Course 71254 - Introduction to Image Processing and Analysis

Comparison Between Traditional vs. Deep Learning Image Processing For Damaged Leaf Area

Submitted by:

Yonatan Schwartz

Nethanel Fishman

<u>Table of content</u>

1. Introduction

Image processing and analysis are essential research and technological fields, crucial for various applications, including precision agriculture, object detection in images, and environmental monitoring. These technologies enable identifying visual patterns and quantifying and analyzing changes at physical or biological levels. Traditional image processing methods are based on direct mathematical manipulations, such as filters, masks, edge detection, color thresholds, and histograms. This method has proven reliable and easy to implement and does not require special devices or processing power. However, these methods can be limited in handling high-complexity images, such as analyzing overlapping regions, significant noise, distinguishing objects, and dealing with uneven or suboptimal image quality [1][2]. With the emergence of deep learning and conventional neural networks (CNN), image processing has significantly improved. CNN, such as YOLO (You Only Look Once), provides accurate and fast tool for object detection and segmentation of relevant image regions. YOLO focuses on real-time object detection while balancing speed and performance. It is particularly suited for agricultural tasks, such as detecting leaf defects, due to its ability to analyze images of varying resolutions under diverse conditions accurately and quickly [3]. This project compares traditional image processing methods with deep-learning-based image processing, focusing on analyzing damaged areas in leaves. Binary masks are used to segment and quantify the damaged regions. The primary objective is to identify the most accurate and suitable method for practical applications, considering time and resource constraints [4]. Additionally, the project examines the impact of outliers, such as images with damage ratios exceeding 200%, and evaluates their contribution to overall assessment quality. This integrated understanding can pave the way for better solutions in agro-technological systems.

2. Database

The project's database was downloaded from the Roboflow website. It contains 1,260 images (640 X 640) divided as follows:

- 1,008 images for the training set.
- 126 images for the validation set.
- 126 images for the test set.

It is important to note that the dataset includes images that underwent instance segmentation, which significantly simplified the calculation of the pixel count for the damaged area, enabling effective comparison.

The dataset format was downloaded specifically for training a YOLO model, particularly the YOLOv11n-seg model (fine-tuning). The images in the dataset are in JPG format, which helps maintain the data while ensuring relatively small file sizes.
The dataset comprises three detection and segmentation categories: Rust, Phoma, and Miner. The first two categories, are fungal diseases, whereas Miner refers to leaf damage caused by harmful insects burrowing into the mesophyll tissue of the leaf. These issues are very common in crops, and each has a distinct visual pattern that makes it identifiable. We chose this dataset because it contains enough images to train a model effectively, includes broad and widely recognized detection categories, and the food security and agricultural stability tasks can get easier due to "smart" monitoring of these diseases [2].

3. Results

The results of the project were chosen to be presented in several ways.

First, we will showcase the training results of the model.
Next, we will present the model's prediction results on the validation set, establishing the ground truth for our project.
Following that, we will display the results of the comparison between traditional image processing and deep learning-based image processing, considering deep learning processing as the ground truth. Specifically, we will highlight the percentage of the damaged area calculated using traditional image processing compared to deep learning-based processing.
Finally, we will present a similar comparison in which we excluded images with damaged areas exceeding 200% relative to traditional image processing. These excluded images were poorly captured and considered anomalies. This comparison aims to emphasize the impact of image capture quality on the choice of image processing method.

In the code notebook, we recently conducted a similar comparison where the deviation range was set between 90% and 110%, meaning a 10% deviation, which allows for greater accuracy. In the code notebook, a 3rd comparison version is available, where the lower and upper thresholds can be entered interactively.

An important emphasis to note is that a high prediction accuracy was achieved after training the model (over 90%). Therefore, the comparison between the traditional method and deep learning is made directly against the model and not against the human-performed segmentation of the dataset.

3.1. Training results

3.1.1. Normalized confusion matrix

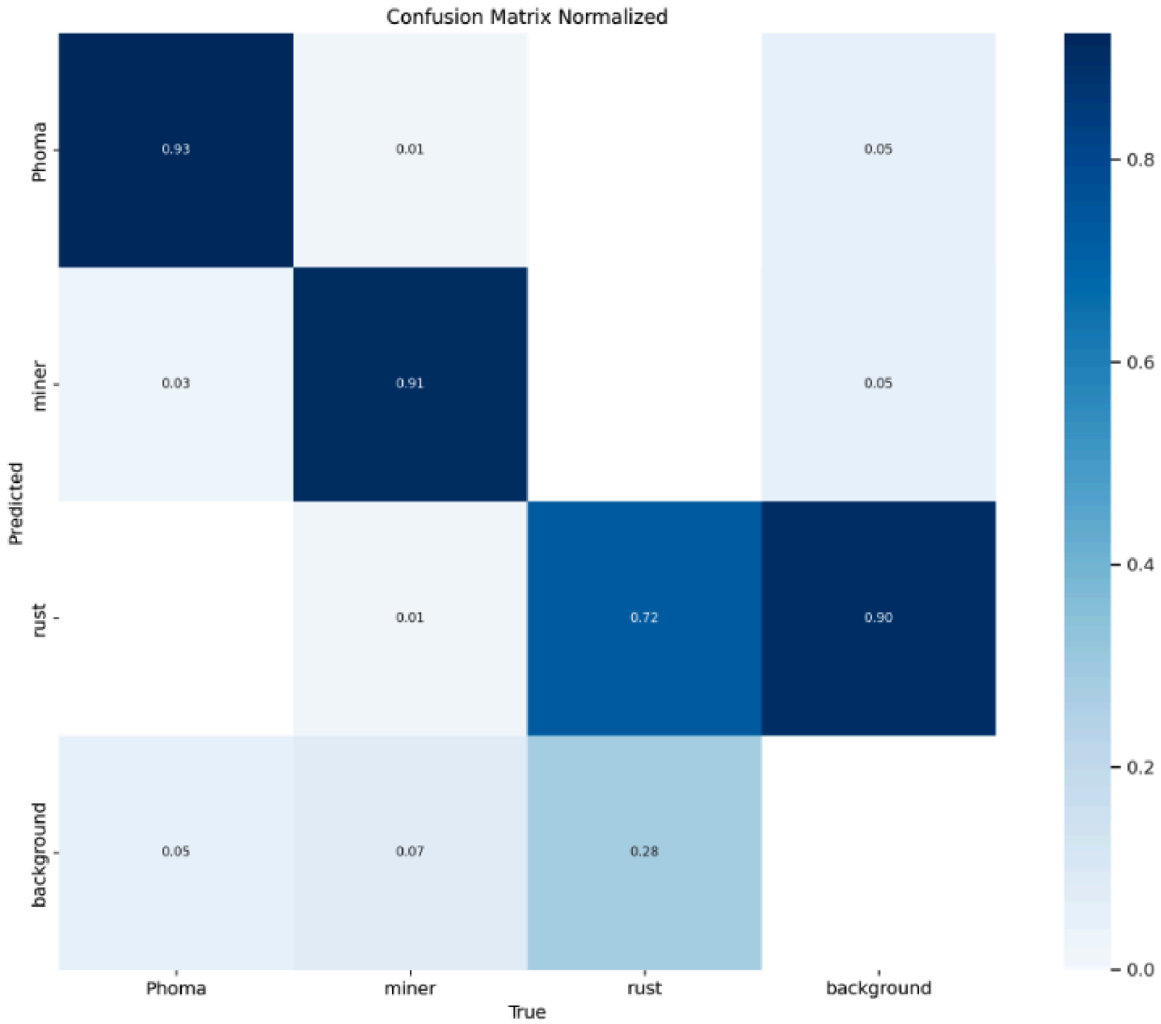

Fig 1 - Normalized confusion matrix

The matrix of the model provides a clear representation of its classification performance for four categories: rust, miner, phoma, and background.
The matrix rows represent the model's predictions, while the columns represent the true labels.

The results show that phoma and miner are well-classified with high accuracy and a low error rate. This suggests that the visual characteristics of these classes are well-defined, and the model has learned to identify them effectively. However, the **rust** class, with a lower accuracy rate, is more challenging for the model. This indicates that additional fine-tuning or better representation of this class in the dataset might be required to improve its performance.

3.1.2 The training results graphs

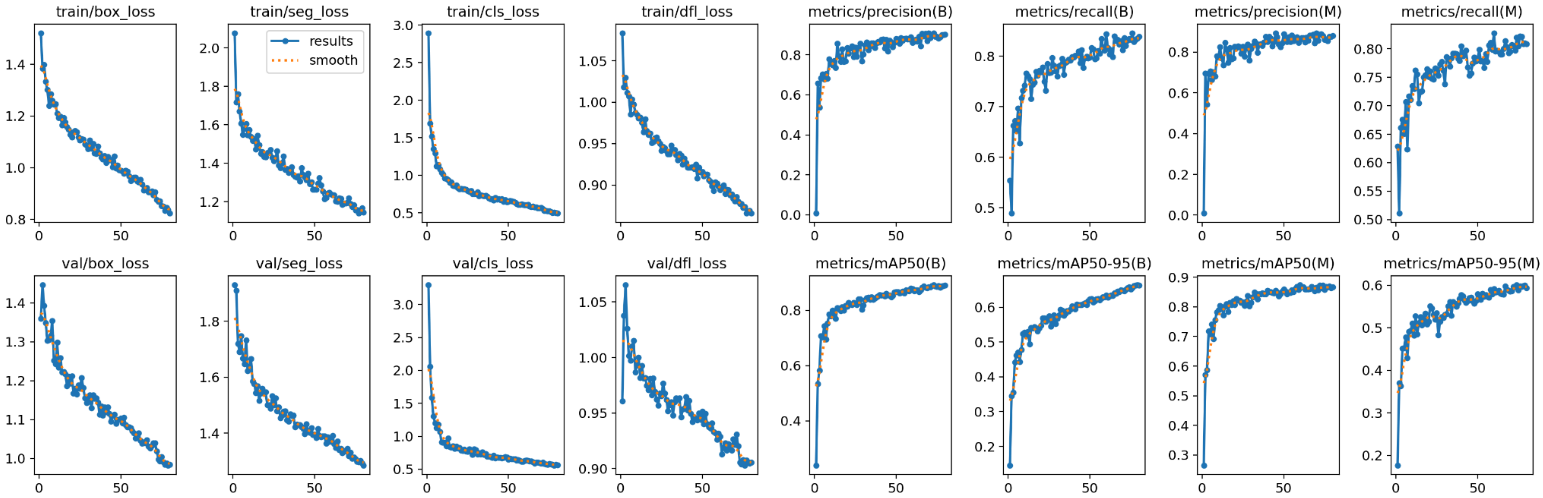

Fig 2- Training graphs results

Overall, the consistent decrease in losses and the increase in performance metrics suggest an effective learning process where the model improves steadily. The similarity between training and validation trends indicates that the model does not suffer from overfitting and performs well on unseen data. The high values of recall, precision, and mAP (mean Average Precision) confirm the model's ability to perform accurate segmentation and classification. This reflects the model's success in learning the data patterns and achieving robust generalization.

3.2. Model prediction results on val set

Table 1- Performance Metrics for Object Detection and Instance Segmentation by Class

Class	Images	Intances	BOX P	R	mAP50	mAP50-95	Mask P	R	mAP50	mAP50-95
All	126	739	0.88	0.839	0.891	0.665	0.854	0.807	0.863	0.595
Phoma	46	107	0.938	0.916	0.976	0.84	0.951	0.925	0.98	0.791
Miner	50	136	0.946	0.926	0.956	0.725	0.917	0.89	0.93	0.669
Rust	37	496	0.756	0.673	0.74	0.431	0.693	0.605	0.68	0.326

The table presents the model's performance results on the test set, the three main classes, Phoma, Miner, and Rust, and an overall evaluation across all classes (All). The model demonstrates particularly high precision for the Phoma and Miner classes, with precision values of 0.938 and 0.946, respectively, indicating that most detections are accurate. Recall values are also high for these classes, with 0.916 for Phoma and 0.926 for Miner, highlighting the model's ability to detect most of the actual instances. The mAP@50 metric confirms the model's high detection accuracy, with 0.98 for Phoma and 0.93 for Miner. However, for the Rust class, performance is notably lower. The precision value stands at 0.756, indicating a higher percentage of false positives, and the recall value of 0.673 suggests that the model detects only a portion of the actual instances. Additionally, Rust's mAP@50 and mAP@50:95 values are significantly lower (0.68 and 0.326, respectively), underscoring the difficulty in detecting this class. Overall, the model achieves solid results, with an overall precision of 0.88, recall of 0.839, and mAP@50 of 0.891, though improving detection for Rust is necessary to ensure consistent performance across all classes.

3.3. Comparison between traditional and deep learning image processing

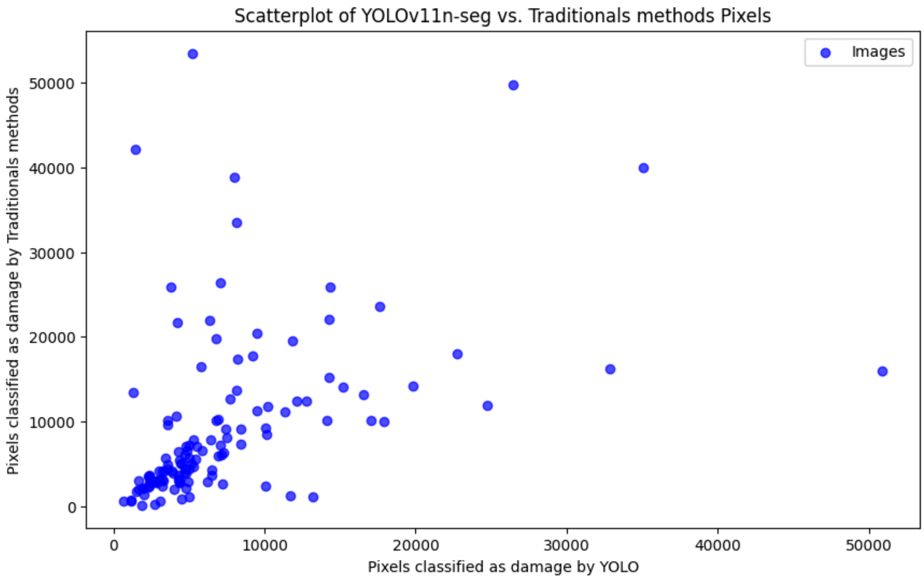

Fig 3- Comparison between YOLOv11n-seg vs. Traditional methods

The scatterplot compares the number of pixels classified as damaged by YOLOv11n-seg and traditional methods. The data points show a wide distribution, with most images concentrated at lower pixel counts, while a few outliers display significantly higher values in both methods.

3.4 Comparison with thresholds

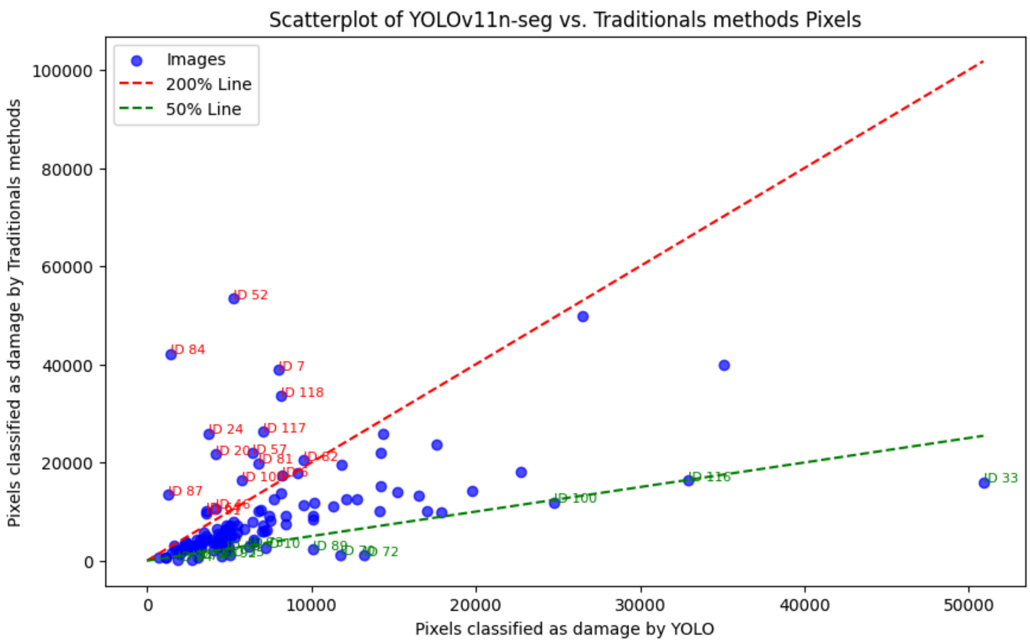

Fig 4– Comparison between YOLOv11n-seg vs. Traditional methods with thresholds

4. Discussion

The model's performance provides valuable insights into damage detection and agricultural segmentation capabilities. Analysis of the results highlights the model's impressive accuracy in identifying the phoma and miner classes, demonstrated by high Precision and Recall scores. These results strongly represent these classes' visual features in the dataset, enabling consistent recognition by the model. In contrast, the rust class poses significant challenges, with noticeably lower performance in Precision, Recall, and mAP metrics. These gaps emphasize the need to expand and diversify the dataset for the rust class or to fine-tune the model better to capture its unique features. The outcomes of the training and validation processes indicate a consistent reduction in losses and improvement in performance metrics, such as accuracy and recall, which suggest effective learning without signs of overfitting. Notably, the high mAP values for phoma (0.98) and miner (0.93) confirm the model's precise segmentation and classification capability. However, the low performance for rust (mAP: 0.68, mAP@50:95: 0.326) highlights the need for further refinements to achieve balanced performance across all categories. The comparison between the traditional method and deep learning model underscores the latter's significant advantages. While the traditional method tends to overestimate damaged areas, with an average ratio of 163.76%, the deep learning model demonstrates greater resilience under varying conditions, such as poor image quality or shadowed regions. Applying a 10% deviation threshold, as commonly accepted in studies, reveals that the traditional method falls within this range for only 24 out of 125 images. In contrast, the deep learning model exhibits far greater consistency. Nonetheless, the traditional method offers practical benefits: it is faster, more straightforward to implement, and accessible to users without advanced technical expertise. It does not require high computational power, pre-trained models, or an annotated dataset, relying instead on basic color threshold adjustments. This makes it a suitable immediate solution when high precision is not essential. However, its limitations—such as overestimating damaged areas and sensitivity to varying imaging conditions—restrict its utility to specific applications. A thoughtful integration of both methods could leverage the speed and simplicity of the traditional approach alongside the precision and robustness of deep learning.

5. Conclusion

The findings of this project emphasize the significant potential of deep learning-based models for accurately detecting and segmenting agricultural damage. The model demonstrated exceptional performance in identifying well-represented classes, such as Phoma and Miner, with high precision, recall, and mAP values. However, the Rust class posed a challenge, underscoring the importance of expanding and diversifying the dataset and improving the model to address this class's unique characteristics.

While the traditional method offers advantages such as speed, simplicity, and accessibility, its tendency to overestimate damaged areas and its limitations under variable imaging conditions make it less reliable for in-depth analyses. In contrast, the deep learning-based model demonstrated high robustness to varying conditions and proved a more accurate and reliable segmentation tool.

From this, it can be concluded that the quality of the images used in both methods is critical—for refining deep learning models improving their performance, and achieving better sensitivity and accuracy in traditional image processing methods.

The project highlights several factors influencing the choice of image processing and analysis methods. Generally, if consistent, high-quality imaging of leaves can be achieved, combined with basic programming knowledge and awareness of potential inaccuracies in results, traditional methods may suffice. Conversely, if precision is critical, there is expertise in model training, and systematic imaging is not required, the deep learning approach is recommended.

Finally, we propose future research directions that incorporate damage causation inputs and track damage progression over time. Such data could be a high-quality database linking damage development, severity, and characteristics to their cause. This connection could provide valuable agronomic insights for developing DSS models to manage pest and disease populations.

6. Bibliography

1. Pride, L., Vallad, G., & Agehara, S. (2020). How to Measure Leaf Disease Damage Using Image Analysis in ImageJ. University of Florida, IFAS Extension.

2. Vanlalhruaizela, B., Lalremsangi, L., Lalrinnungi, L., Pragnya, N., & Vinutha, M. S. (2024). Plant Disease Detection. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, 12(5), 4925-4935. https://doi.org/10.22214/ijraset.2024.62750

3. Pacal, I., Kunduracioglu, I., Alma, M. H., Deveci, M., Kadry, S., Nedoma, J., Slany, V., & Martinek, R. (2024). A Systematic Review of Deep Learning Techniques for Plant Diseases. *Artificial Intelligence Review*, 57, 304-343. https://doi.org/10.1007/s10462-024-10944-7

4. Mitra, A., Mohanty, S. P., & Kougianos, E. (2022). aGROdet: A Novel Framework for Plant Disease Detection and Leaf Damage Estimation. In *Internet of Things: IoT through a Multi-disciplinary Perspective* (pp. 3–22). Springer. https://doi.org/10.1007/978-3-031-18872-5_1

7. Links

Code link

Github link