[11] **Paul Genre O. Lobaton**
[2] **Jean Eileen C. Magtiay**
[3] **Ven Gabriel E. Mendez**
[4] **Guillan Jude G. Moster**
[5] **Michael M. Casabuena**
[6] **Dr. Maricel M. Gaspar**
[7] **Philip V. Mojares**

LEAFLET: A Web-Based Leaf Classification System Using Convolutional Neural Networks

Journal of Electrical Systems

Abstract: - The Convolutional Neural Network (CNN), a crucial component of deep learning neural networks, is a network model frequently used in the classification of images, target recognition, and other applications. Utilizing this network model, this study aims to find the best CNN model compatible for the web-based application that can identify endemic trees found locally at FAITH College's Serenity and Labyrinth Garden with 90% accuracy rate. Similar to this study, Salem et al. (2019) tested different classifiers with both Flavia and their own dataset and found that K-Nearest Neighbors (KNN) displayed the highest precision among all. The study focuses on a web application utilizing the five pre-trained models namely VGG-16, ResNet50, Inception V3, Xception and Sequential, and it was found that the best performing among those is the VGG-16 which was then used for leaf classification, achieving a remarkable 94% accuracy, surpassing the 90% objective. The application allows users to scan leaves of specific trees in a designated area, with accuracy not guaranteed beyond this zone. The application's performance is influenced by the user's device, internet connection, and image quality.

Keywords: web application, leaf classification, convolutional neural networks, transfer learning, and deep learning model.

I. INTRODUCTION

The Philippines' forests hold and protect the country's unique and highly diverse biodiversity which cover 7.2 million hectares or approximately 24 percent of the country's total land area (Dabbadie, 2023). Among these forests are the 3,600 identified native trees, 67 percent of which can only be found in the country (de Jesus, 2021). Having that said, one way to truly appreciate the beauty of the country's nature is to connect with it through learning. Utilizing the emergence of new technologies, artificial intelligence (AI) can be used to develop systems that have the intellectual process characteristic of humans (Copeland, 2023) such as identifying patterns.

By harnessing the capabilities of artificial intelligence, particularly through machine learning techniques, systems can now emulate human intelligence to process and identify. Deep learning, a subset of machine learning, has significantly contributed to this progress even more so as it utilizes more options and alternatives. One of these powerful deep learning techniques, Convolutional Neural Networks or CNN, is a type of feedforward neural network that contains a convolutional layer with convolutional calculation function and depth structure of a multilayer neural network (Saha, 2023). Deep learning and artificial intelligence are both developing at a rapid rate, to the point where convolutional neural networks have more and more advantages in image classification (Huellman, 2022). Thus, resulting in its widespread use in business, agriculture, the military, and other sectors, such as botany. Plants are extensively scattered throughout nature, and with this idea, the initial step in botanical study or research is the identification of different plant species.

The Philippines, though rich in these trees and other plant specimens, lacks the proper tool for identifying local trees in our country. While some leaf classifications for trees internationally exist today, they apply the problem of not having all the needed data, mostly for trees endemic to one country like us. In addition, most of the leaf classification programs available in the market are developed as mobile applications, hence the creation of a web-based one which focuses on ease of use. Applying these ideas locally, the students from FAITH Colleges created a program that will allow the users to upload and scan an image of a leaf, which is able to identify from which tree the leaf comes from.

The application focused solely on classifying the leaves from the following 14 trees that can be found on Serenity and Labyrinth Garden of the said institution: Palawan Cherry *(balayong)*, Guava tree *(bayabas*), Betis tree**,** Dao tree**,** Blackboard tree (*dita*), Soursop tree *(Guyabano)*, Ilang-ilang tree, White lead tree *(Ipil)*, Sandpaper tree

[1,2,3,4] Students, FAITH Colleges, Tanauan City, Batangas, Philippines 4233
[5]Faculty-Adviser, Laguna State Polytechnic University, Los Baños, Laguna, Philippines 4030
[6,7]Faculty-Advisers, FAITH Colleges, Tanauan City, Batangas, Philippines 4233
[1]s2020103952@firstasia.edu.ph, [2]s2020102878@firstasia.edu.ph, [3]s2020102316@firstasia.edu.ph, [4]s2020102536@firstasia.edu.ph, [5] michaelcasabuena.dev@gmail.com, [6]mlmalabanan@firstasia.edu.ph, [7]pvmojares@firstasia.edu.ph

(Kalios), Ironwood tree *(Kamagong)*, Molave tree *(Mulawin)*, Philippine mahogany *(Narra)*, Mandarin orange tree *(Sinturis)*, and Lauan tree *(Yakal)*. With the help of Jupyter notebook, Python and its libraries – especially TensorFlow and Keras, HyperText Markup Language (HTML), JavaScript, and Bootstrap, it was possible to create a functioning web application named Leaflet, a leaf classification application that provides additional information about the leaf classified.

In addition to this, the researchers created this study that aims to find the best CNN model among the five (5) models tested that will generate a web-based application that can identify endemic trees from the Philippines with 90% accuracy rate, starting locally at FAITH Colleges. This study also aims to provide users with three (3) class probabilities where their detected leaf may fall. Moreover, this was created so that a local-based web application can be available to users keeping in mind the application's ease of use as it is convenient and accessible: taking away the hassle of downloading and installing mobile applications.

II. LITERATURE REVIEW

Nidhi and Yadav (2020) conducted a study about plant leaf classification using the Five Layer Convolutional Neural Network (FLCNNet) method to classify plants based on their shape. The study focuses on how the model performs between two sets of Flavia dataset: one with augmentation and one without one to check the effectiveness of augmentation since that is the aim of their study. It was found that the augmented datasets are more effective and more feasible for classification without misguiding the network to learn. They were able to obtain 98.7% accuracy through the model they used.

Another study regarding plant leaf classification was conducted by Rizk (2019) which uses a dual path and dual feature model testing it to the Flavia dataset. The author made use of shape and venation on the leaves as its classification features, making use of the sobel operators for the vein extraction. Sobel operator is a popular algorithm used for edge detection that uses two special kernels, one to detect vertical edges and the other to detect horizontal ones (Collins, 2019). The study found that shape is an essential feature in plant leaf classification, with venation as a complementary feature, however, the lack of venation on the leaves has caused the drop in accuracy. Nevertheless, the author was able to create a model with 96.8% recognition rate.

Yang et al. (2020) created a study that focuses on classifying leaves with a complicated background. Using their own dataset, they trained a model for leaf segmentation, the process of extracting the leaf area from the background of an image (Fan et al., 2022), using 2000 images fed into a Mask Region-based Convolutional Neural Network (Mask R-CNN). Additionally, they used 1500 images fed into a very deep convolutional network with 16 layers (VGG-16) to train a model for leaf classification. All in all, this study used more than 4000 images for model training and testing. Upon testing the VGG-16 model using 150 test images, the average accuracy value reached up to 91.5%, which indicates that the two networks mentioned can be used to segment and classify leaf images with complicated backgrounds.

One study from Kanda et al. (2021) made use of two different networks under deep learning: the Conditional Generative Adversarial Network (cGAN) used for generating synthetic data and Convolutional Neural Network (CNN) which they used for feature extraction. Moreover, they used a Logistic Regression classifier for efficient classification of the plant species. This method is seen as effective on seven datasets used for testing, generating a 96.1% accuracy on average. On the other hand, when the model was tested on individual datasets, it generated a 99.0 to 100% accuracy.

In the study by Turkoglu and Hanbay (2019), the concept of leaf feature extraction was redefined. Rather than focusing on the entire leaf structure, the researchers proposed a method that involved segmenting leaf images into smaller parts. This approach aimed to utilize specific image processing techniques to extract crucial features. By incorporating color analysis, vein identification, Fourier descriptors, and GLCM methods, the study achieved a remarkably high accuracy rate of 99.1% using the Flavia leaf dataset.

The study conducted by Keivani et al. (2020) focused on plant leaf image detection and classification, introducing an innovative method that combined Shape, color, vein, gist, and LBP features, utilizing a decision tree for classification. Notably, the research addressed the challenge of processing congested and uneven edge images, traditionally known for their computational complexity, by simplifying image processing using gist feature vectors, resulting in an accuracy exceeding 95%. Employing the Flavia and Folio Leaf Datasets, the study introduced a novel

feature reduction method, PBPSO, enhancing the performance of BPSO. Comparative analysis revealed the superior performance of the proposed system over existing techniques, with the decision tree method exhibiting outstanding accuracy rates of 98.58% and 90.02% on the Flavia and Folio leaf datasets, respectively, emphasizing the significance of the comprehensive feature set in effective plant leaf detection and classification.

The study by Wang (2019) introduces an enhanced convolutional neural network structure specifically tailored for leaf classification with limited sample sizes, addressing challenges related to sparse samples and various classification tasks. By extracting image features using this network and constructing a metric space based on feature similarity, the study emphasizes the pivotal role of metric space quality and supervised sample selection in determining the nearest neighbor classifier's accuracy. Notably, the research demonstrates that with just 20 training samples, the method achieves high classification accuracy, recording rates of 95.32%, 91.37%, and 91.75% on the Flavia, Swedish, and Leafsnap datasets, respectively. These findings position this approach as a competitive solution in the domain of deep learning-based leaf classification, offering potential for effectively handling challenges arising from limited sample sizes in diverse classification tasks.

Salem et al. (2019) The study investigates plant type identification in images, addressing challenges related to varying leaf shapes due to age and changing colors under diverse weather conditions. It assesses various handcrafted visual leaf features, extraction techniques, and classification methods, presenting a novel five-step algorithm for plant type recognition from leaf images. Evaluation on the ‘Flavia’ dataset (1600 leaf images) and a self-collected dataset (625 leaf images) demonstrates the algorithm's effectiveness. Different classifiers such as KNN, decision tree, naïve Bayes, and SVM were tested, with KNN displaying precision and recall rates of 97.6% and 98.8% on the ‘Flavia’ dataset and 96.1% and 97.3% on the self-collected dataset. The study affirms that the proposed approach, with efficient segmentation techniques on raw leaf images, provides highly accurate plant type recognition. Moreover, in comparison with AlexNet, a CNN-based approach, the handcrafted feature-based method proves more robust, especially with smaller training datasets.

III. METHODS

CNN Architectures and Transfer Learning Approach

A. Convolutional Neural Network (CNN)

Convolutional Neural Network or CNN is a deep learning algorithm that gained momentum and popularity in several studies and applications due to its capacity to learn relevant features from input images at different convolutional levels like the human eyes and brain capacity. Alzubaidi et al. (2021) found that the most used and well-performing CNN architectures are the VGGNet, ResNet50, Inception V3, Xception, and Sequential architectures. The Table below shows the comparison of the CNN architectures based on their layers and parameters. These architectures will be utilized throughout the study and the best performing among them will be chosen. This paper used the pre-trained weights of these architectures on the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) dataset during the training.

Table 1. CNN architecture comparison

CNN Architecture	No. of Layers	Parameters (millions)
VGG-16	16	14.8
ResNet50	50	22.2
Inception V3	42	22.2
Xception	71	21.5
Sequential	12	16.9

ImageNet is a research project to develop a large database of images with annotations e.g., images and their labels. Pre-trained models like VGG-16, InceptionV3, Xception and ResNet are already trained on ImageNet

which is composed of disparate categories of images. These models are built from scratch and trained by using high-end GPUs over millions of images consisting of thousands of image categories. As the model is trained on a huge dataset, it has learned a good representation of low-level features like spatial, edges, rotation, lighting, shapes, and these features can be shared across to enable the knowledge transfer and act as a feature extractor for new images in different computer vision problems. These new images might be of completely different categories from the source dataset, but the pretrained model should still be able to extract relevant features from these images based on the principles of transfer learning.

In addition, before leveraging transfer learning, our approach involved building a sequential model from scratch. This initial step was taken to gain a deeper understanding of how Convolutional Neural Networks (CNNs) function at a slightly lower level. Constructing a model from scratch allowed us to grasp the intricacies of filters and the role of each layer within a CNN. This foundational understanding proved to be effective, providing insights that influenced our subsequent use of transfer learning with pre-trained architectures. However, recognizing the vast potential offered by transfer learning, the researchers ultimately opted for this approach. The utilization of pre-trained models, already trained on a vast dataset comprising over 1.2 million images, allows us to take advantage of the knowledge and representations encapsulated in these models. Given the constraints of time and hardware resources, creating a model from scratch that could rival the performance of pre-trained architectures became impractical. Transfer learning enables us to leverage the advancements in the field, benefiting from models that have learned intricate features and patterns from diverse datasets, while also aligning with the resource limitations inherent in our study.

B. Transfer Learning

Transfer learning in the context of deep learning is a deep learning approach that allows the development of a network model using a dataset of fewer than 10,000 staples but on top of the pre-trained model architecture (Zhuang, 2019). In transfer learning, the network uses the information from primary tasks to enhance the generalization about a new task. Moreover, the last few layers of the trained network are replaced with new layers in transfer learning, such as the fully connected layer and a SoftMax classification layer, to learn only the features of the new dataset and its classes. Figure 1 shows the transfer learning pipeline as used in the context of this paper.

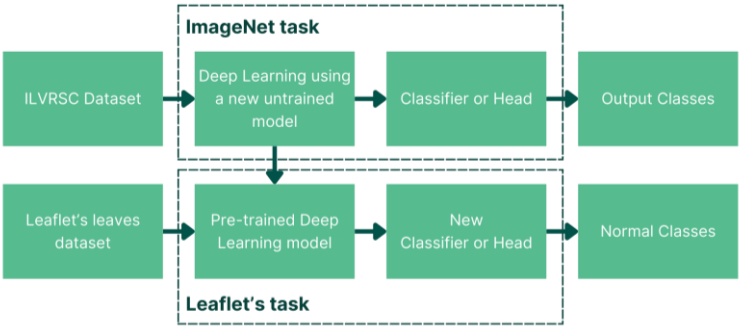

Figure 1. Transfer Learning Pipeline

Data Gathering

The data gathering process for the leaf classification project involved capturing images of ten different plant species: Palawan Cherry *(balayong)*, Guava tree *(bayabas*), Betis tree**,** Dao tree**,** Blackboard tree (*dita*), Soursop tree *(Guyabano)*, Ilang-ilang tree, White lead tree *(Ipil)*, Sandpaper tree *(Kalios)*, Ironwood tree *(Kamagong)*, Molave tree *(Mulawin)*, Philippine mahogany *(Narra)*, Mandarin orange tree *(Sinturis)*, and Lauan tree *(Yakal)*. The images were taken using multiple smartphones, and a total of 250 pictures were captured for each plant, resulting in a dataset of 3,500 images.

The plants selected for image capture were located in the "Serenity Garden" and “The Labyrinth” at FAITH Colleges. These areas provided a suitable environment for the selected plant species, ensuring the availability of healthy and representative leaves for photography. Figure 2 below shows how the researchers gathered data.

Figure 2. Researchers gathering data at Serenity Garden

Data Pre-processing

A. Reading and Image Data Conversion

To maintain data quality and dataset relevance, the researchers refined our selection process. The researchers meticulously handpicked 200 pictures from the initial set of 250 images per plant for training and validation. The selection criteria included evaluating the quality of the picture, ensuring minimal shadows, and verifying that the entire leaf was visible in the image. These criteria aimed to minimize potential biases and inconsistencies in the dataset, ensuring that the selected images were representative of each plant species. The selected images were instrumental in training and evaluating our leaf classification model, contributing to the accuracy and effectiveness of our research findings.

Furthermore, adhering to the standard 80 / 20 split in data science, which was driven by the Pareto principle (Ahmed, 2022), the researchers randomly allocated 160 images for training and reserved the remaining 40 for validation.

In the process of preparing image data for analysis, the researchers executed a series of steps to transform the raw image data into a suitable format for further work. Initially, the image data was read and subsequently opened as a binarized image, simplifying the way the data could be stored and transmitted. These transformations enabled seamless integration of the image data into the subsequent stages of the project.

B. Resizing the image

To accommodate the unique characteristics of certain leaves, such as those of Ilang-ilang and Mulawin as shown in Figure 3, the researchers deliberately set the image size to 225x300 pixels, resulting in an aspect ratio of 4:3. Notably, some leaves exhibit a longer length than width, and adhering to the standard square aspect ratio of 224x224 pixels could potentially lead to the exclusion of crucial details. The adoption of a 225x300 image size offers enhanced flexibility in capturing the varying proportions of leaves. Through testing, it was observed that images resized to this dimension consistently outperformed those constrained to a ‘squared’ format. Beyond accommodating leaf morphology, this deliberate resizing also expedited the processing time and ensured computational efficiency.

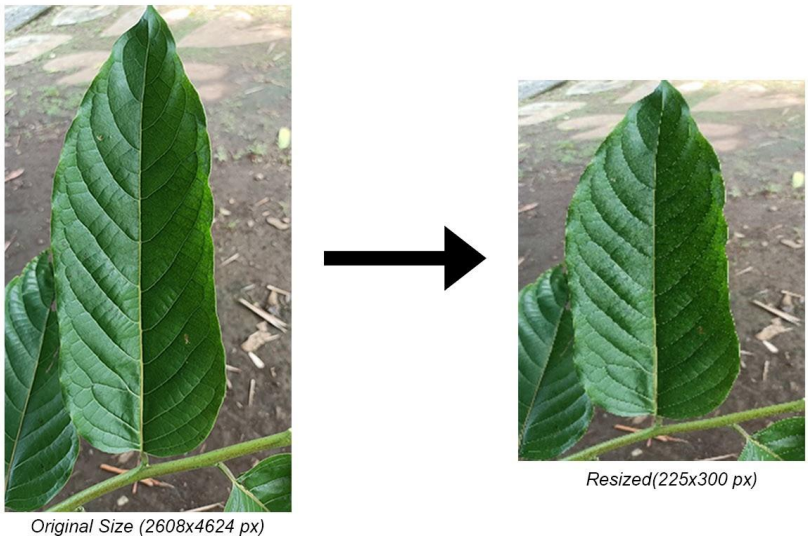

Figure 3. Image resizing of Ilang-ilang

Data Augmentation

In the data augmentation phase, the team implemented various transformations using TensorFlow's Keras library, specifically the ImageDataGenerator, to enrich the dataset utilized for training. This strategic process was crucial for enhancing the model's robustness and adaptability to variations in input images. To elaborate, the images underwent augmentation through diverse techniques, including rotation, width and height shifting, zooming, and horizontal flipping. These transformations introduced an additional layer of diversity into the dataset, effectively multiplying the effective size of the training data. Notably, the team observed optimal model performance when employing a fill mode set to 'nearest' during the augmentation process as seen in Figure 5 below.

The team configured a batch size of 32 for efficient processing during both training and testing phases. The batches were shuffled, enhancing the diversity introduced by data augmentation and preventing the model from learning the order of images. Additionally, a ‘categorical’ class mode was employed, indicating that the labels were one-hot encoded for multi-class classification. This meticulous approach to data augmentation not only multiplied the effective size of the training dataset but also played a pivotal role in boosting the model's performance and generalization capabilities across various input scenarios.

Figure 4. Augmented images

Training and Building the Model

This study uses the pre-trained VGG-16, ResNet50, Inception V3, and Xception architectures to develop a Leaf Classification system through transfer learning to identify the most suitable model among said architectures for the specific task. The researchers accomplished this phase by feeding the training dataset onto the defined architectures and replacing the last fully connected layer and softmax of each, therefore allowing the networks to learn only the features of the leaves dataset the researchers have gathered. Table 2 below shows the configuration used in this study to train the CNN architectures on a Jupyter Notebook inside Visual Studio Code (VS Code) with the use of a dedicated GPU.

The researchers chose a learning rate of 0.001 and the RMSprop optimization algorithm for training due to their proven effectiveness in optimizing neural networks. A learning rate of 0.001 is a commonly used value that strikes a balance between convergence speed and stability

RMSprop is selected for its adaptive learning rate capabilities, which help overcome challenges such as vanishing or exploding gradients. This is supported by Zhu et al. (2021), who emphasize that the choice of the optimizer is crucial in the training process. RMSprop, in particular, combines the exponential moving average of the gradient square to adjust the learning rate. This combination of a moderate learning rate and RMSprop is known to promote stable and efficient training, making it a reliable choice for neural network optimization in various tasks, including our specific use case.

Table 2. CNN architectures' configuration settings per parameter

Training Parameter	Configuration Setting
Batch Size	32
Max Epoch	50
Learning Rate	0.001
Shuffle	True
Training Function	RMSprop

Model Saving and Loading

To ensure that the trained model and its learned weights could be used for future deployments or evaluations, a method for saving and loading the model was implemented. This process involved storing both the model architecture (in JSON format) and the model weights (in HDF5 format). This capability ensured the practicality and versatility of the developed model beyond the research phase.

Model Implementation in the Web Application

In the final step of implementing the model into the web application, the goal was to predict the most likely classes for a given image. The following process was used to make predictions and present them in a user-friendly format:

1. Predicting Class Probabilities: The web app utilized the loaded model (__model) to predict the probabilities for each class that an input image could belong to. This involved feeding the preprocessed image into the model and obtaining an array of probabilities for all possible classes.

2. Selecting the Top 3 Predicted Classes: From the array of class probabilities, the app identified the top 3 classes with the highest probabilities. This allowed for a concise and informative prediction result.

3. Determining Class Names and Probabilities: The app mapped the indices of the top 3 predicted classes to their corresponding class names and probability scores. This information was then structured for presentation.

4. Result Formatting: The prediction result was structured as a dictionary containing the following elements:

- ‘prediction’: The single class label with the highest probability, serving as the primary prediction.
- ‘class_probabilities’: The probability scores associated with the top 3 predicted classes, expressed as percentages.
- ‘class_name_probabilities’: The names of the top 3 predicted classes.

5. Final Output: The result dictionary was organized as a list and printed, making it ready for response through the web app. This concise format ensured that the user received clear and informative predictions.

This implementation was essential for making the model accessible and user-friendly within the web application, providing users with valuable insights and predictions for the input images.

Using the web application

The web application uses a pickle file that contains the created model, scaler, and dictionary for classification of the leaves which means that the server should be running so that the web application can handle responses from the user and make a prediction. As mentioned, the model recognizes Palawan Cherry *(balayong)*, Guava tree *(bayabas*), Betis tree**,** Dao tree**,** Blackboard tree (*dita*), Soursop tree *(Guyabano)*, Ilang-ilang tree, White lead tree *(Ipil)*, Sandpaper tree *(Kalios)*, Ironwood tree *(Kamagong)*, Molave tree *(Mulawin)*, Philippine mahogany *(Narra)*, Mandarin orange tree *(Sinturis)*, and Lauan tree *(Yakal)* – which are the images trained for the model and are the contents of the dictionary. To visualize how the data is being processed within the web application, the data flow diagram is shown below.

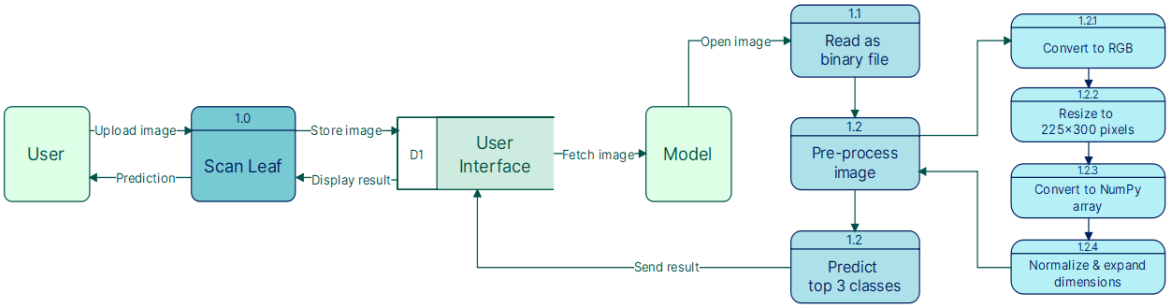

Figure 5. Data flow diagram for Leaflet

Based on this diagram, the user, an external entity, will upload an image that will be directed to the process of scanning the leaf, which involves storing the image temporarily to the user interface (UI) of the web application. Now that the UI holds the image, it will then be fetched by the model to open and read as a binary file suitable for preprocessing. Under the pre-processing are the subprocesses which includes RGB conversion, image resizing, conversion to NumPy array, and normalization. The pre-processing will be followed by predicting the top 3 classes, and this result will be sent back to the user interface so that the user can see the result for the prediction.

Deploying the system to the server

Upon constructing and seamlessly integrating the machine learning model into the web application, the deployment process involved transferring all necessary files, including the frontend components, the model itself, and its associated weights, to the designated physical server. The backend logic is implemented using the Flask framework, which manages server-side operations and facilitates communication between the frontend and the machine learning model. To ensure continuous server operation, the Python script responsible for running the Flask server was executed in the background using the Unix command ‘nohup’ (No Hang Up), preventing termination when the terminal session concludes. This approach ensures the persistent availability and reliability of the web application by allowing the Flask server to run independently of the terminal or user interactions.

Development phase

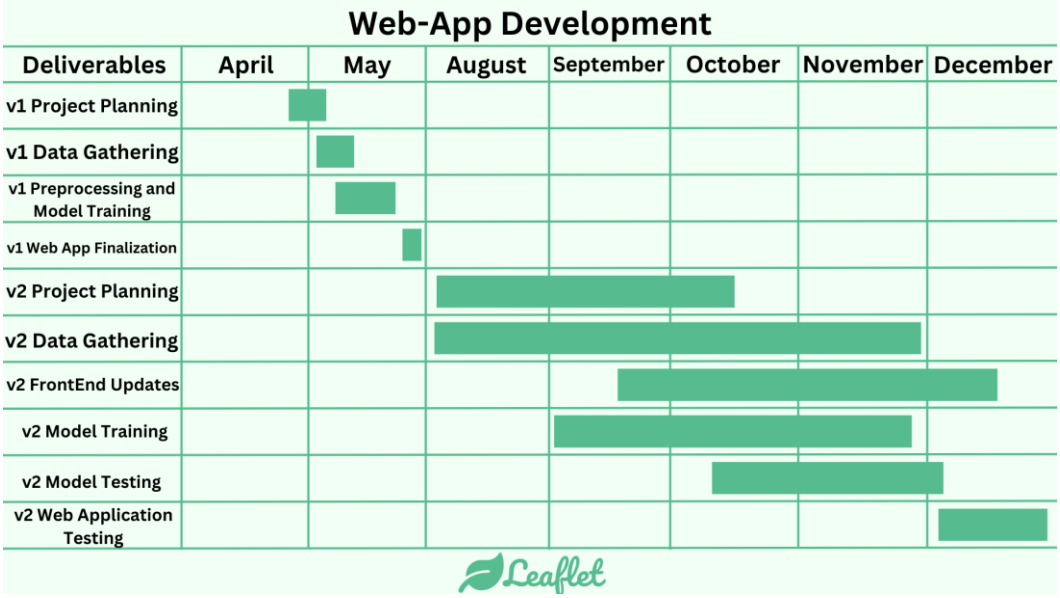

Figure 6. Gantt Chart for Leaflet

The figure above shows the progression of the application's development. The Gantt Chart highlights all the steps and deliverables that the development team has undertaken for the whole development phase of the project. Transpiring from the month of April 2023 to December 2023, the project development took a total of half a year in total and had different versions in it. With each version and implementation, the web-application saw great improvements both in its User Interface, Model, and Server Implementation.

IV. RESULTS AND DISCUSSION

Architecture Comparison

Upon training the four (4) CNN architectures mentioned above, Table 3 shows the accuracy of each upon being trained on nine (9) trees only.

Table 3. Model Comparison

Model	Accuracy
VGG-16	94%
ResNet50	89%
Inception V3	88%
Xception	91%
Sequential	80%

It can be seen in the table that the best performing CNN model among them is VGG-16 with 94% accuracy rate, exceeding the 90% expected mentioned in the objective of this study. Hence, the said model was the one optimized and was used to train five (5) more trees, making it capable of detecting 14 trees in total. Figure 9 shows the finalized architecture of the VGG-16 model utilized by the researchers in developing the model for the web application.

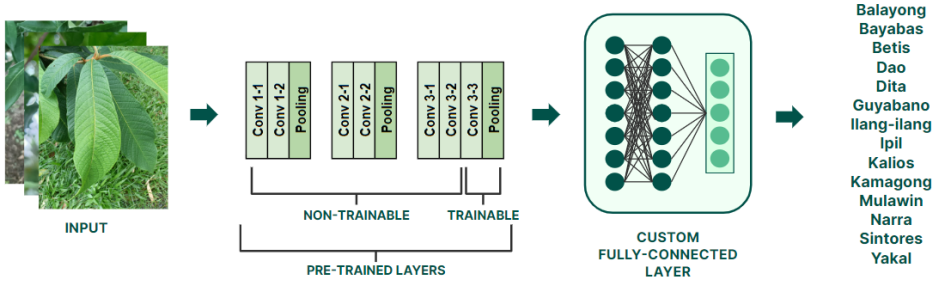

Figure 7. VGG-16 Architecture

On the other hand, the figure below presents the results of a tree classification process using the VGG-16 model trained on a dataset of 14 different tree classes. This report details the model's performance, including metrics like precision, recall, and F1-score for each class. By providing insights into how well the model distinguishes between categories, this report acts as a valuable assessment tool.

	precision	recall	f1-score	support
Balayong	0.97	0.90	0.94	40
Bayabas	0.95	0.93	0.94	40
Betis	1.00	1.00	1.00	40
Dao	0.95	0.97	0.96	40
Dita	0.93	0.95	0.94	40
Guyabano	0.93	0.95	0.94	40
Ilang-Ilang	0.88	0.93	0.90	40
Ipil	0.93	0.97	0.95	40
Kalios	0.93	0.97	0.95	40
Kamagong	0.93	1.00	0.96	40
Mulawin	0.82	0.90	0.86	40
Narra	0.97	0.82	0.89	40
Sintores	1.00	0.82	0.90	40
Yakal	0.98	1.00	0.99	40
accuracy			0.94	560
macro avg	0.94	0.94	0.94	560
weighted avg	0.94	0.94	0.94	560

Figure 8. VGG-16 Classification Report

The classification report reveals the strengths and weaknesses of the trained model for different tree classes. Betis stands out with a perfect F1-score of 1.0, showcasing impeccable accuracy. Conversely, Mulawin struggles with an F1-score of 0.86, indicating room for improvement. This detailed breakdown serves as a valuable tool for optimizing the model's performance across all tree classes. Moreover, the VGG-16 model capable of detecting 14 tree classes garnered an overall score of 94%.

Class Probabilities

After running the image through the model for prediction, the web application would fetch the top 3 highest predictions along with their respective probabilities and names before serializing the result in a JSON format. The formatted result will then be retrieved by JavaScript as a text string through the response of the initially declared XMLHttpRequest. From there, the JavaScript will change the inner HTML of the placeholder text into the actual result of the prediction. The figure below shows the modification of the inner HTML texts from placeholder to actual results, including the probabilities and names of the classes detected by the model.

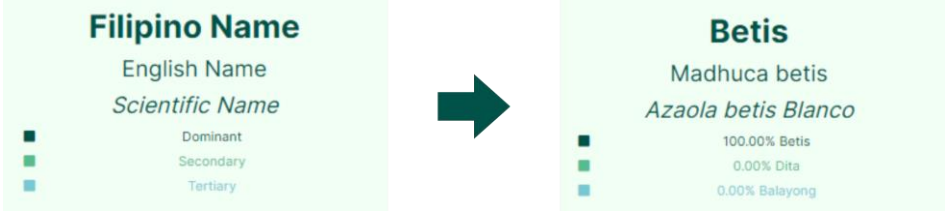

Figure 9. From placeholder to results in Leaflet Frontend

Web Application

The researchers thought of using the web application, developed using HTML, CSS, JavaScript, and Bootstrap, so that the users will be able to use the model with ease using any device, anywhere as long as they have an internet connection. The overall color scheme used was green so that it will stick to the main subject of the development of the project which are trees and leaves. Figure 7 below shows the desktop and mobile view of the web application. The web application has three (3) modules: the first one is the *Scan*, which includes the main feature of the application, and where the leaf classification happens; the second is the *Dictionary*, which includes the information about the application and the leaves that it can classify; and lastly, the *About*, which includes the details about the developers of the web application.

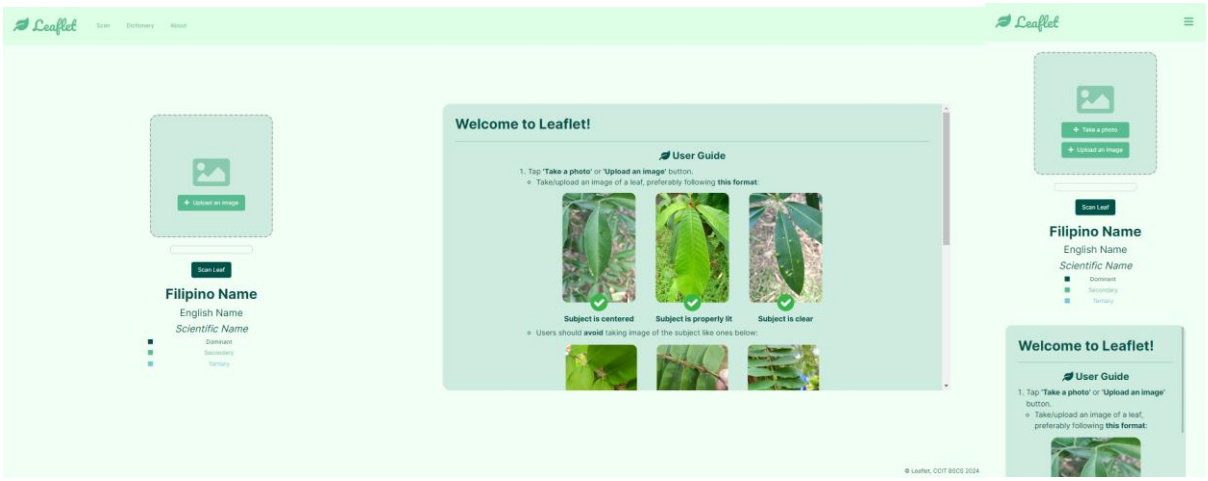

Figure 10. Leaflet web application

To fully understand the flow and behavior of the application, figure 8 below shows the activity diagram of the application. This will describe what should happen in the system for easier intake.

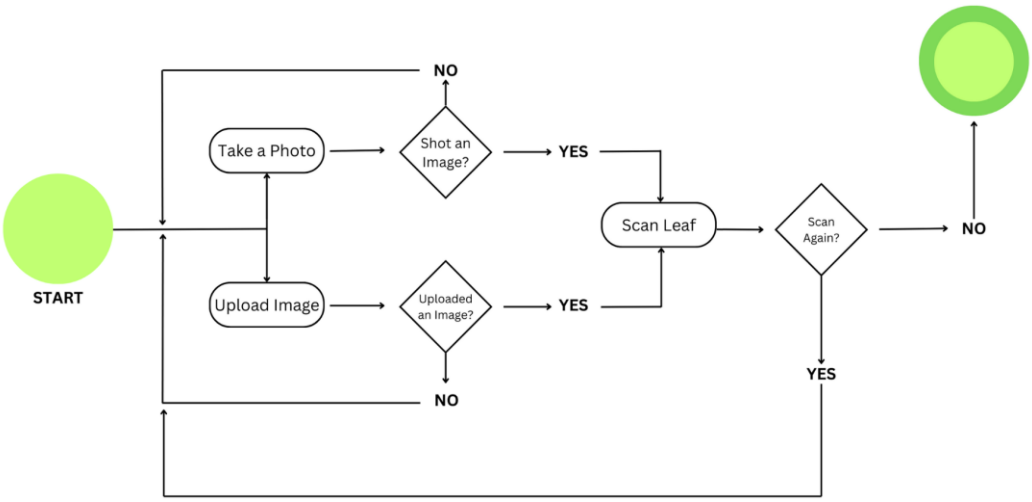

Figure 11. Activity diagram for Leaflet

Based on the diagram above, the user of the application will be starting from the start node and will be met with two choices depending on their device at hand. If a user is using a phone to access and utilize the web-application the user can either choose to take a photo with their device or access their file manager to upload an image to the scanner. If a user is using a PC or Laptop to access the web-application, they are only given the choice of uploading an image via their file manager into the scanner. If by some chance the user decides to cancel the upload or capture step, the user will return to the homepage of the application, hence starting from the very start. After successfully taking an image or uploading an image into the application, the application's scanner will examine the image until it can output a top 3 prediction of the leaf that it believes to be the answer. After this process, users can either choose to leave the application, or scan another leaf or tree that the user desires to scan.

The “Scan Again” Button will be available during this time and once clicked, the user will be returned to the start, restarting the web-application’s process.

V. LIMITATIONS AND DELIMITATIONS

This study only includes classifying the trees that can be seen inside the Serenity and Labyrinth Garden of FAITH Colleges, any other trees beyond the said area will not be included due to time and resources constraints faced by the researchers. Some easily identifiable trees like Papaya and Bamboo Trees weren’t included in the model since individuals, even without the help of the web application, can already identify the said trees. Additionally, this study assumes a controlled environment, minimizing the impact of external factors like weather conditions or seasonal changes on leaf appearance.

When using the application, the result of the model is rather sensitive to the image quality uploaded by the user, hence, lighting conditions, resolution, and background will affect the result of the prediction. With that said, the classification of the leaves should be done ideally within the said area to ensure a higher probability of correct predictions.

Web Application Diagrams

This subsection will include the diagrams related to Leaflet, discussing the overall or/and specific process of the system.

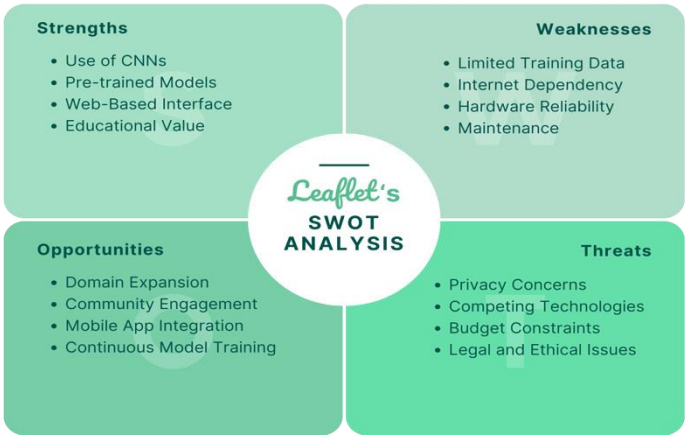

Figure 12. SWOT analysis for Leaflet

Leaflet, a web-based leaf classification application, capitalizes on Convolutional Neural Networks (CNNs) for efficient feature extraction, ensuring accurate leaf identification. The use of pre-trained models minimizes the need for extensive training data and accelerates model development. The web interface provides remote accessibility, specifically catering to the classification of leaves in a school garden, enhancing accuracy for this specific domain. While the project offers educational value, potential weaknesses include hardware reliability, internet dependency, and challenges in interpreting model decisions. Opportunities for Leaflet include collaboration with educational institutions, expanding to diverse domains, community engagement, and developing a mobile app version for enhanced accessibility. Threats encompass privacy concerns, competition from alternative technologies, budget constraints, and the necessity to navigate legal and ethical issues related to data protection and AI ethics. Success lies in addressing these aspects, ensuring regular maintenance, and adapting to evolving technologies and user needs.

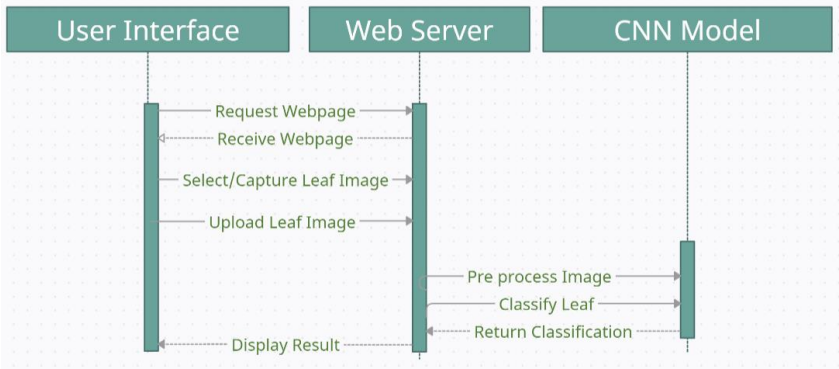

Figure 13. Sequence diagram for Leaflet

The diagram above represents the flow of interactions between three components: User Interface, Web Server, and CNN Model. The sequence of events starts with the user requesting a webpage from the web server. The web server then sends the webpage to the user interface. The user interface, in turn, selects or captures a leaf image and uploads it to the webserver. The webserver preprocesses the uploaded leaf image and sends it to the CNN Model. The CNN Model then classifies the leaf using the preprocessed image and returns the classification result to the web server. Finally, the web server sends the classification result to the user interface, which displays the result to the user.

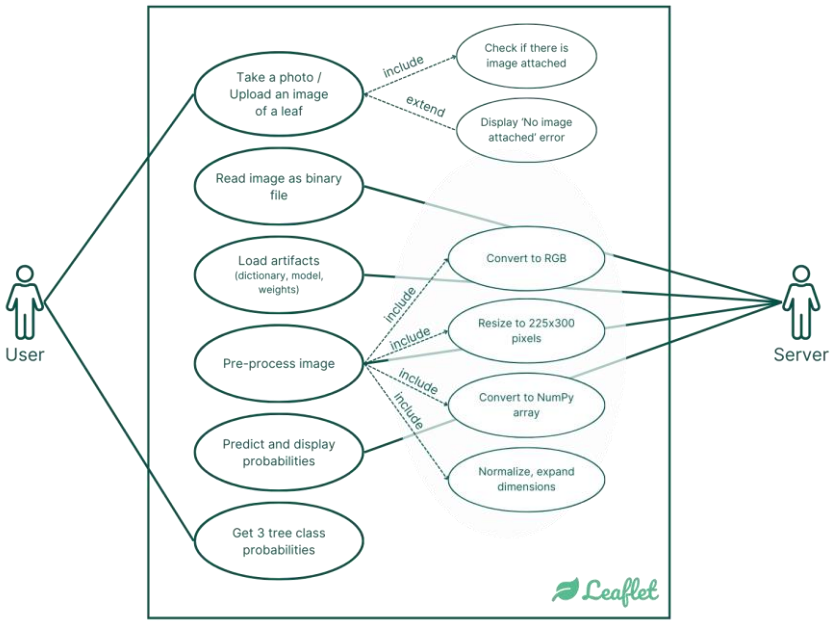

Figure 14. Use case diagram for Leaflet

In the leaf classification web application developed using the VGG-16 model, the user, as the primary actor, initiates the process by either taking a photo or uploading an image of a leaf. The system checks if an image is attached, and if not, it displays a 'No image attached' error. Simultaneously, the server, acting as the secondary actor, reads the uploaded image as a binary file and loads essential artifacts such as a dictionary, model, and weights for leaf classification. The server then undertakes a series of pre-processing steps on the image, including conversion to RGB, resizing to 225x300 pixels, conversion to a NumPy array, and normalization with expanded dimensions to align with the VGG-16 model requirements. Subsequently, the server predicts the class probabilities for the leaf image and presents the user with the top 3 probabilities. This use case diagram elegantly captures the interaction between the user and server in the seamless execution of the leaf classification process.

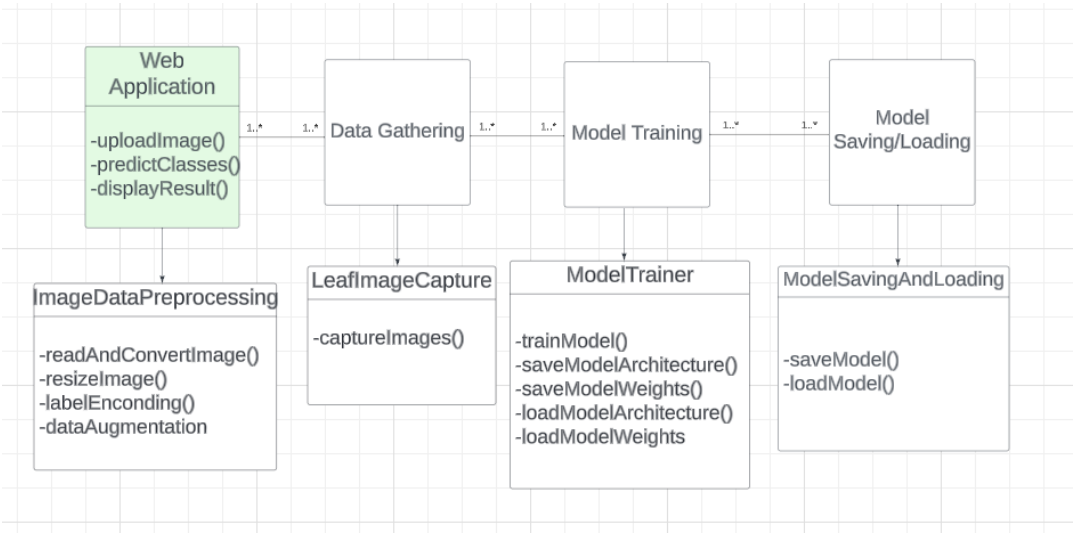

Figure 15. Class diagram for Leaflet

The diagram above presents the architecture of a system for image classification. The system consists of four main components interconnected with each other. The first component is the Web Application, which enables users to upload images, predict classes, and display the results, which is connected to the Image Data Preprocessing sub-component, responsible for performing tasks such as reading and converting images, resizing images, and more. The second component is Data Gathering, which includes the Leaf Image Capture sub-component that is responsible for capturing images. The third component is Model Training, which includes the Model Trainer sub-component that is responsible for training the machine learning model, also, includes saving and loading the model architecture and weights. The fourth component is Model Saving/Loading, which includes the Model Saving and Model Loading sub-components which are responsible for saving the trained model and loading a saved model, respectively. Overall, this system architecture provides a structured approach to image classification, allowing users to upload images, preprocess them, train a model, and save and load the trained model for future use.

VI. CONCLUSIONS AND RECOMMENDATIONS

Based on the findings of the study, the multiple conclusions were drawn. The web application employs the VGG-16 model as its core neural network, chosen for its exceptional accuracy, surpassing the other four models utilized and trained in the system. This choice ensures optimal performance and reliable results in handling the application's main task. Additionally, the web application allows its users to scan the leaves of 14 trees from FAITH's Serenity and Labyrinth Garden. Scanning leaves beyond the said area will not guarantee a high accuracy score. Users can verify the leaves scanned through the image attached in the result container which also includes the details of the leaves scanned.

Users are provided with a user guide for the image taking process of scanning the leaves. It allows its users to capture the leaves in any way that they want to capture the image in. However, users will be met with a lower accuracy output and prediction if the users will not follow the said guide and instructions that was provided. After scanning, the application's performance will be mainly affected by the user's device and internet connection. Mobile Data usage and WiFi Connectivity will each have different processing time for the web application to predict the scanned leaf. In addition, the quality of the user's uploaded image will affect the performance of the model due to the controlled environment that the researchers' used.

The researchers would like to recommend that the future researchers will use more images when training to ensure a higher diversity in dataset and minimize the risk of overfitting, resulting in a more accurate classification model. On top of that, the device used in training plays a huge role in the speed of building the model, hence, it is recommended that, if possible, a desktop computer with higher Graphics Processing Unit (GPU) should be used. Laptops could be used but be reminded that this could pose a huge risk on the said device, based on the researchers' experience.

REFERENCES

[1] Ahmed. (2022, October 18). *The Motivation for Train-Test Split*. Retrieved from https://medium.com/@nahmed3536/the-motivation-for-train-test-split-2b1837f596c3

[2] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O. Santamaria, J., Fadhel, M. A., Al-Amidie, M., and Farhan, L. (2021, March). "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *J. Big Data*, vol. 8, no. 1, p. 53.

[3] Azlah, M. A. F., Chua, L. S., Rahmad, F. R., Abdullah, F. I., & Wan Alwi, S. R. (2019). Review on Techniques for Plant Leaf Classification and Recognition. Computers, 8(4), 77. doi:10.3390/computers8040077

[4] Collins, A.S. (2019, December 17). *How the Sobel Operator Works.* Retrieved from https://automaticaddison.com/how-the-sobel-operator-works/

[5] Copeland, B.J. (2023, October 11). Artificial intelligence. Retrieved from https://www.britannica.com/technology/artificial-intelligence

[6] Dabbadie, L. (2023, July 17). *The revival of a damaged Philippines watershed is helping improve nutrition and livelihoods of communities.* Retrieved from https://www.fao.org/news/countries-good-practices/article/en/c/1644557/

[7] de Jesus, A. (2021, March 6). *Why we need to plant native Philippine trees.* Retrieved from https://business.inquirer.net/319013/why-we-need-to-plant-native-phi lippine-trees

[8] Fan, X., Zhou, R., Tjahjadi, T. Choudhury, S. D. & Ye, Q. (2022). A Segmentation-Guided

[9] Deep Learning Framework for Leaf Counting. *Frontiers in Plant Science, 13*(1), 1 - 13. https://doi.org/10.3389/fpls.2022.844522

[10] Huellman, T. (2022, November 16). *Image processing: How do image classifiers work?* levity.ai. Retrieved October 20, 2023, from https://levity.ai/blog/how-do-ima ge-classifiers-work

[11] Keivani, M., Mazloum, J., Sedaghatfar, E., & Tavakoli, M. B. (2020). Automated Analysis of Leaf Shape, Texture, and Color Features for Plant Classification. https://www.academia.edu/download/88721176/25544.pdf

[12] Nidhi & Yadav, J. (2020, July 26). Plant Leaf Classification Using Convolutional Neural Network. *Recent Advances in Computer Science and Communications, 15*(3), 421. https://doi.org/10.2174/2666255813999200904162029

[13] Rizk, S. (2019). *Plant Leaf Classification Using Dual Path Convolutional Neural Networks* [Master's thesis, Notre Dame University-Louaize]. Notre Dame University-Louaize Institutional Repository.

[14] Rouse, M. (2023, October 11). Deep Neural Network. Retrieved from https://www.techopedia.com/definition/32902/deep-neural-network

[15] Saha, S. (2023, April 8). A Comprehensive Guide to Convolutional Neural Networks —the ELI5 way. Retrieved from https://towardsdatascience.com/a-com prehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[16] Saleem, G., Akhtar, M., Ahmed, N., & Qureshi, W. S. (2019). Automated analysis of visual leaf shape features for plant classification. Computers and Electronics in Agriculture, 157, 270–280. doi:10.1016/j.compag.2018.12.038

[17] Urwin, M. (2023, May 24). 20 Deep Learning Applications You Should Know. Retrieved from https://builtin.com/artificial-intelligence/deep-learning-applications

[18] Turkoglu, M. & Hanbay, D. (2019). Recognition of plant leaves: An approach with hybrid features produced by dividing leaf images into two and four parts. Applied Mathematics and Computation, 352, 1-14. https://doi.org/10.1016/j.amc.2019.01.054

[19] Wang, B., & Wang, D. (2019). Plant Leaves Classification: A Few-Shot Learning Method Based on Siamese Network. IEEE Access, 7, 151754–151763. doi:10.1109/access.2019.2947510

[20] Yang, K., Zhong W. & Li , F. (2020). Leaf Segmentation and Classification with a Complicated Background Using Deep Learning. *Agronomy, 10*(11), 1721. https://doi.org/10.3390/agronomy10111721

[21] Zhuang, F. (2019). "A Comprehensive Survey on Transfer Learning"

[22] Zhu, Y., Li, G., Wang, R., Tang, S., Su, H., & Cao, K. (2021). Intelligent fault diagnosis of hydraulic piston pump based on wavelet analysis and improved AlexNet. Sensors, 21(2), 549.