Chapter 5

Achievement and Results

After detailing the data acquisition and preparation phase, we dive in details with the modeling approaches that we opted for. Then, we will present the hardware, software and libraries used in this project . Finally, we will explain the obtained results and interpret them after presenting the chosen evaluation metrics.

5.1 Modeling

In spite of the fact that deep learning has picked up centrality all through the last two decades, a growing number of engineers and researchers have built frameworks for many tasks. Fortunately, we have a set of papers proposing different frameworks that allow us to create tools that give better degree of abstraction.

5.1.1 Fusion Color and Infrared Image

In order to have a high resolution image taken under uncontrolled conditions, we selected two prominent approaches from those listed in the state of the art. The first one proposes to merge color and infrared images and implements a convolution neural network as suggested in [Yu Zhanga et al, 2020]. The selected model have three fusion methods; mean, max and sum. The second method proposes a super-resolution GAN architecture suggested in [Ledig et al,2016]. We implemented the two selected neural network architectures using Pytorch as a processing engine.

An infrared image and color image are given as input to the fusion model and the fused image is given as input to the super-resolution model which returns a high resolution image as illustrated in the figure n°5.1. In order to improve the performance of the model, we used well-trained weights and the best used learning rate.

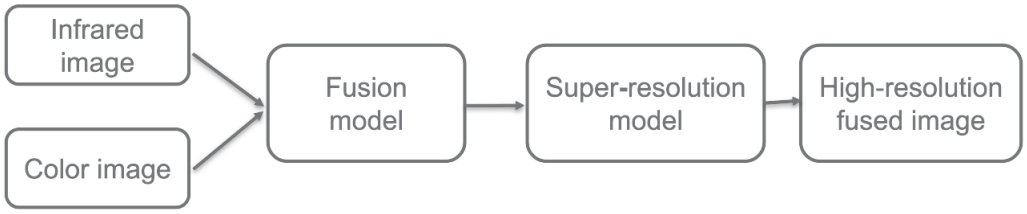

Figure 5.1: The Fusion Workflow

5.1.2 Object Detection and Classification

Regarding the state-of-the art and our need for a robust and quick model, we find out that YOLOv5 is the most suitable model for object detection and classification. Usually, training one model requires 48 hours for 1000 epochs and results are not good as another framework. Compared to this, the PyTorch framework utilizes less time and was a lifesaver in terms of speed and precision detection. We performed all our experiments with the PyTorch framework.

The model takes as input an annotated images in YOLO format for the training, validation and testing phases as shown in the proposed workflow figure n°5.2.

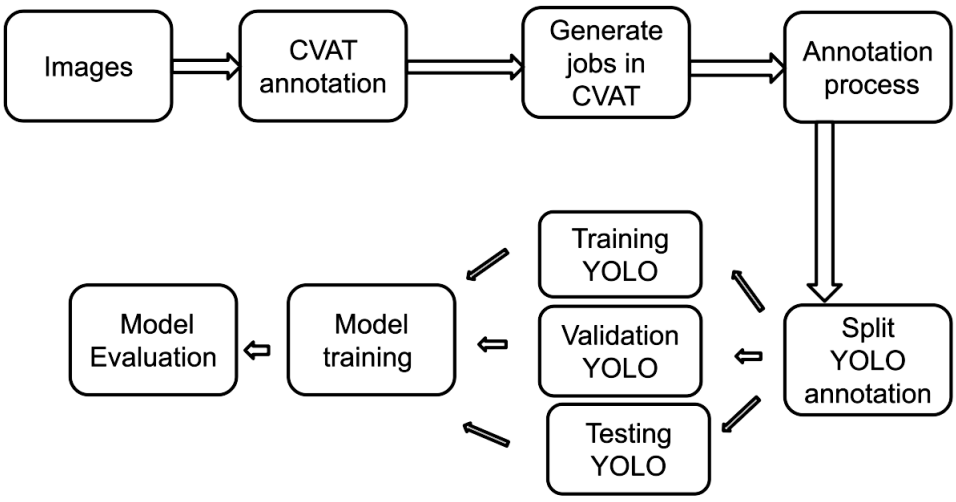

Figure 5.2: Proposed Workflow

5.2 Setup of The Work Environment

In this section, we present the specifications of the machines as well as different software and technologies used in this work.

5.2.1 Hardware Environment

The principal development tool was a personal computer used for several activities,like writing code, debugging, reviewing documents and taking notes having the following characteristics:

- PC1
  - **CPU :** Intel i5-7600 3.5 GHz
  - **RAM :** 8 GB
  - **Hard Drive :** 200 GB
  - **GPU :** Intel HD Graphics 630
  - **Operating system :** Windows 10 Enterprise

  However, a more powerful computer was required for computer-intensive algorithms in order to train and evaluate complex learning model with the following characteristics:
- PC2
  - **CPU :** Intel i7-7820HQ 2.9 GHz
  - **RAM :** 32 GB
  - **Hard Drive :** 500 GB
  - **GPU :** Nvidia Quadro P3000
  - **Operating system :** Windows 10 Pro

5.2.2 Software Environment

- **Python:** is one of the most prevalent programming languages thanks to its extensive set of libraries that implement different solutions and ease development. Therefore, it evolved into the go-to language in Data Science in recent times. We opted for working with this Python in this research project thanks to its several reasons:
  - it is flexible language which is a required quality for the language we have to work with.
  - it has various and regularly evolving libraries.
  - it is an easy readable language having a simplified syntax.
- **Anaconda:** is an open source distribution of python programming language for scientific computing such as machine learning , data science, data processing etc. It has a desktop graphical user interface that enables launching applications.
- **Google Colab:** an online code editor used to write, run and test python code without the necessity for the set up of the environment. The code is executed on distributed cloud machines and relieves development machine's resource consumption.

- **JupyterLab:** is an online user interface for Jupyter projects that build applications in an interactional computing sessions. Thanks to its fast visualization of results, Jupyter notebooks are commonly used in machine learning projects.

5.2.3 Frameworks and Libraries

- **Numpy** is one of the most used Python library that offers implementation for mathematical functions and assistance for wide multi-dimensional arrays and matrices.
- **Scikit learn** is a well-used open source Python library that offers tools for machine learning.
- **Scikit image** is Python library used in image processing and offers implementations for the latest image processing functions.
- **Pandas** is a library that offers data structures and operations for manipulating numeric tables.
- **PyTorch** is the first framework utilized in arising of deep learning models. It has high-level computational and deep neural network functions.It also has adjustable, fast code development and is usually used in the research works.
- **Tensorflow** is mostly implemented in complex Neural Network algorithms. It could be ruuned in different CPUs/GPUs.
- **Keras** offers a high level API for complex Neural Network algorithms which considerably facilitates the code and reduces development time.

5.3 Evaluation Metrics

This section is a brief summary of the notions mandatory to cover the evaluation metrics used in this study. Mean Average Precision (mAP) is the foremost feasible and frequently utilized valuation approach for object detection. This evaluation metric is utilized to measure their performance by the foremost prevalent object detection frameworks. To understand mAP, firstly, it is required to apprehend the sensitivity and specificity of a binary classification test as statistical reviews.

Sensitivity is frequently named to be a true positive rate or recall. It evaluates the percentage of true positive (TP)components detected precisely. Announcement of true positive and false positive (FP) for object detection depends on an indicated IoU threshold for object detection. The IoU level is commonly set at 50%, 75%, or 95%. The TP and FP are hypothesized as below:

- **True Positive (TP):** Proportion of true positives appropriately identified with an IOU $\geq$ an indicated limit. When object detection is carried out, a truly positive item in the frame is correctly acknowledged.

- **False Positive (FP):** Proportion of true negatives with an IOU less than or equal to an indicated limit that were incorrectly classified as positive. This could be noticed as faulty detection.

The true negative proportion is additionally another designation for specificity. The following are the descriptions of "True Negatives (TN) " and " False Negatives (FN) ":

- **True Negative (TN):** Proportion of true negatives that are effectively detected, for instance true negatives for object detection, are states when non-object areas are precisely classed as non-object areas.
- **False Negative (FN):** A state in which a detection algorithm breaks down to acknowledge the object.

Accuracy: The Accuracy metric which permits us to have an overall idea of the efficiency of the model.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

Precision: Precision alludes to a model's aptitude to identify only appropriate objects, in other words the proportion of correct positive forecasts. It is given by

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{Alldetections} \tag{5.2}$$

Recall: The aptitude of a model to find all pertinent examples is known as recall. It signifies the percentage of true positives that are effectively recognized. It is given by

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{Allpositiveinstances} \tag{5.3}$$

F1-Measure: The F-score is a model's accuracy metric that is utilized in binary classification meaning to classify samples into "positive" or "negative". It is a prevalent metric for imbalanced classification. It joins the model's precision and recall and is calculated as follows

$$F_1 = \frac{2 \times Precision + Recall}{Precision + Recall} \tag{5.4}$$

Precision-Recall Curve: It is a prevalent statistic for observing the performance of an object detection network. An excellent object detector of a particular class is one whose accuracy remains high as recall boosts. An optimization model detector can distinguish all significant items (zero false positives means high accuracy) and find all ground truth items (zero false negatives = high recall). The count of targeted objects rises with time when training an object detector. As a consequence, the recurrence of false positives improves, arising from a loss in accuracy. Therefore the precision-recall curve subsequently starts with high precision values and consequently falls, as the object detector aims to retrieve all ground truth instances. **Average Precision (AP):** An Average Precision (AP) is inferred by standardizing the precision-recall curve's zig-zag structure and after that deciding the area under it. An average precision might

be used to match the capability of various detectors. In reality, a graph with a recall proportion between 0 and 1 is presented and the precision proportion is upgraded with the most noteworthy precision for every recall. For instance, in PASCAL Visual Object Classes (VOC) competition , to demonstrate, the AP (average precision) is computed by assembling the precision for a series of eleven precisely divided recall levels, more particularly 0, 0.1, 0.2,..., 1 as shown in the figure n°5.3. It might be expressed mathematically as

$$AP = \frac{1}{11} \sum_{r \in (0.0...1.0)} AP_r \tag{5.5}$$

$$AP = \frac{1}{11} \sum_{r \in (0.0...1.0)} P_{interp(r)} \tag{5.6}$$

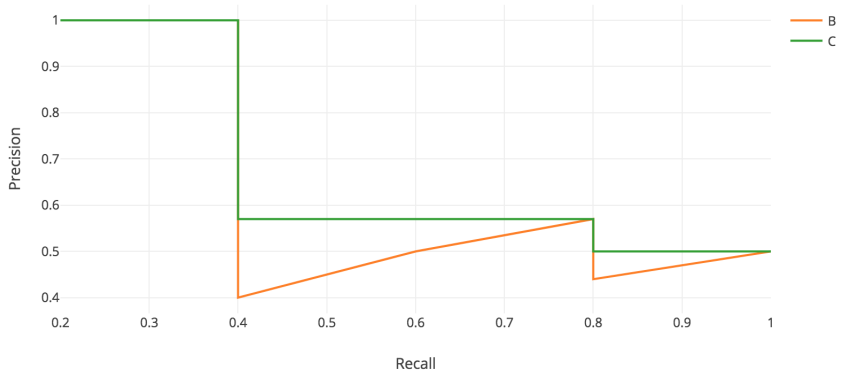

Figure 5.3: The Maximum Precision Score (in green) At Each Recall Rate (0, 0.1,.., 0.9, and 1.0)

Mean Average Precision (mAP): The mean average precision (mAP) is the mean of all classes' AP. Nevertheless, it is commonly designed as AP. mAP@[.5:.95] signifies average mAP over various IoU limits, from 0.5 to 0.95, step 0.05 (0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95). Generally, Mean Average Precision is the most used with imbalanced dataset.

We choose to work with accuracy, recall, precision and F1 as classification metrics and with mean average precision as an object detection metric thanks to its capabilities to assess our model performance and its compatibility with our imbalanced data sets.

5.4 Results and Interpretations

In this section, we will show the outcomes we obtained and we will study them and interpret them.

5.4.1 Datasets Performance Comparison

YOLOv5 required a number of objects more around 350 for the training phase. As we did not get the required number of objects for Iceberg and Romain typologies, we did not applied

YOLOv5 on them. We tested them with the set of typologies together. We tested Batavia, multifeuille, Beurre and chene data sets alone and a data set englobing all typologies.

5.4.1.1 Comparison of The Fused and D2 Datasets Performance

- We tested our YOLOv5 model on our fused data sets with the three fusion method: mean, max and sum and D2 data set, with rotation and flip applied as data augmentation techniques.
- The table 5.1 represents the results of D2 data set Whereas the tables 5.2, 5.3, 5.4 represent respectively the results of the fused data sets with the fusion methods; mean, max and sum.
  - **Object detection:**
    * The object detection performance differs from one typology to an other in all the data sets as shown in tested fused images in the figures n°5.4, 5.5 and 5.6; the confidence scores varies from one typology to an other.

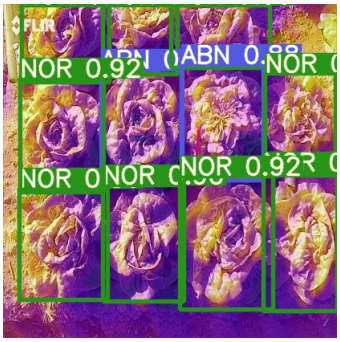

Figure 5.4: Beurre

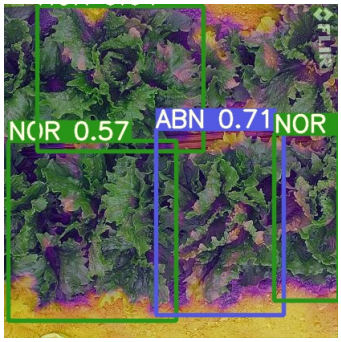

Figure 5.5: Iceberg

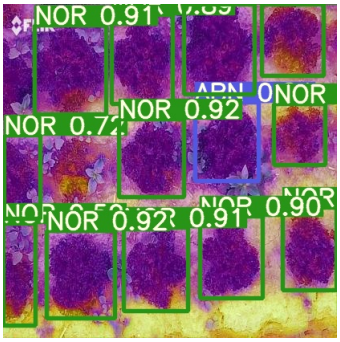

Figure 5.6: Multifeuille

    * The mAP metric also shows that the model performance for object detection differs according to the typology. The best rates corresponds to the beurre typology with the highest value of the mAP equal to 0.6945 using D2 data set with data augmentation techniques, in table n°5.1. The second values goes to Chene typology with a mAP value equal to 0.5937 with D2 data set applied on it data augmentation. The least values goes to Batavia typology with a value of 0.329 with D2 data set without data augmentation as shown in table n°5.1.
    * One of the reasons of the difference of results between typologies is the fact that the beurre and chene typologies are planted spaced however the lettuces of batavia typology are planted too close to each other; there is an overlapping between the lettuces. The shape of the lettuce is also a factor for the object detection; the more the lettuce is rounded with, the better the model detects the object. The tested images with Iceberg have the least outcomes of object detection as the foliage is incredibly interlocking. The figures n°5.7, 5.8, 5.9, 5.10, 5.11

and 5.12 indicate the shape of the different typologies and how close the lettuces are planted.

Figure 5.7: Beurre

Figure 5.8: Chene

Figure 5.9: Batavia

Figure 5.10: Iceberg

Figure 5.11: Romaine

Figure 5.12: Multifeuille

* The results of the four data sets are close and the difference is in a range of [0.01, 0,17]. The D2 data set gave the best results in the majority of the sets except with batavia and multifeuille typologies. For the fused data sets, the mean data set have higher values of detection than the other fused data sets.
* The mAP values are better with data augmentation data sets for all the data sets. The more the model learn to detect one object from different position in the image, the more the model capacity to detect objects increase.
* The D2 data set with data augmentation have the highest value 0.59. The mAP of all data sets of the all typologies have better results than many typologies alone; this is due to the fact that the more we give images and objects to the model, the more the model learn to detect objects.

– **Classification**

* The classification metrics values have shown that the more the model detect objects the better the classification is ; the beurre typology have also the highest values comparing with the other typologies.
* One reasons for this difference is the fact that some typologies have more off-types that others; Romaine typology is characterized as typology by having less number of off-types. Actually, the more the model get trained on abnormal lettuces , the more the model can make a correct classification. Moreover, some typologies have more balanced data set, as a result the model learns better as indicated in the tables in the chapter 4. Besides, some typologies have distinctive

off-types as shown the figure n°5.13 others have less notable off-types as shown in the figure n°5.14.

Figure 5.13: Indistinctive Batavia Off-type

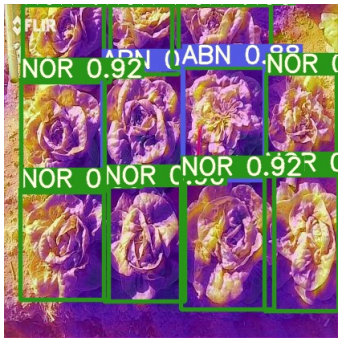

Figure 5.14: Distinctive Beurre Typology

5.4.1.2 Raw RGB Dataset: D1

- Table 5.1 shows the outcomes of the RGB data sets D1; with and without applying data augmentation techniques.
- Due to the fact that this data set includes more images and raw images, the results, in object detection and classification, with all the data sets are better than D1 data set. The mAP acheived 0.7452 in all typologies data set with data augmentation. The F1 reached 0.8705 with the same data sets. The multifeuille data set without applying data augmentation had the lowest results in object detection equal to 0.4556 and in classification 0.6493.

Conclusion

In this chapter, we started by detailing the methods we have selected. Later, we cited the setup of the work environment. Then we presented the evaluation metrics that fit our model and finally we analysed and interpreted the obtained results.

Table 5.1: Classification Results on RGB dataset (D2)

Typology	Data augmentation		Recall	Accuracy	Precision	F_1	mAP
	Without	Flip & Rotation					
Batavia	x		0.618	0.38	0.486	0.544	0.329
Batavia		x	0.5538	0.44	0.6240	0.5868	0.3873
Multifeuille	x		0.720	0.46	0.367	0.486	0.4189
Multifeuille		x	0.672	0.765	0.723	0.697	0.544
Beurre	x		0.8564	0.77	0.8057	0.8328	0.6758
Beurre		x	0.8774	0.83	0.8997	0.8884	0.6945
Chene	x		0.7261	0.555	0.7928	0.759	0.5797
Chene		x	0.7464	0.69	0.8246	0.8126	0.5937
All typologies	x		0.790	0.835	0.801	0.795	0.560
All typologies		x	0.78	0.835	0.81	0.795	0.59

Table 5.2: Classification Results on Fused Dataset with Mean as Fusion Method

Typology	Data augmentation		Recall	Accuracy	Precision	F_1	mAP
	without	Flip & Rotation					
Batavia	x		0.672	0.4	0.5325	0.5942	0.3927
Batavia		x	0.779	0.625	0.553	0.647	0.484
Multifeuille	x		0.753	0.43	0.408	0.529	0.4398
Multifeuille		x	0.6897	0.74	0.7287	0.7087	0.578
Beurre	x		0.7966	0.79	0.8613	0.8277	0.6044
Beurre		x	0.855	0.815	0.872	0.863	0.644
Chene	x		0.6432	0.555	0.6369	0.6400	0.4215
Chene		x	0.7270	0.725	0.6859	0.7059	0.4369
All typologies	x		0.6503	0.665	0.6217	0.6357	0.4431
All typologies		x	0.6704	0.72	0.6681	0.6693	0.4591

Table 5.3: Classification Results on Fused Dataset with Max as Fusion Method

Typology	Data augmentation		Recall	Accuracy	Precision	F_1	mAP
	without	Flip & Rotation					
Batavia	x		0.751	0.385	0.457	0.568	0.382
Batavia		x	0.6060	0.495	0.5935	0.5997	0.3490
Multifeuille	x		0.6726	0.42	0.4022	0.5034	0.4326
Multifeuille		x	0.733	0.562	0.824	0.776	0.5715
Beurre	x		0.810	0.795	0.798	0.804	0.570
Beurre		x	0.9189	0.735	0.7401	0.8199	0.6324
Chene	x		0.860	0.54	0.589	0.589	0.3745
Chene		x	0.782	0.59	0.492	0.604	0.428
All typologies	x		0.649	0.61	0.657	0.653	0.472
All typologies		x	0.710	0.705	0.708	0.709	0.508

Table 5.4: Classification Results on Fused Dataset with Sum as Fusion Method

Typology	Data augmentation		Recall	Accuracy	Precision	F_1	mAP
	without	Flip & Rotation					
Batavia	x		0.6698	0.385	0.3504	0.4694	0.2242
Batavia		x	0.581	0.385	0.442	0.502	0.342
Multifeuille	x		0.82	0.46	0.3660	0.4646	0.380
Multifeuille		x	0.7696	0.405	0.4007	0.5270	0.4144
Beurre	x		0.8419	0.685	0.6398	0.7271	0.5407
Beurre		x	0.6868	0.745	0.8130	0.744	0.5750
Chene	x		0.7025	0.61	0.6614	0.6813	0.4537
Chene		x	0.7582	0.735	0.6785	0.7160	0.4482
All typologies	x		0.6739	0.685	0.6590	0.6664	0.4915
All typologies		x	0.7267	0.7368	0.7864	0.7563	0.5774

Table 5.5: Classification Results on RGB Dataset (D1)

Typology	Data augmentation		Recall	Accuracy	Precision	F_1	mAP
	Flip&Rotation	Flip&Rotation&Saturation					
Batavia	NO	NO	0.7231	0.69	0.8334	0.7743	0.6210
Batavia	YES	NO	0.6965	0.7	0.8119	0.7498	0.6297
Batavia	NO	YES	0.7106	0.685	0.8201	0.7614	0.6321
Multifeu.	NO	NO	0.7164	0.445	0.5937	0.6493	0.4556
Multifeu.	YES	NO	0.7735	0.745	0.7909	0.7821	0.5604
Multifeu.	NO	YES	0.7584	0.71	0.8493	0.8013	0.5896
Beurre	NO	NO	0.8564	0.77	0.8057	0.8328	0.6758
Beurre	YES	NO	0.8774	0.83	0.8997	0.8884	0.6945
Beurre	NO	YES	0.9215	0.88	0.9037	0.9125	0.7482
Chene	NO	NO	0.7261	0.555	0.7928	0.7590	0.5797
Chene	YES	NO	0.7464	0.69	0.8246	0.8126	0.5937
Chene	NO	YES	0.7604	0.86	0.9067	0.8271	0.6264
All	NO	NO	0.8231	0.8	0.8462	0.8345	0.7324
All	YES	NO	0.7986	0.785	0.8645	0.8302	0.7452
All	NO	YES	0.8551	0.86	0.8864	0.8705	0.7695