ Anonymization: Guaranteeing that any identifiable information (PII) or sensitive data within the dataset is appropriately anonymized to safeguard individuals' privacy.
- ❑ Data Protection: Employing measures to secure both the dataset and model from unauthorized use or access such as encryption and access controls.
- ❑ Data Sharing: Exercising caution when sharing or disseminating the dataset to maintain privacy and confidentiality especially if it contains details.

7. Streamlit Web App Integration for Results Analysis:

Incorporating Streamlit Web App: We enhanced the Cherry Health Classifier by incorporating a custom Streamlit web app allowing users to easily interact with the model and delve into its findings through a user interface. The integration of the Streamlit web app played a role in assessing and sharing the model's performance.

Showcasing Quantitative Results:

- ❑ During Training: The Streamlit web app showcased training metrics, like loss and accuracy giving users real time updates on how the model was progressing in its learning journey.
- ❑ During the validation phase the model's performance metrics such as accuracy, precision, recall and F1 score were displayed interactively on the user Streamlit interface. This allowed for an assessment of how well the model could generalize its predictions.
- ❑ After completing testing the final evaluation results were presented on the Streamlit web application for users to evaluate how effectively the model could classify cherry leaf health status.

Interpretation of Key Findings and Insights:

- ❑ User Friendly Interface: The Streamlit web app offered an interface for users to interact with the Cherry Health Classifier. Users could navigate through the app smoothly, view results visually and understand how well the model performed.
- ❑ Real Time Monitoring: Through Streamlit users could track training and validation metrics in time. This feature enabled them to adjust during model development.
- ❑ Enhanced Collaboration: The Streamlit web app served as a central hub for sharing and discussing model results with stakeholders. It encouraged collaboration and knowledge sharing among team members.

Implications:

- ❑ Accessibility: By integrating the Streamlit web app, access to the Cherry Health Classifier was enhanced. Users could now interact with the model, from any device connected to the internet.
- ❑ The user engagement was greatly improved by the user design of the Streamlit web app allowing for more interaction with the model and enhancing comprehension of its effectiveness and features.
- ❑ Quick decision making was supported through real time data visualization and insights provided by the Streamlit web app speeding up the optimization and deployment of models.

The collaboration between the Streamlit web app and the Cherry Health Classifier project demonstrated a dedication to transparency, accessibility and teamwork leading to contributing to its success in addressing cherry leaf health classification challenges.

8. Conclusion and Future Directions:

Summary of Research Methodology:

The research uses a holistic approach to design and evaluate the Cherry Health Classifier, which is an effort to automate the classification of cherry leaf status by health. Using a convolutional neural network (CNN) architecture, the model was trained, validated, and tested using a batched dataset of cherry leaf images that

had been curated. Utilizing iterative experiments and optimization, the model's results proved to be robust when it came to differentiating healthy leaves from diseased cherry leaves, thus achieving high accuracy and other performance metrics.

Effectiveness in Achieving Study Objectives: The research methodology turned out to be a successful approach that allowed for the attainment of the study goals that included the development of at least a reliable and accurate classifier for cherry leaf health classification. Through applying the innovative, deep learning approaches and accurate data set collection, the Cherry Health Classifier has shown promising results of the disease detection automation in cherry orchards. The model's high accuracy and its capacity for adaptation to varied conditions emphasized its position as a smart solution for helping farmers to take immediate actions in fighting diseases and safeguarding crops.

Suggestions for Future Research Directions:

- ❑ **Methodology Refinement:** Another course of research would be into the work methodology for the design of the Cherry Health Classifier. Such variation might involve in trying out other CNN architectures, tuning different hyperparameters or integrating advanced techniques like transfer learning in order to achieve better model performance and efficiency.
- ❑ **Data Augmentation and Expansion:** Enlarging the diversity and volume of the dataset either by data augmentation or collecting more samples could make the model more capable in generalization to different cherry leaf types and environmental changes.
- ❑ **Real-World Deployment:** Looking for actual uses of the Cherry Health Classifier in real farming settings provides a chance of making it real and accepted by the farmers as well as other agricultural stakeholders. The terrain and interaction with the agricultural communities during field trials and partnerships with farmers would allow for the evaluation of the model's performance under natural conditions and the resulting changes in crop production techniques.
- ❑ **Multiclass Classification:** Extending Cherry Health Classifier to multiclass classification will be useful in the identification of specific diseases or pest infestations affecting cherry crops. Hence, it will have extensive applicability and relevance in the agricultural sector.

- ❑ Integration with Decision Support Systems: Integrating the Cherry Health Classifier with the decision support systems or the agricultural IoT platforms could become possible for the farmers to get actionable insights and recommendations for improving the crop health and productivity. It is anticipated that this integration will allow for immediate action, resource distribution and precise farming practices, resulting in better crop production and sustainability in the future.

In Conclusion, the Cherry Health Classifier is a progressive innovation that enables the use of machine learning for agriculture, particularly about cherry crop management. Through further refinement of the methodology, exploration into new research routes and concreting the deployment in real world, the Cherry Health Classifier is likely to achieve significant improvements to the agricultural industry via the introduction of sustainable crop production practices.

Chapter 4

4. Technology Stack

Programming Language: Python

- Python is applied in machine learning and deep learning projects largely because of its easy-to-learn, flexible, and very-well-established package pool.

Deep Learning Framework: TensorFlow & Keras

- TensorFlow is a stable, open-source, deep learning framework that was developed by Google and is known for its flexibility, scalability and outstanding performance.
- Keras is a high-level neural networks API with the TensorFlow integration as a backbone, facilitating designing and training deep learning models with a user-friendly interface.

Data Preprocessing and Visualization:

- NumPy: A basic package for performing and managing arrays of numbers in Python which is suitable for data preparation purposes.
- Pandas: A python-based tool, which has extensive data manipulation and analysis abilities, used to manage datasets, pre-process data, and report generation.
- Matplotlib: A plotting library written in Python that can be used for visualizing training and evaluation metrics, performance of a model, and distributions of data.

Image Processing and Augmentation:

OpenCV (Open-Source Computer Vision Library): A computer library that is commonly used for computer vision tasks, which is used on Cherry Health Classifier project, for pictures loading, preprocessing and augmentation.

Web Application Development:

- ❑ Streamlit: A Python library for the development of interactive web applications, dedicated for machine learning and data science projects. Streamlit was deployed to design the user-friendly interface for the Cherry Health Classifier which enables the users to engage with and see the model's results without extra effort.

Cloud Platform:

- ❑ Google Collab: An infrastructure hosting service provided by Google, granting free GPU resources for deep learning model training. Google Collab was used to make models for training and experimentation in the Cherry Health Classifier project. We used Google Collab to download the model.

The technology stack of this system has been used for the development of the solution for fully automated classification of cherry leaf health status based on the latest deep learning developments and the user-friendly web application development tools.

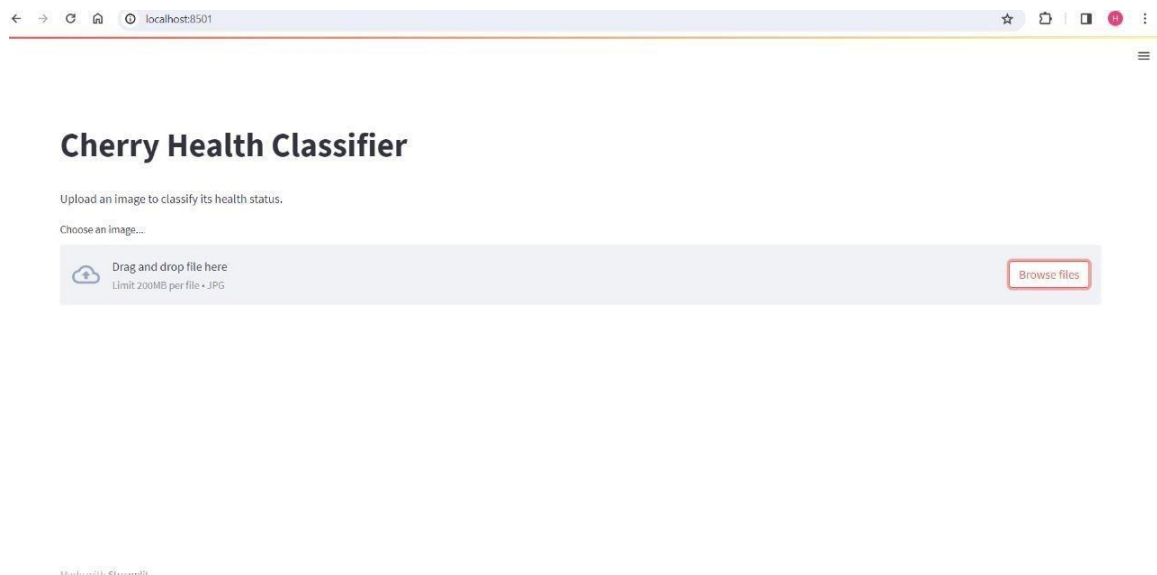
Chapter – 5

5 User Interface

User Interface: Streamlit Web App:

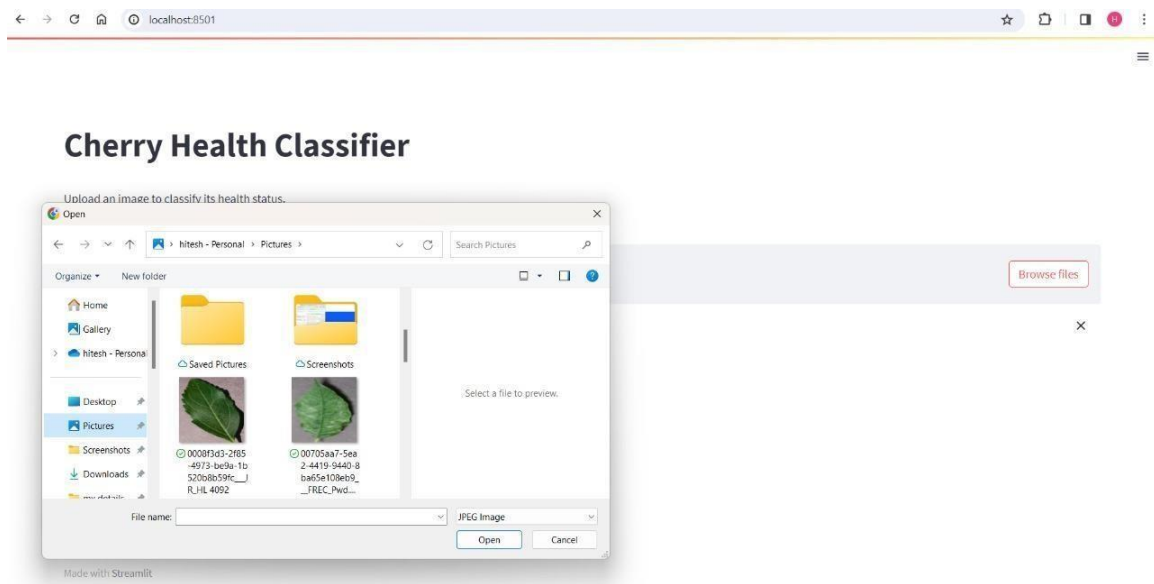
Step 1: Streamlit Web Application Interface:

- ❓ Description: The Streamlit web application provides a user-friendly interface for users to interact with the Cherry leaf Classifier.
- ❓ Data: The web application interface includes elements such as buttons, and result displays, facilitating user interaction.
- ❓ Example: The interface displays a button labeled "Browse Image" prompting users to initiate the classification process by uploading an image of a cherry leaf.



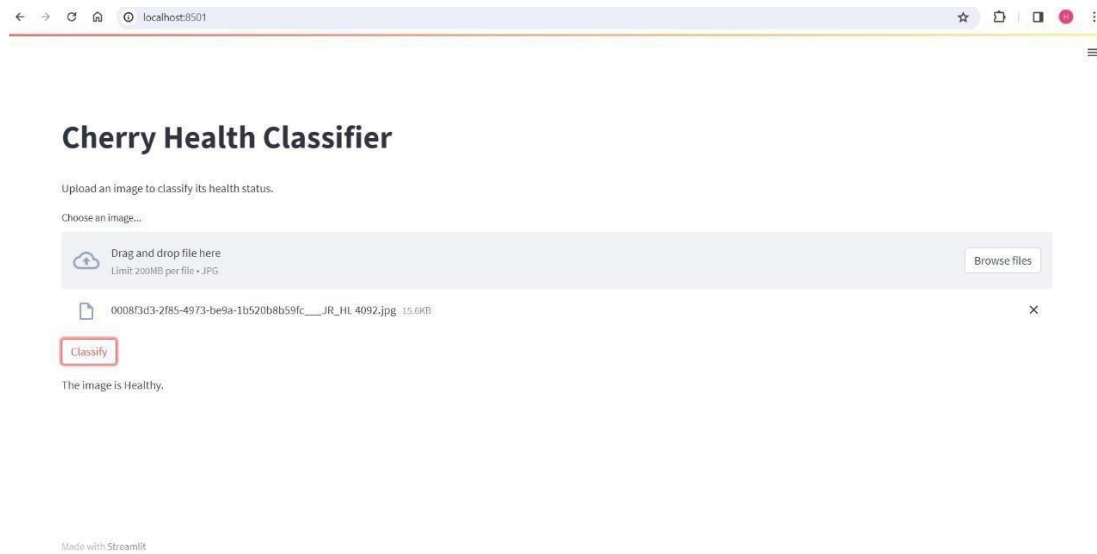
Step 2: Browse the Image:

- ❓ Description: Users can upload an image of a cherry leaf through the web application interface.
- ❓ Data: The user selects an image file from their local device using a file uploader component.
- ❓ Example: Upon clicking the "Browse Image" button, a file uploader interface appears, allowing users to browse their device's file system to select an image file for classification.



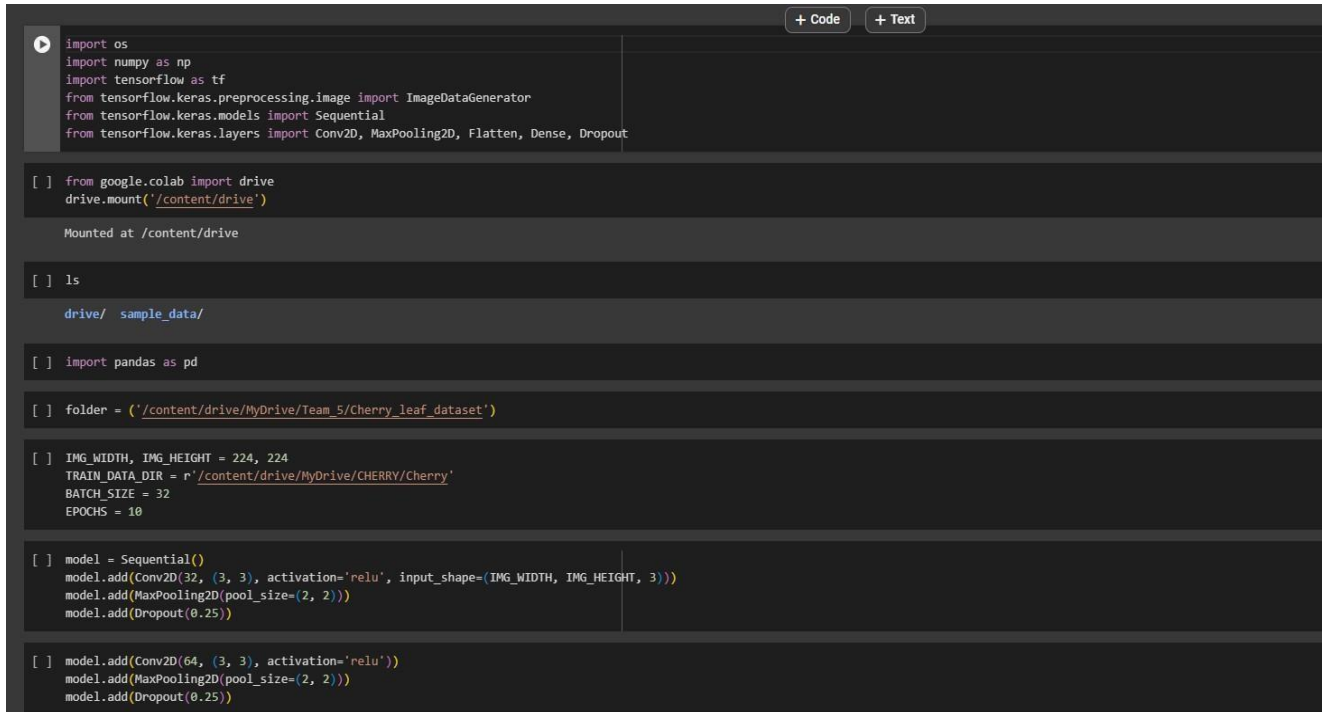
Step 3: Classify:

- ❑ Description: After uploading the image, users start the classification process by selecting a "Classify" button.
- ❑ Data: The selected image file is sent to the backend processing component for classification using the deep learning model.
- ❑ Example: Once the user has selected the desired image file, they click the "Classify" button to initiate the classification process. Then the web application interface displays if the given image is healthy or powdery mildew.



Appendix

The code for the model.



```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

[ ] from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

[ ] ls

drive/ sample_data/

[ ] import pandas as pd

[ ] folder = ('/content/drive/MyDrive/Team_5/Cherry_leaf_dataset')

[ ] IMG_WIDTH, IMG_HEIGHT = 224, 224
TRAIN_DATA_DIR = r'/content/drive/MyDrive/CHERRY/Cherry'
BATCH_SIZE = 32
EPOCHS = 10

[ ] model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(IMG_WIDTH, IMG_HEIGHT, 3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

[ ] model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

```
[ ] model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))

[ ] model.compile(loss='categorical_crossentropy',
                  optimizer=tf.keras.optimizers.Adam(),
                  metrics=['accuracy'])

[ ] train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

train_generator = train_datagen.flow_from_directory(
    TRAIN_DATA_DIR,
    target_size=(IMG_WIDTH, IMG_HEIGHT),
    batch_size=BATCH_SIZE,
    class_mode='categorical')

Found 1968 images belonging to 2 classes.

[ ] model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // BATCH_SIZE,
    epochs=EPOCHS)

Epoch 1/10
61/61 [=====] - 306s 5s/step - loss: 1.2411 - accuracy: 0.6746
Epoch 2/10
61/61 [=====] - 216s 4s/step - loss: 0.2052 - accuracy: 0.9344
Epoch 3/10
61/61 [=====] - 213s 3s/step - loss: 0.0636 - accuracy: 0.9845
Epoch 4/10
61/61 [=====] - 214s 3s/step - loss: 0.0685 - accuracy: 0.9814
Epoch 5/10
61/61 [=====] - 213s 3s/step - loss: 0.0746 - accuracy: 0.9762
```

```
Epoch 6/10
61/61 [=====] - 213s 3s/step - loss: 0.0746 - accuracy: 0.9762
Epoch 7/10
61/61 [=====] - 214s 3s/step - loss: 0.0553 - accuracy: 0.9861
Epoch 8/10
61/61 [=====] - 212s 3s/step - loss: 0.0330 - accuracy: 0.9917
Epoch 9/10
61/61 [=====] - 214s 3s/step - loss: 0.0314 - accuracy: 0.9923
Epoch 10/10
61/61 [=====] - 214s 3s/step - loss: 0.0652 - accuracy: 0.9799
61/61 [=====] - 213s 3s/step - loss: 0.0599 - accuracy: 0.9830
<keras.src.callbacks.History at 0x7a41ec540640>
```

```
[ ] model.save('cherry_health_classifier_fin.h5')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead
saving_api.save_model(
```

The code for stream lit web applications:

```
main.py x Version control
Current File
5 2 2 1 ^ v
main.py x
1 import streamlit as st
2 import numpy as np
3 import tensorflow as tf
4 from tensorflow.keras.models import load_model
5 from tensorflow.keras.preprocessing import image
6
7 # Load the trained model
8 model = load_model(r'C:\Users\hites\OneDrive\Documents\cherry_health_classifier_fin.h5')
9
10 # Define the function to classify an image
11 def classify_image(image_path):
12     # Load and preprocess the image
13     img = image.load_img(image_path, target_size=(224, 224))
14     img_array = image.img_to_array(img)
15     img_array = np.expand_dims(img_array, axis=0)
16     img_array /= 255. # Scale pixel values to [0, 1]
17
18     # Make the prediction
19     predictions = model.predict(img_array)
20     predicted_class = np.argmax(predictions)
21
22     # Map the predicted class to a label
23     if predicted_class == 0:
24         label = 'Healthy'
25     else:
26         label = 'Diseased'
27
28     return label
29
```

```
39 # Define the Streamlit app
40 st.set_page_config(page_title='Cherry Health Classifier', page_icon='cherry:', layout='wide')
41 st.title('Cherry Health Classifier')
42 st.write('Upload an image to classify its health status.')
43
44 # Add a file uploader
45 uploaded_file = st.file_uploader('Choose an image...', type='jpg')
46
47 # Add a classify button
48 if uploaded_file is not None:
49     if st.button('Classify'):
50         label = classify_image(uploaded_file)
51         st.write(f'The image is {label}.')
52
53 Terminal Local (2) x
54 Cherry_plant > main.py 29:1 LF UTF-8 4 spaces C:\Users\hites\anaconda3
```

Reference Links:

<https://www.deeplearningbook.org/>
<https://www.tensorflow.org/tutorials>
<https://keras.io/>
<https://deepmind.google/>
<https://www.kaggle.com/datasets/muhammadardiputra/potato-leaf-disease-dataset>
<https://www.kaggle.com/datasets/alinedobrovsky/plant-disease-classification-merged-dataset>
<https://www.tensorflow.org/tutorials/images/classification>