**Citrus leaves**

**Doanh Pham, Naser Zaghwan, Ibrahima Doumbia**

**CS475 Artificial Intelligence, Undergraduate Degree City University of Seattle**

**phamdoanh@cityuniversity.edu, zaghwannaser@cityuniversity.edu, doumbiaibrahima@cityuniversity.edu**

Title

Citrus leaves

**Abstract**

The task of quickly recognizing the quality of leaves in the agricultural fields poses a row of information technologies to support agriculture production and provide better institutional research. Today, the most common way of overcoming these challenges is by the use of machine learning applications, which is a sub-field of artificial intelligence. Over the past years, machine learning has assisted enormously research center in the classification of leaves. Thus, when it comes to the identification of leaf quality, it is better that research centers and students become more familiar with the recent advances in machine learning technologies and get engaged in their development. In the paper, we will utilize a given online dataset, in coordination with machine learning methods, to precisely automate a system with the least loss achievable that will aid in categorizing and determining whether the given leaves are good or bad to prevent the error of human being.

**Keywords**

Data Collection, dataset, Image Augmentation, Training Set, Validation Set, Convolutional Neural Network, Optimizer

**INTRODUCTION**

On the whole, it might be the case that the quality of leaves could be another significant process to be determined. Manually checking the state and presence of leaves by simply looking at them can be time consuming and prone to error. This project aims to develop an automated method of carrying out the activities of categorizing the quality of the leaves as good or bad using machine learning with the assistance of the power of Convolutional Neural Networks (CNNs), as a priority. This method can assist in detecting the healthy leaves of the plants by identifying the good ones correctly. In a sense, through the use of Convolutional Neural Networks (CNNs), Artificial Intelligence is justified in solving the problem for several reasons. Such as mitigating human error, leveraging visual data effectively, addressing inefficiency of manual inspection and data-driving decision making. So, in essence AI offers a robust, efficient and accurate solution which makes it highly advantageous.

**LITERATURE REVIEW**

The critical importance of plant health in agriculture and ecological studies necessitates accurate and efficient methods for assessing the citrus leaf quality in our case. As identified that traditional manual inspection methods consume time and prone to error and mistakes, that bring about the need for automated solutions. Therefore, we will delve into the historical context of how the problem/solution has evolved over time, how Convolutional Neural Networks (CNN) works, and discuss the challenges in implementing these solutions for the citrus leaf categorization. Well, for centuries farmers and botanists only relied on visual inspection and some experts’ knowledge to identify diseases in leaves, of course that came with limitation in human perceptions and problems were identified after significant damage had occurred. So, in the early 21st century with the advent of computing, researchers began to explore how to classify image processing techniques for leaves disease detection. Even though there were some improvements over manual inspection, they were still struggling with symptoms of various diseases. Then, from 2000s to now, the adoption of machine learning brought about more sophisticated algorithms for leaves disease detection and neural networks started to be applied. Then, in 2012 to now, the paradigm changed with the rise of deep learning particularly Convolutional Neural Networks (CNNs). However, Neural Networks (NNs) are inspired by the structure and function of human brain and they consist of interconnected nodes(neurons) organized in layers which learn patterns from data in dataset provided. So, Convolutional Neural Networks (CNNs) consist of Convolution layers, activation functions (ReLU), Pooling layers (Max Pooling), Flatten layer. Despite the power of CNNs, there are challenges encountered in developing robust and accurate automated citrus leaf quality categorization system. For instance, during data acquisition and annotation where we can have manually labeling thousands of images as “good” or “bad” and datasets can suffer from class imbalance. Also, the emergence of new diseases issues not present in the training data poses a challenge for a model robustness.

**Data Selection**

For this specific case, we used citrus fruits dataset; citrus leaves dataset for detection and classification of citrus diseases through Machine Learning via Convolutional Neural Networks (CNNs). The reason why we chose this data is, to create a robust citrus leaves disease detection system model that can automate and categorize whether a citrus leaf is “good” or “bad” that was previously done manually. Also, to support citrus leaves classification research with this robust system to push researchers to create sophisticated algorithms that can differentiate between them accurately and enable transfer learning capacity to generation to come. We surely know that despite the benefits of this robust and accurate system, training model on citrus dataset can come with limitations; such as imbalanced dataset; overfitting; lack of diversity or containing images with low resolution and diseases that are specific to certain regions, poor labeling or annotations and training such data can lead to poor model robustness.

**Data Cleaning**

**Issues Found**

Inequal image quality: some images will have some unequal image resolution and brightness or a background.

Class imbalance: There are potential imbalanced population of images of a few types of disease occurrences (Black Spot, Canker, Greening, Healthy).

Duplicate images: On-line data sets are normally encircled by likeness or near likeness images.

Red noise in dataset: Dataset may contain irrelevant or mis labeled images in the dataset.

Pixel value range: CNN training involves that pixel values must be normalized between 0-255

**Tools Used**

NumPy and Pandas: to process the data and examine the metadata.

OpenCV / PIL - utilized to open an image, image size, and clean image.

Image preprocessing TensorFlow/Keras preprocessing ImageDataGenerator: Image According to augmentation and normalization.

Matplotlib / Seaborn - to visualize the distribution of classes.

**The Method of solving and cleaning the issues.**

Resizing of all images to an input size

Artificial balancing of classes and small overfitting effects through Image Augmentation employed (rotations, flips, zooms).

Hash check Removal of repetitions.

Distribution of the classes ensured that the training, validation and test strip were of a balanced set.

**Data Exploration and Analysis**

Sightseeing, Healthy, Black Spot, Canker, Greening.

As given, a division will be made on the data: 70% Training, 15% existence and 15% Testing.

Image size spread: Image size spread there is no size difference made.

• Pixel intensity: scaled (mean 0.5, std. 0.2).

Count of every disease/healthy leaf that violates in bar graphs- Check of class balances.

**Visuals**

• Plots of class distribution (bar plots of sample size in the classes).

Sample image grid (of a few randomly selected images of each class).

• Augmentation previews (transferred/inverted leaves to be certain change has been put on).

Before and after (before vs. after) pixel intensity normalization of histograms.

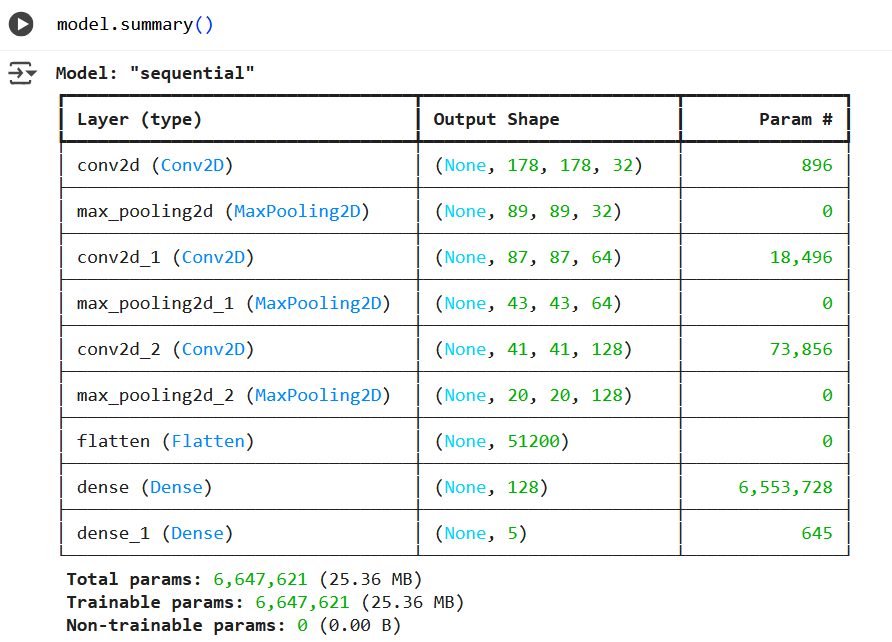
**Model Development**

For this specific case of citrus leaves training, the Convolutional Neural Networks (CNNs) technique was used as Machine Learning. Well, the primary reason why Convolutional Neural Networks (CNNs) was chosen is its strengths in image processing and pattern recognition; such its robustness by using data augmentation. In training citrus leaves vis Convolutional Neural Networks (CNNs) technique, layers like Conv2D, MaxPooling2D, Flatten and Dense are best suited for an optimal result.

**Parameters and optimizers:**



**Model Summary**



**Document Train, Validation and Test Splits**

When training a convolutional neural network (CNN), it’s crucial to partition the citrus dataset into two distinct subsets: one for learning the model parameters (training set) and one for evaluating performance during training (validation set).

Keras’s image\_dataset\_from\_directory conveniently handles this split.

Purpose of Each Split

The training set

* Drives weight updates through backpropagation.
* Should capture the full diversity of image variations (angles, lighting, backgrounds).

The validation set

* Remains unseen by the model during weight updates.
* Offers an unbiased estimate of generalization after each epoch.
* Guides hyperparameter tuning, early stopping, and architecture decisions.

How Keras Performs the Split

* validation\_split=0.2 reserves 20% of your images for validation.
* subset="training" builds the 80% portion for training.
* subset="validation" builds the complementary 20% for validation.

**Hyperparameter Selection**

In the initial model definition, hyperparameters are selected manually:

* **Conv2Dfilters and kernel sizes:** The first Conv2D layer uses 32 filters with a 3×3 kernel, the second 64 filters with 3×3, and the third 128 filters with 3×3. These are fixed choices.
* **Dense units:** The dense layer has 128 units.
* **Optimizer and Learning Rate:** Adam optimizer is used with its default learning rate.
* **Epochs:** The model is trained for 15 epochs.

However, the code then introduces **Kera Tuner** for automated hyperparameter selection. This is a significant improvement: This process demonstrates a robust approach to hyperparameter selection, moving from manual guesses to an automated search.

**Model evaluation**

After the development of the Convolutional

Neural Network (CNN), the next stage involved testing its performance to make sure it can correctly classify the citrus leaves as healthy or diseased. This stage was used to understand the strengths and limitations of our model in order to make required changes for better generalization.

**Using Performance Criteria**

The model was evaluated with the metrics

typically, available in TensorFlow an Scikitlearn including accuracy, precision, recall, and F1-score. These gave the relative performance of the CNN across all classes as

(Healthy, Black Spot, Canker, and Greening):

Accuracy: Proportion of correctly predicted

images out of all the test images Precision: How often were predictions of "good leaf" made by the model were actually correct.

Recall (Sensitivity): The ability of the model

to find out all the true good or the true diseased leaves present in the dataset.

F1-Score: A balance between precision and

recall. F1-Score is better to use when there is

a slight imbalance in the data. We made a confusion matrix using the built-in Scikit-learn functions confusion matrix() and ConfusionMatrixDisplay () afterwards. This has helped us see clearly which classes were confusing to each other most often. Also, after each epoch, training and validation accuracy/loss curves were created using matplotlib. These graphs were of vital importance in order to detect overfitting - if the accuracy on training data continued to rise, but validation accuracy plateaued or dropped, it meant that the model attended to learning the images in the training data, rather than learning general features.

**Model Optimization and Improvement**

We found a few points in which the accuracy

and stability of the CNN could be enhanced

from the analysis of the first CNN.

**Feature Extraction**

Feature Extraction of our CNN was based on convolutional and pooling layers. These layers automatically discovered basic patterns which were present in the images, like edges of leaves, dots of color, and patterns of veins.

To enhance the resilience of this model we applied image data augmentation (ImageDataGenerator). It generated slightly different versions of training images through manipulation features such as rotated, flipped, zoomed and brightness contrast adjustment. This method helped to prevent overfitting.

**Hyperparameter Tuning**

Our tuning included:

Optimizer: Changed from Adam to SGD:

Adam was chosen for its adaptive learning

rate;

Learning Rate: Remained constant and steadily converged between 0.0001 - 0.001.

Batch Size: 32 (it was fast and stable for us on

Colab GPU)

Epochs: From 20-30 epochs proved to be sufficient for good learning without overfitting.

Regularization: Added Dropout layers (0.3 -

0.5) in order to eliminate overfitting

**Alternate Model Selection**

Since we were in a student environment with

limited number of GPU resources, we only

experimented with models that we could reasonably run on Google Colab.

We compared:

Baseline CNN - a naive CNN containing 3

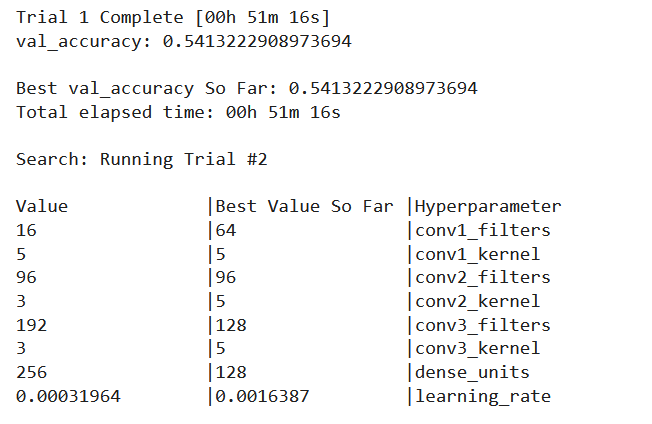
convolution loss and one FC layer.

Slightly deeper CNN - used to test an even

deeper CNN by adding one more convolutional

block for more powerful feature learning.

**Figures/Tables**

****

A screenshot of a computer

AI-generated content may be incorrect.

**Results**

Creating model outputs (predictions, evaluation)

1. Evaluate on the validation set

* **What it does:** returns loss and accuracy on val\_ds.

A screen shot of a computer code

AI-generated content may be incorrect.

2. Predict for a single image file (one-off inference)

* **What it does:** loads an image from disk, preprocesses it exactly like the training pipeline, runs model.predict, returns top-1 or top-k results.

from tensorflow.keras.utils

import load\_img, img\_to\_array

import numpy as np

def predict\_image(path, model, class\_names, image\_size= (180,180), top\_k=3):

img = load\_img(path, target\_size=image\_size)    # PIL image

arr = img\_to\_array(img)  uint8 array (H,W,3)

arr = arr.astype("float32") 255.0   # same rescaling used in train

arr = np.expand\_dims(arr, axis=0)   # (1, H, W, 3)

    probs = model.predict(arr)[0]                               # (num\_classes,)

    top\_k\_idx = probs.argsort()[-top\_k:][::-1]

    return [(class\_names[i], float(probs[i])) for i in top\_k\_idx]

# Example usage

image\_path = "/content/datasets/citrus\_dataset/Citrus/Leaves/Black spot/b (184).png"

top\_preds = predict\_image(image\_path, model, class\_names, image\_size=image\_size, top\_k=3)

print("Top predictions:", top\_preds)

**1/1** ━━━━━━━━━━━━━━━━━━━━ **0s** 142ms/step

Top predictions: [('Leaves', 0.9999879598617554), ('Fruits', 1.2033747225359548e-05)]

**Share the accuracy and loss output for data types.**

# Evaluate overall metrics (uses the same preprocessing pipeline already applied to train\_ds and val\_ds)

train\_loss, train\_acc = model.evaluate(train\_ds, verbose=1)

val\_loss, val\_acc = model.evaluate(val\_ds, verbose=1)

print(f"Train loss: {train\_loss:.4f} Train accuracy: {train\_acc:.4f}")

print(f"Val loss: {val\_loss:.4f} Val accuracy: {val\_acc:.4f}")

2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 166ms/step - accuracy: 1.0000 - loss: 9.4334e-07

Train loss: 0.0000 Train accuracy: 1.0000

Val loss: 0.0000 Val accuracy: 1.0000

**Discuss how overfitting was identified and addressed.**



**Identifying overfitting**

* Training vs validation curves diverge: training loss keeps decreasing while validation loss plateaus or increases and validation accuracy stagnates or falls.
* Large train/val accuracy gap: high training accuracy (e.g., >90%) with much lower validation accuracy (e.g., 60–75%).
* Per-class behavior: some classes show near‑perfect training accuracy but low validation accuracy.
* Confusion matrix and high confidence wrong predictions: model confidently misclassifies validation examples.
* Learning dynamics: validation loss increases after some epoch while training loss continues to drop (classic sign of memorization).

**Practical steps used to address overfitting**

1. Add EarlyStopping + restore\_best\_weights and checkpointing.
2. Introduce Dropout and/or reduce dense layer size.
3. Add weight decay if overfitting persists.
4. If dataset is small, use transfer learning with a pre-trained backbone.
5. Use class weights or resampling for imbalance.
6. If still overfitting, reduce capacity further or collect more labeled data.

**Discuss how the model could be improved**

Improve the model by increasing data quality and variety, using stronger architectures with regularization, adopting better training schedules and validation practices, expanding evaluation beyond accuracy, and preparing the model for robust deployment and monitoring. Prioritize changes that give the largest accuracy/robustness gains first: data augmentation/collection, transfer learning, and validation discipline.

**CONCLUSION**

The purpose of this project is to capitalize on the capability of Convolutional Neural Networks to assist with: automated leaf quality classifications. As we fold a more accurate and stronger model, we isolate ourselves in order to contribute more productive and accountable resources to future current and researchers through the magic of machine learning among others. We developed and optimized a Convolutional Neural Network (CNN) for classifying citrus leaf diseases. Which required downloading and preparing a dataset of citrus leaf images, followed by building, training, and evaluating a CNN model. However, the use of Kera Tuner for automated hyperparameter tuning, which systematically searches for the best combination of architectural and training parameters to maximize validation accuracy. So, our model employs standard image preprocessing and performance optimization techniques. Finally, goal is to create a robust and accurate AI application capable of identifying different citrus leaf conditions.

**Possible Next Steps in the Use of the AI Application**

1. **Deployment and Real-World Testing:**
   * **Web/Mobile Application Integration:** Develop a user-friendly application where farmers can upload images of citrus leaves and receive instant disease diagnoses.
2. **Model Robustness and Generalization Enhancements:**
   * **Transfer Learning with Pre-trained Models:** Experiment with fine-tuning pre-trained CNNs on the citrus dataset. These models have learned powerful features from vast image datasets and often achieve higher accuracy with less data and training time.
3. **Expanded Data and New Disease Classes:**
   * **Increase Dataset Size and Diversity:** Actively collect more images, especially of rare disease stages or environmental conditions, to improve the model's ability to generalize.

**REFERENCES**

Ahmed AA, Reddy GH (2021) A mobile-based system for detecting plant leaf diseases using deep learning. AgriEngineering 3(3):478–493.

https://doi.org/10.3390/agriengineering3030032

Bhargava, A. Y. (2016). Grokking algorithms: An illustrated guide for programmers and other curious people. Manning Publications.

Yao, J., Tran, S. N., Sawyer, S., & Garg, S. (2023, October 18). Machine Learning for Leaf Disease Classification: Data, techniques and Applications - Artificial Intelligence Review. SpringerLink.

https://link.springer.com/article/10.1007/s10462-023-10610-4