University
Mohammed VI
Polytechnic

ROYAUME DU MAROC

--*-*-*-*-*-*-*

MOHAMMED VI POLYTECHNIC UNIVERSITY

--*-*-*-*-*-*

Technical Test

Data Science Engineer

--*-*-*-*-*-*

Author :

Pierjos Francis COLERE MBOUKOU

Pierjos.COLERE@um6p.ma

March, 3rd 2024

In this project, we present our approach to the classification of healthy and diseased trees. To do this, we use the Cherry Disease Detection Dataset, a multi-modal, multi-angular dataset that was created to monitor cherry tree growth, including stress analysis and prediction. The dataset encompasses 577 cherry trees observed throughout a full crop season, spanning from July 2021 to July 2022. First, we processed the data by applying a few transformations.

Next, we construct different unimodal architectures and compare them with each other. For each modality, we select the best architecture in terms of test performance. We then use these to build different multimodal architectures. This project led to the adoption of an API deployment architecture for the best model resulting from our experimentation.

I. Data Collection and Restructuring

Firstly, we commence by gathering data from the designated source [1], and subsequently, we store it onto Google Drive. Following the storage phase, we proceed to restructure the data, aiming to enhance its accessibility for the subsequent stages of our project. This restructuring process involves refining the original dataset structure to optimize its utility and efficiency for our analytical tasks.

```
Cherry Tree Disease Detection Dataset
|   Readme.docx
|   Tree_Mapping.xls
|
+---Date
|   |   date.csv
|   |
|   +---Aerial_UAV_photos
|   |   |       Image01
|   |   |       Image02
|   |   |         <…>
|   |   |
|   +---Ground_RGB_Photos
|   |   +---Armillaria_Stage_1
|   |   |       Image01
|   |   |       Image02
|   |   |         <…>
|   |   +---Armillaria_Stage_2
|   |   |       Image01
|   |   |       Image02
|   |   |         <…>
|   |   +---Armillaria_Stage_3
|   |   |       Image01
|   |   |       Image02
|   |   |         <…>
|   |   \---Healthy
|   |   |       Image01
|   |   |       Image02
|   |   |         <…>
|   |   |
|   \---Ground_Multispectral_Photos
|       +---Armillaria_Stage_1
|       |   +---Folder01
|       |   |       Image01
|       |   |       Image01
|       |   |       MetadataFile01
|       |   |         <…>
|
```

Now, it turns out that within the Ground Multispectral and UAV modalities, there are sub-modalities, namely:

- ⇨ **UAV**: NDVI, Green, NIR, RED, REDEDGE.
- ⇨ **Multispectral**: Green, NIR, RED, REDEDGE, RGB.

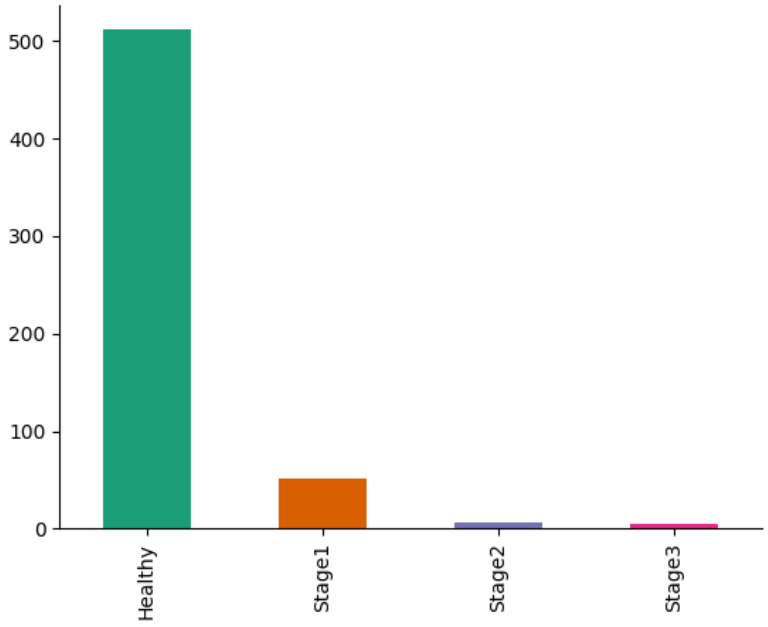
Hence, it is important to distinguish them and place them in separate folders. To achieve this, we have created a Python function called ***read_and_format_data***.

II. Data Visualization

1. Distribution of Modalities

In the image below, we can observe the distribution of different classes across the dataset collected on 05/26/2022. An important observation is that the vast majority of trees in this dataset show no signs of Armillaria infection, as they are classified as "Healthy" (89.02%). Approximately 8.88% of trees are at stage 1, indicating symptoms or early signs of Armillaria infection. Trees at stage 2 represent only a small proportion, approximately 1.22%.

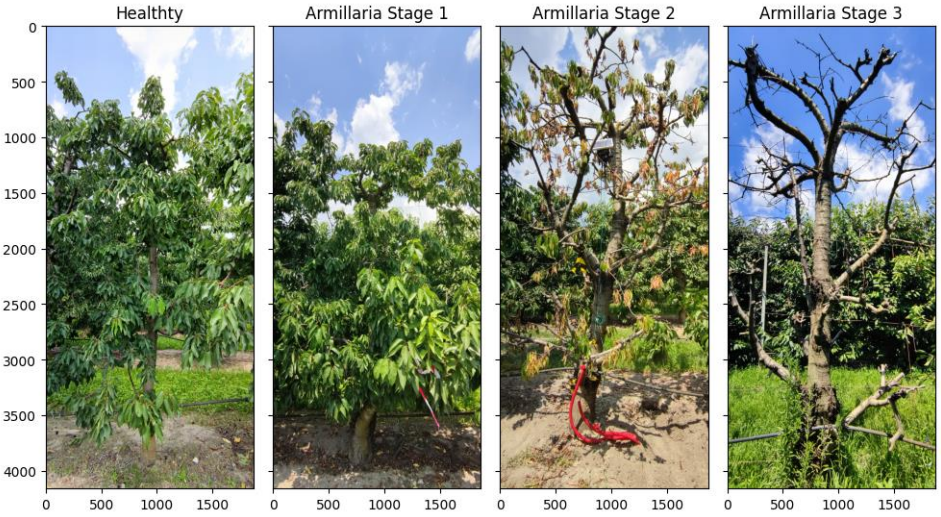
However, they exhibit more pronounced symptoms or a more advanced progression of Armillaria compared to stage 1 trees. The smallest group consists of trees at stage 3, which are in an advanced stage of Armillaria infection. These trees display severe symptoms such as total leaf loss, as we will further examine in the subsequent analysis.



2. Image Visualization

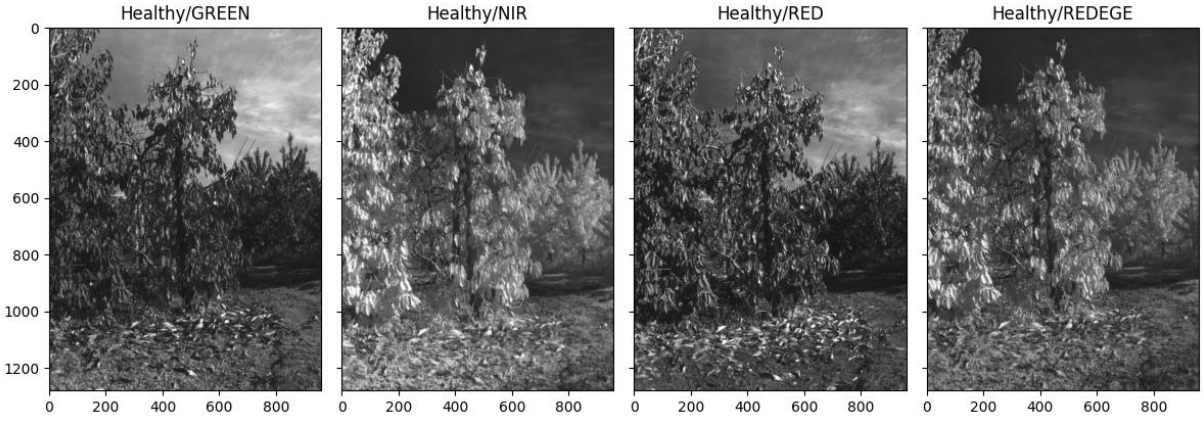
a. RGB

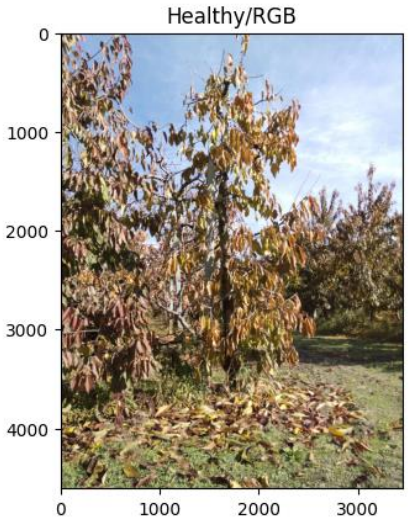
It is noteworthy that as the tree is more severely affected by Armillaria at an advanced stage, it exhibits significant leaf loss. However, visually distinguishing between a healthy tree and one at stage 1 of infection becomes challenging. An advanced hypothesis suggests that the presence of leaves on the tree plays a crucial role in evaluating the presence or absence of Armillaria.



b. Multispectral

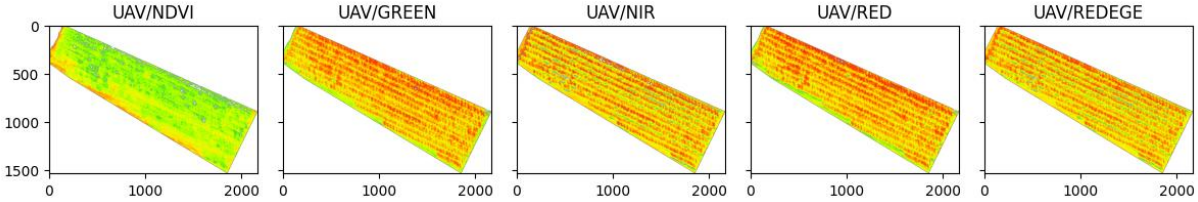
Here is a set of images illustrating the various sub-modalities of Multispectral. These images are presented below to allow you to observe the different facets and variations of the spectrum used in this technology. Each image represents a specific sub-modality, thereby providing a detailed view of the specific features captured by the Multispectral system.





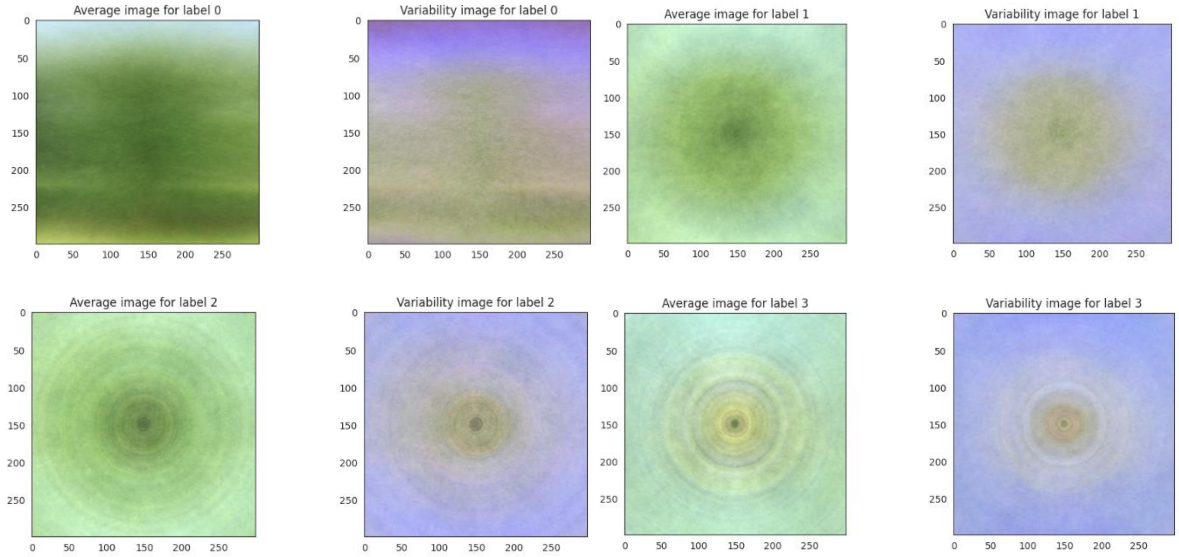
c. UAV

Below is a visual representation of the sub-modalities of UAV remote sensing. It can be observed that in the Normalized Difference Vegetation Index (NDVI), the green color is dominant compared to the other sub-modalities, which predominantly exhibit red tones.

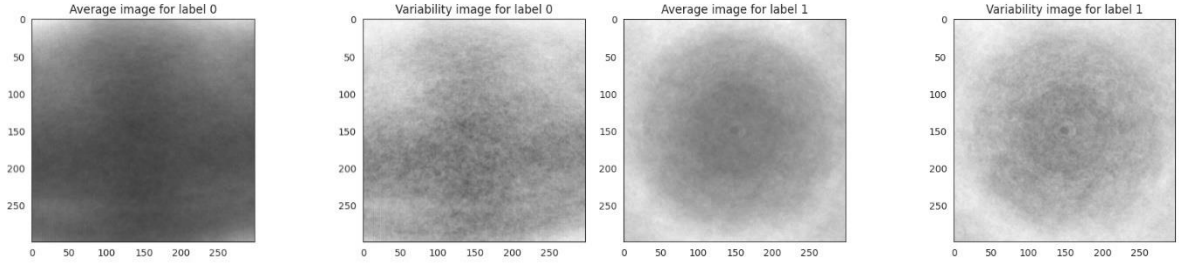


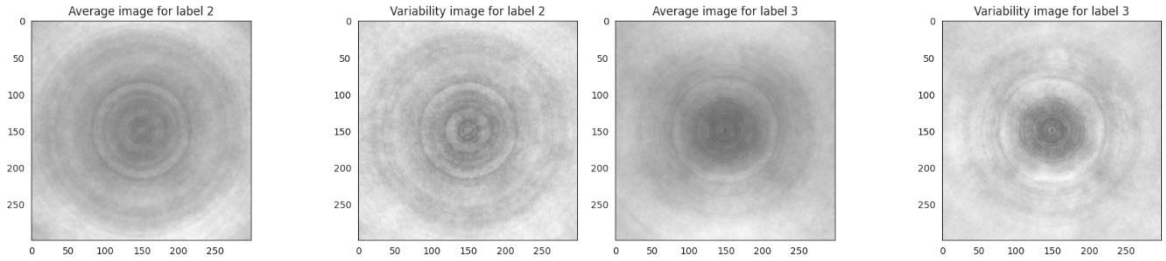
3. Mean and Standard Deviation Analysis

As part of our exploratory study, we aim to examine the behavior of pixels in terms of mean and standard deviation for each class. When we compute these two measures, we observe a correlation between the extent of Armillaria infection on the tree and a circular effect on the mean and dispersion (standard deviation) images at the center of the mean image. These conclusions apply to both RGB modalities and multispectral data, as highlighted in the following visualizations.



Average and variability images for Ground RGB





Average and variability images for Multispectral: Green Case

These findings underline the challenge in discerning healthy trees from those at stage 1 of infection, as well as differentiating between trees at stage 2 and those at stage 3. Despite efforts to analyze mean and standard deviation metrics, the overlapping characteristics among these stages hinder clear visual classification. This ambiguity suggests the need for further refinement in our analysis techniques or the exploration of additional features to enhance classification accuracy.

III. Data Preprocessing

In order to successfully carry out this project and achieve good results in modeling, it is important to follow the following steps to meet the requirements of a Data Science project.

1. Data Joining

By utilizing the data from the **"*date.csv*"** file, we construct the following datasets for each modality by adding certain columns such as tree, level, and URL (where the corresponding image for this tree is stored).

`rgb_df.head()`

	tree	modality	level	url
0	10-1	rgb	Stage1	/content/drive/My Drive/DataScienceEngineer/26...
1	10-3	rgb	Stage1	/content/drive/My Drive/DataScienceEngineer/26...
2	2-17	rgb	Stage1	/content/drive/My Drive/DataScienceEngineer/26...
3	2-25	rgb	Stage1	/content/drive/My Drive/DataScienceEngineer/26...
4	3-26	rgb	Stage1	/content/drive/My Drive/DataScienceEngineer/26...

`uav_df.head()`

	modality	submodality	url
0	uav	ndvi	/content/drive/My Drive/DataScienceEngineer/26...
1	uav	green	/content/drive/My Drive/DataScienceEngineer/26...
2	uav	nir	/content/drive/My Drive/DataScienceEngineer/26...
3	uav	red	/content/drive/My Drive/DataScienceEngineer/26...
4	uav	rededge	/content/drive/My Drive/DataScienceEngineer/26...

`multispectral_df.head()`

	modality	submodality	url	tree	level
0	multispectral	rgb	/content/drive/My Drive/DataScienceEngineer/26...	9-4	Stage1
1	multispectral	rgb	/content/drive/My Drive/DataScienceEngineer/26...	9-38	Stage1
2	multispectral	rgb	/content/drive/My Drive/DataScienceEngineer/26...	8-40	Stage1
3	multispectral	rgb	/content/drive/My Drive/DataScienceEngineer/26...	8-5	Stage1
4	multispectral	rgb	/content/drive/My Drive/DataScienceEngineer/26...	8-57	Stage1

To link RGB and multispectral data, we perform a join operation based on a common identifier, which in this case is the tree. This process involves combining the data from both modalities into a single dataset where each entry corresponds to a specific tree, incorporating relevant columns such as tree ID, RGB features, multispectral features, and additional metadata like tree level and URL.

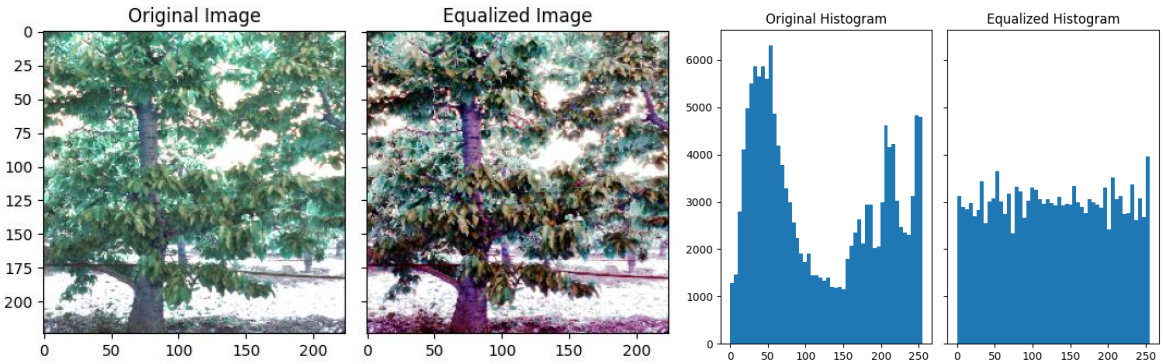
Once the join operation is completed, the resulting dataset is saved onto Google Drive. This ensures that the joined data is readily accessible and can be efficiently utilized in subsequent stages of the project without the need to repeat the costly join operations. By storing the data on Google Drive, it becomes easily shareable among team members and can be accessed from various platforms, facilitating collaborative work on the project. Additionally, this approach helps to streamline the workflow and reduce computational overhead, as the joined data can be directly accessed for further analysis and model development without the need for repetitive data processing steps.

	tree	modality_rgb	level	url_rgb	modality_multispectral	submodality	url_multispectral
0	10-1	rgb	Stage1	/content/drive/My Drive/DataScienceEngineer/26...	multispectral	rgb	/content/drive/My Drive/DataScienceEngineer/26...
1	10-1	rgb	Stage1	/content/drive/My Drive/DataScienceEngineer/26...	multispectral	rededge	/content/drive/My Drive/DataScienceEngineer/26...
2	10-1	rgb	Stage1	/content/drive/My Drive/DataScienceEngineer/26...	multispectral	red	/content/drive/My Drive/DataScienceEngineer/26...
3	10-1	rgb	Stage1	/content/drive/My Drive/DataScienceEngineer/26...	multispectral	nir	/content/drive/My Drive/DataScienceEngineer/26...
4	10-1	rgb	Stage1	/content/drive/My Drive/DataScienceEngineer/26...	multispectral	green	/content/drive/My Drive/DataScienceEngineer/26...

2. Histogram Equalization

Histogram equalization is a technique used to enhance the contrast of an image by adjusting the distribution of pixel intensities. This method redistributes the pixel values in such a way that the histogram of the resulting image becomes more uniformly distributed across the intensity range. By doing so, it effectively stretches out the intensity levels, thereby improving the overall contrast of the image. Before applying histogram equalization, the original image may suffer from poor contrast, with certain areas appearing too dark or too bright. Histogram equalization aims to address this issue by redistributing the pixel values to better utilize the available intensity range, resulting in a more balanced and visually appealing image.

The figures presented below illustrate the effectiveness of histogram equalization by showcasing the original image alongside its histogram. Subsequently, the processed image and its corresponding histogram are displayed, highlighting the improvement in contrast achieved through histogram equalization. This transformation enhances the visibility of features within the image, making them more discernible for further analysis and interpretation.



3. Data Augmentation

As seen earlier, the number of images per class is unevenly distributed. However, utilizing imbalanced data in training a Machine Learning model can lead to poor performance. The model tends to learn predominantly from the majority class, a phenomenon known as "*overfitting*".

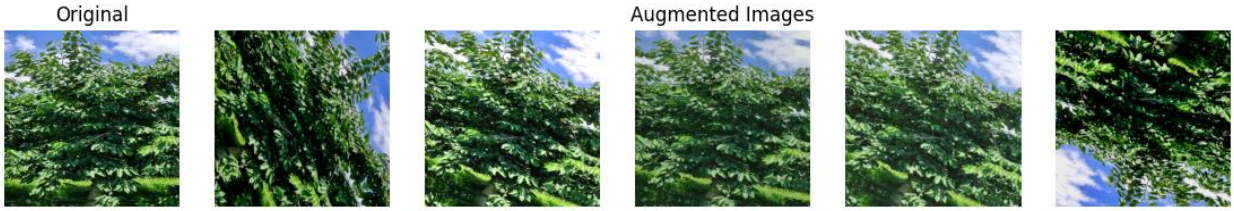
Overfitting occurs when a model becomes overly tailored to the training data, hindering its ability to generalize well to new, unseen data. To mitigate this risk, data augmentation is employed, generating new synthetic training data like the existing data but with variations to provide a more robust representation of underlying patterns.

Addressing the dataset imbalance, we applied oversampling techniques to equalize the dominant class. This involved employing various image modifications like rotations, vertical and horizontal flips, brightness adjustments (both reduction and increase), and cropping. These techniques generated new synthetic data, augmenting the original dataset, and balancing the distribution of images across different classes.

To achieve image balance, we can utilize a method of generating synthetic data based on the number of images in each class. Mathematically, let N be the total number of classes, i be the class with the highest number of images, and N_k be the number of images for class k. The number of images to generate N'_k for class k can be determined by the following equation:

$$N'_k = \frac{(N_i - N_k)}{N_k}, k = 1, \dots, N$$

We applied the same process to all modalities to balance the minority classes. An example illustrating this data augmentation process is as follows:



4. Data Normalization and Rescaling

After augmenting the dataset, we conducted two main preprocessing steps. Initially, we rescaled the intensity values of the images from integers ranging from 0 to 255 to floating-point numbers between 0 and 1. This transformation is crucial as most deep learning architectures require input images with normalized intensity values. By scaling the intensity values to a common range, we ensure consistency and facilitate the learning process of the model.

Since we plan to utilize DenseNet201 as a pre-trained deep convolutional network, which recommends input images in a fixed format of 224x224 pixels, we resized all images to meet this requirement. Resizing the images ensures compatibility with the architecture's input specifications and enables efficient processing during model training and evaluation.

Additionally, due to the substantial size of the augmented dataset, we opted to save the resized data as NumPy (.npy) files. This format offers efficient storage and retrieval capabilities, eliminating the need to repeat data processing steps in future tasks such as model training and evaluation. By storing the preprocessed data in a structured file format, we streamline the workflow and enhance computational efficiency in subsequent stages of the project.

5. Data Splitting

To evaluate our architectures effectively and address the issue of overfitting during training, we adopt a strategy of splitting the dataset into two subsets: one for training and the other for testing

and validating the architecture. Specifically, we allocate 30% of the data for validation and reserve the remaining 70% for training purposes.

In addition to this standard train-test split, we employ cross-validation, specifically K-Fold cross-validation, to ensure robustness and generalizability of the model results to unseen data. This process involves dividing the original dataset into k folds or subsets of equal size. Then, in each iteration, K-1 folds are used for training the model while the remaining fold is used as a validation set to evaluate the model's performance. This process is repeated K times, with each fold serving as the validation set exactly once.

By averaging the performance metrics obtained across all iterations, we obtain a more reliable estimate of the model's performance. In our study, we set K to 5, meaning the dataset is divided into 5 equal parts, and the model is trained and evaluated 5 times, each time with a different fold used as the validation set. This approach allows us to assess the model's performance comprehensively and mitigate the risk of overfitting to the training data.

Once the training phase is completed, we validate our architecture on the test data, which were not seen during the training process. This step is crucial for assessing the model's performance in real-world scenarios and ensuring that it can generalize well to unseen data. By evaluating the model on the test set, we can measure its ability to accurately classify or predict outcomes on new, previously unseen instances. This validation step provides valuable insights into the model's performance and helps validate its effectiveness in practical applications.

6. Model Compilation

- **Loss**: A loss function is a function that compares the target and predicted output values; measures how well the neural network models the training data. When training, we aim to minimize this loss between the predicted and target outputs. Since the problem has been treated as multiclass classification, ***categorical_crossentropy*** (also called Softmax Loss. It is a Softmax activation plus a Cross-Entropy loss) was used.
- **Optimizer**: An optimizer is a function or algorithm that is created and used for neural network attribute modification (i.e., weights, learning rates) for the purpose of speeding up convergence while minimizing loss and maximizing accuracy. ***adam*** was chosen going through the trial-and-error phase.
- **Metrics**: ***accuracy*** Calculates how often predictions equal labels. This metric creates two local variables, total and count that are used to compute the frequency with which ***y_pred*** matches ***y_true***.

7. Data Modeling: Parametrization

In this section, we will model the data to predict whether the tree is healthy or affected by Armillaria, and thus its stage. We will proceed in two phases. First, we will construct unimodal architectures (using a single modality) for each modality. For multispectral data, we will have a total of 5 architectures for each sub-modality, and we will compare them against each other. We will select the one that provides the best performance and construct a multimodal architecture with Ground RGB.

It is important to note that this approach is minimalist in terms of resource utilization (RAM, Disk, Time) and complexity. Depending on the available resources, we could work with all modalities, but we are concerned that all these modalities (aggregation, union) may not significantly contribute to the Armillaria detection task. It's possible that aggregating these modalities might

yield the same performance as using just one modality. In such a case, what would be the benefit of using this aggregation? Moreover, since we are working with images, it is prudent to leverage pre-trained convolutional models. Based on our experience in image processing and Deep Learning, we select DenseNet201, one of the architectures that consistently delivers good results in our studies [2, 3].

To prevent overfitting, we introduce Dropout layers that randomly deactivate 25% of neurons. Additionally, we incorporate **EarlyStopping** and **ReduceLROnPlateau** callbacks, which respectively halt training if the loss value does not significantly change after 10 epochs and adjust the learning rate value after 10 epochs if the loss value remains unchanged. These parameters act as regulators to overcome overfitting that may occur during training. We have also defined other parameters such as EPOCHS set to 100 and BATCH_SIZE set to 32.

IV. Results and Discussion

In this chapter, we present a comprehensive performance evaluation of architecture models used for cervical lesion classification. Our objective is to identify the most efficient architecture among these models and provide an in-depth analysis of their performance.

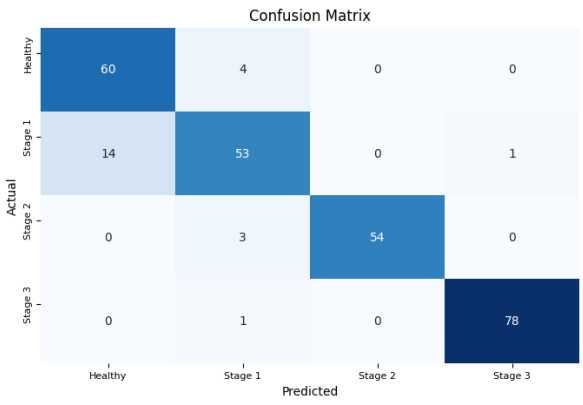
1. Unimodal Architectures

We have chosen DenseNet201 as the base model and added dense connected layers along with the Dropout technique.

1.1. Ground RGB

Regarding the "**Ground RGB**" modality, the confusion matrix below demonstrates the model's satisfactory performance, with promising results. Indeed, the model achieved an accuracy of 91.42% in predicting whether the tree is healthy or affected by Armillaria. This attests to the effectiveness of the model in detecting this disease. Furthermore, it's noteworthy that the model accurately identifies trees at stages 3 and 2 of Armillaria. This underscores the model's ability to make a clear distinction between healthy trees and those heavily infected by the disease.

However, it's important to mention that there were some prediction errors for trees at stage 1 of Armillaria. Some of these trees were incorrectly classified as healthy or already at an advanced stage of the disease. It's worth noting that this can be considered normal, as during the exploratory phase, it was challenging to definitively distinguish between healthy trees and those at stage 1. This difficulty in distinction may explain these few prediction errors for this specific category.



In conclusion, the model based on the "Ground RGB" modality exhibits overall satisfactory performance, with excellent accuracy in detecting trees affected by Armillaria. However, improvements could be made to reduce prediction errors for trees at stage 1 of the disease.

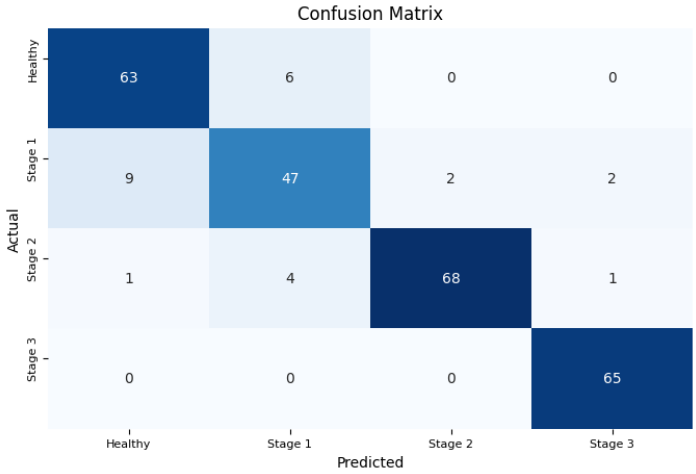
1.2. Ground Multispectral

It is noteworthy that the REDEDGE modality stands out clearly with an accuracy of 90.67%, positioning it at the top of the rankings. The RGB modality secures the second position with an accuracy of 89.02%, while GREEN occupies the third position (85.61%). On the other hand, the NIR and RED modalities are both equivalent with an accuracy of 85.45%, placing them at the bottom of the rankings.

These results suggest that the use of REDEDGE could be advantageous for applications requiring high precision. Additionally, the remaining modalities demonstrate acceptable performance on the test data. Therefore, it would be prudent to limit the use to a single modality rather than multiple modalities, as employing all these modalities would entail considerable complexity without bringing significant differences in results, especially in the case of multispectral data.

	Accuracy (%)	Rank
GREEN	85.61	3
NIR	85.45	4
RED	85.45	4
REDEDGE	90.67	1
RGB	89.02	2

Here is the confusion matrix for the REDEDGE modality, summarizing the prediction results and comparing the actual data to those predicted by the model.

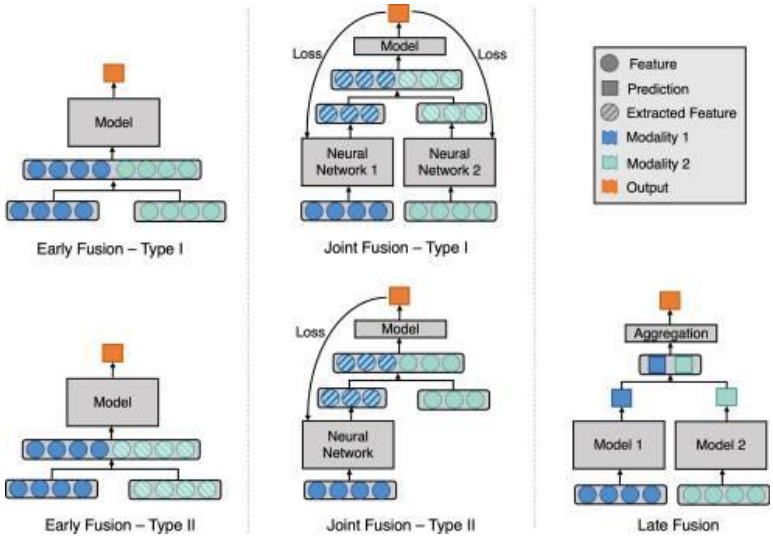


In summary, by focusing on the REDEDGE and Ground RGB modalities, it's possible to achieve better performance while reducing the overall complexity of the task at hand. In this context, it's important to emphasize the need to strike a balance between the desired performance and the complexity associated with using multiple modalities.
In the next steps, we will combine REDEDGE and RGB modalities and compare their performances against these individual modalities. Then, we will attempt to add UAV images, which may introduce noise but could potentially improve overall performance.

2. Multimodal Architectures

In this section, we employ data fusion, which involves assembling data from multiple modalities to extract complementary and more comprehensive information for more powerful machine learning models, as opposed to relying on a single data modality.

The figure below, sourced from [4], illustrates the three main merging strategies: early merging, joint merging, and late merging. We will define and describe each merging strategy in detail here:



Early fusion (left figure) involves concatenating original or extracted features at the input level. Joint fusion (middle figure) also combines input-level features, but the loss is passed back to the feature extraction model. Late fusion (right figure) combines predictions at the decision level.

Early fusion integrates all features from the start, forming a comprehensive multimedia feature representation. It involves a single learning phase but faces the challenge of combining diverse features into a unified representation. Late Fusion is simpler than early fusion when data sources vary significantly. It often performs better due to independent error handling. However, it has a high computation cost and may lead to a loss of correlation in the mixed feature space.

2.1. Late Fusion

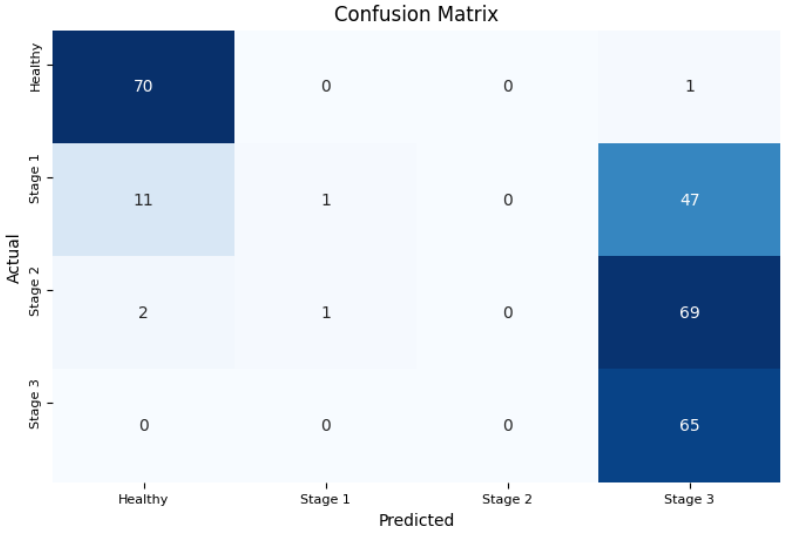
The following results are obtained using the Late Fusion technique between the Ground RGB and REDEDGE (multispectral) modalities, employing the same parameters as for the unimodal modality.

Dataset	Accuracy (%)	Precision (%)	Recall(%)
Training	47.19	78.26	20.22
Test	50.94	85.37	26.22

It is evident that the model has not learned sufficient data, leading to the problem of underfitting, where the performance on the test data is better than on the training data.

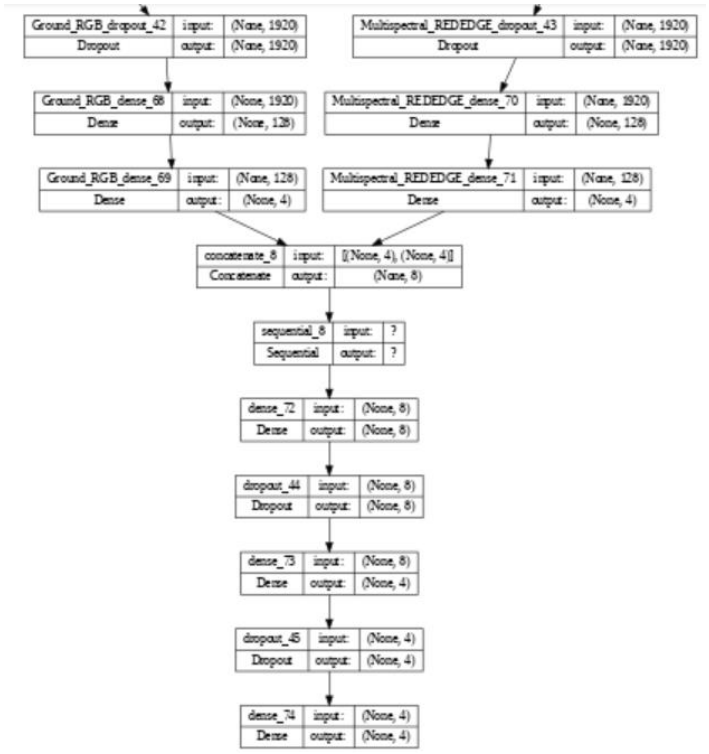
The late fusion technique combines predictions at the decision level, which means that the final predictions are made by combining the outputs of individual models (in this case, the models trained on Ground RGB and REDEDGE modalities). While this approach can yield accurate

results, it may struggle to capture more complex relationships between features, leading to underfitting. As a result, while the model performs well in generalizing between healthy and diseased trees, it may not adequately differentiate between different stages of Armillaria infection. This limitation becomes apparent when examining the confusion matrix, which shows a tendency to classify all diseased trees as stage 3, indicating a potential area for improvement in the model's performance.



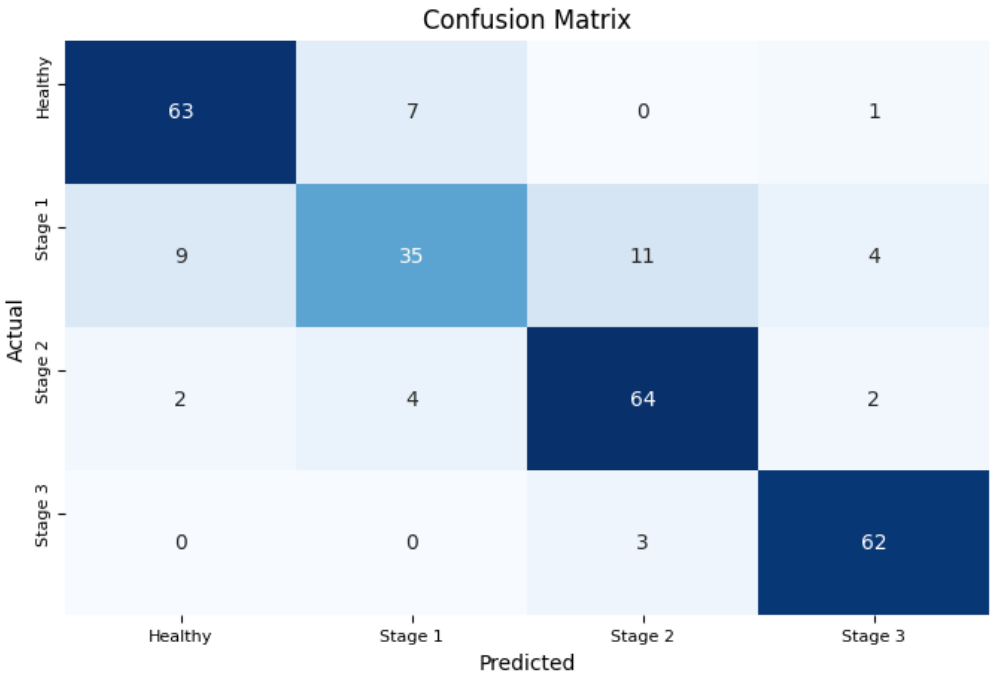
Note: If we focus solely on distinguishing between healthy and diseased trees without considering their stage of development, it could be beneficial.

Here's a summarized overview of the structure (final layers) of the late fusion architecture we used to model our problem.



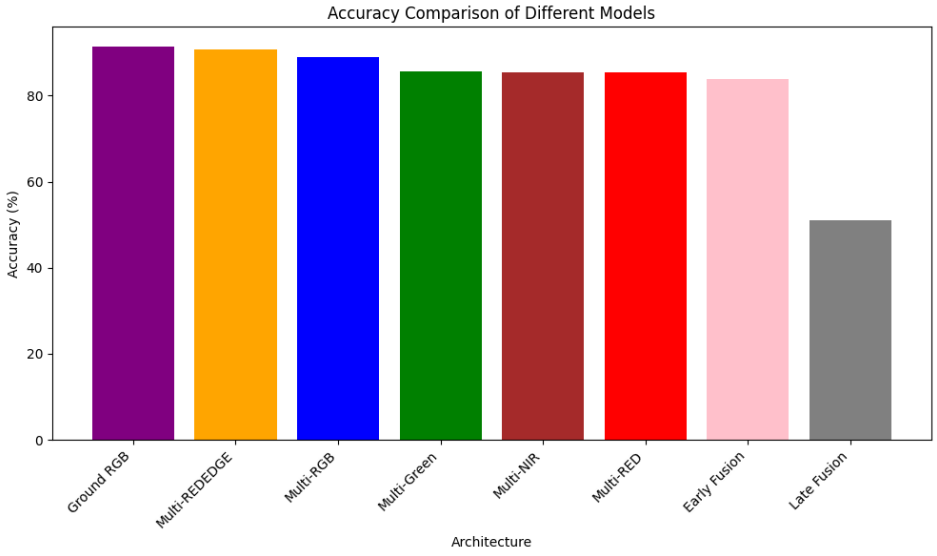
2.2. Early Fusion

Early fusion outperforms Late Fusion. The accuracy, precision, and recall metrics are 83.90% for the test data. However, it appears that the model has overfitted as the metrics are higher, reaching 97%, for the training data. A more detailed analysis of the performance can be seen in the following confusion matrix:



V. Conclusions and Perspectives

The graph above synthesizes the performances of these architectures. It is noticeable that in this scenario, the use of multimodal architectures does not provide significant advantages. Unimodal models outperform multimodal architectures (Early Fusion and Late Fusion) by a large margin.



Clearly, multimodal architectures are much more complex than unimodal architectures. Among all modalities, it is evident that the "Ground RGB" modality outperforms the others. Therefore, our recommendation is to use it as the final model in production for classifying the different health states of trees.

Furthermore, integrating UAV images into this model is no longer necessary. Additionally, since there is only one image for each UAV sub-modality, such as the NDVI sub-modality, it would introduce noise and increase the complexity of the architecture in terms of time and resources required. Moreover, given that there is only one data point per sub-category, it would not have a significant impact on the model and could even decrease its performance. In summary, there is no benefit in using it. Since our goal in this project is to deploy a multimodal architecture, we can use the Early Fusion architecture to containerize it in the final application.

In terms of future perspectives, one approach could be to consider all affected trees, regardless of their stage 2 or 3, as affected (in a single class). This would transform the multi-class classification problem into a binary classification. This approach would not only increase the proportion of diseased trees but also improve the overall model performance.

Another strategy to consider would be to use hybrid techniques and autoencoders to improve performance. This would involve using deep learning for feature extraction and then applying classical machine learning models for classification. This combination of techniques would leverage the advantages of each method, thereby optimizing classification performance.

VI. Flask Application

Currently, we are fully engaged in the development of the Flask application, a crucial component for effectively integrating our final model. We are committed to delivering a high-quality product, ready for use, by the end of the week of March 4th. This development period is essential to allow us to explore and implement all the necessary features for an optimal user experience. Additionally, we are dedicating time to ensuring that our application fully meets the required multimodality criteria by effectively integrating various data sources. We are confident that this approach will enable us to provide a robust and comprehensive solution that meets the needs and expectations involved in the project.

Références

[1] Christos Chaschatzis et al. (2022). Cherry Tree Disease Detection Dataset [Data set]. Zenodo. URL: https://zenodo.org/records/7144071

[2] Idlahcen, F.; Mboukou, P.; Zerouaoui, H. and Idri, A. (2022). Whole-slide Classification of HE-stained Cervix Uteri Tissue using Deep Neural Networks. In Proceedings of the 14th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - Volume 1: KDIR, ISBN 978-989-758-614-9, ISSN 2184-3228, pages 322-329

[3] Idlahcen, Ferdaous & Idri, Ali & Zerouaoui, Hasnae. (2023). Integrating Autoencoder-Based Hybrid Models into Cervical Carcinoma Prediction from Liquid-Based Cytology. 343-350. 10.5220/0012084600003541.

[4] Shih-Cheng Huang et al. Fusion of medical imaging and electronic health records using deep learning: a systematic review and implementation guidelines. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7567861/