Measuring YOLOv8 Performance on Diverse Object Detection Datasets

Petec Răzvan-Gabriel

Advanced Methods in Data Analysis – Research Report 1
Specialization: Applied Computation Intelligence
Group: 246/2

razvan.petec@stud.ubbcluj.ro

ABSTRACT

Object detection is an important task in computer vision, used in areas like autonomous vehicles, robotics, and surveillance. YOLOv8, one of the latest versions of the YOLO series, represents one of the fastest, accurate and efficient object detection model. This study tests YOLOv8 on three datasets: SkyFusion, which focuses on tiny aerial objects; Traffic Signs, for detecting road signs; and Multilabel Fruits, which deals with agricultural objects. The results show that YOLOv8 performs well with well-distributed and distinct objects, such as in the Traffic Signs dataset, achieving high precision and recall. However, its performance decreases with small objects and imbalanced datasets, as seen in SkyFusion and Multilabel Fruits. The study highlights YOLOv8's strengths in handling average-sized objects and balanced datasets but points out challenges in generalizing to more diverse scenarios. This research provides insights into YOLOv8's capabilities and areas for improvement in real-time object detection.

1. INTRODUCTION

Object detection is a fundamental task in computer vision that involves identifying and localizing multiple objects within an image or video frame, typically by drawing bounding boxes around the objects and classifying them into predefined categories [1].

The YOLO (You Only Look Once) series of algorithms were firstly introduced by Redmon J. [2] and they gained significant attention due to their ability to perform real-time object detection with high accuracy. This series changed object detection by treating it as a single task. It processes the entire image in one step using a CNN to predict bounding boxes and probabilities. YOLOv8 [3] builds on earlier versions, adding new features and improvements that make it more accurate, efficient, and practical for detecting objects in real time.

To prove the capabilities of this state-of-the-art model, this study will test the model across three distinct datasets, each representing unique real-world object detection challenges:

- **SkyFusion: Aerial Object Detection** [4], dataset designed for detecting small objects in satellite images. This dataset addresses the specific challenges associated with tiny object detection; a task that has been underrepresented in existing datasets.
- **Traffic Signs Detection** [5], dataset focused on identifying and classifying a variety of traffic signs, including traffic lights (Green and Red), speed limit signs and Stop signs. This dataset is crucial for computer vision applications in autonomous driving and traffic monitoring,

providing a diverse range of real-world road scenarios that require accurate detection of different types of traffic signs.

- **Multilabel Fruits Detection** [6], dataset containing images of six types of fruit: Apple, Grapes, Pineapple, Orange, Banana, and Watermelon. This dataset is designed for object detection tasks in agricultural or retail environments, where distinguishing between different types of fruit is essential.

To provide a comprehensive analysis, the report begins with a review of object detection methods, highlighting the evolution of YOLO algorithms and YOLOv8's innovations in speed, accuracy, and loss functions. It then describes the methodology, including datasets (SkyFusion, Traffic Signs, and Multilabel Fruits), training setup, and evaluation metrics such as mAP and classification curves. Then the results of YOLOv8's performance are analysed, addressing challenges like small objects and class imbalance in **SkyFusion**, strong detection in **Traffic Signs**, and misclassifications in the imbalanced **Multilabel Fruits** dataset. The study concludes by summarizing findings and proposing certain improvements.

2. LITERATURE REVIEW

2.1. Object Detection: An Overview of Algorithms

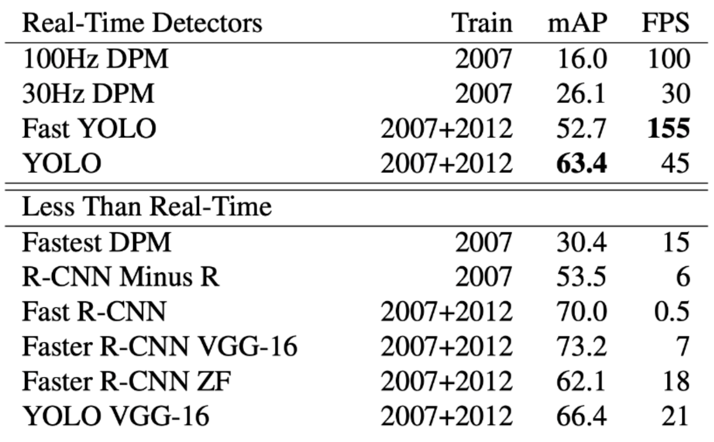
Object detection is a computer vision task that involves identifying and localizing multiple objects within an image or video frame. Formally, given an input image I of dimensions $H \times W \times C$, where H and W represent the height and width of the image and C denotes the number of channels, the goal is to predict a set of N objects $\{O_1, O_2, ..., O_N\}$. Each object O is described by a bounding box $B_i = (x, y, w, h)$, which defines its position and dimensions in the image, and a class label C_i from a predefined set of categories. Additionally, a confidence score S_i is often associated with each detection, indicating the likelihood that O_i corresponds to the predicted class. The model output typically consists of these predictions as $\{(B_i, C_i, S_i)\}_{i=1...N}$, with the aim of maximizing accuracy in object localization and classification while minimizing false positives and false negatives.

Object detection has been a central focus in the field of computer vision, progressing from early handcrafted approaches to advanced deep learning-based techniques. These methods struggled with scalability and accuracy in diverse scenes [7].

The introduction of deep learning transformed object detection by leveraging convolutional neural networks (CNNs) for feature extraction and classification. Two main paradigms emerged: two-stage and one-stage detectors. Two-stage detectors, such as Faster R-CNN, operate by first generating region proposals and then classifying them. Although these models achieve high precision, their sequential processing of proposals results in significant computational overhead, making them impractical for real-time applications [7].

Conversely, one-stage detectors revolutionized the field by integrating object localization and classification into a single step. YOLO (You Only Look Once), introduced by Redmon [2], treated object detection as a regression problem, predicting bounding boxes and class probabilities directly from an image. This unified approach dramatically increased detection speed while maintaining competitive accuracy, making YOLO particularly suitable for applications requiring real-time performance [1].

A comparative analysis of these models reveals the trade-offs between accuracy and efficiency. Two-stage models, exemplified by Faster R-CNN, typically achieve higher accuracy but require significantly more computational resources. In contrast, YOLO-based models prioritize speed and simplicity, achieving real-time performance with slightly lower accuracy. For example, Redmon [2] demonstrated that YOLOv1 could process 45 frames per second with competitive accuracy, a significant improvement over two-stage detectors that struggle with real-time constraints as illustrated in Figure 1.



Real-Time Detectors	Train	mAP	FPS
100Hz DPM	2007	16.0	100
30Hz DPM	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM	2007	30.4	15
R-CNN Minus R	2007	53.5	6
Fast R-CNN	2007+2012	70.0	0.5
Faster R-CNN VGG-16	2007+2012	73.2	7
Faster R-CNN ZF	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Figure 1. *Real-Time Systems on PASCAL VOC 2007. Comparing the performance and speed of fast detectors. [2]*

2.2. *The YOLO Model*

2.2.1. Evolution and Advances

The YOLO series, firstly introduced by Redmon et al. [2], has profoundly influenced object detection by making it a single regression task. This approach processes the entire image in one pass, predicting bounding boxes and class probabilities simultaneously, thereby achieving unprecedented speed and simplicity in detection tasks. Over successive generations, YOLO has evolved significantly (Figure 2), incorporating advanced techniques to improve both accuracy and efficiency.

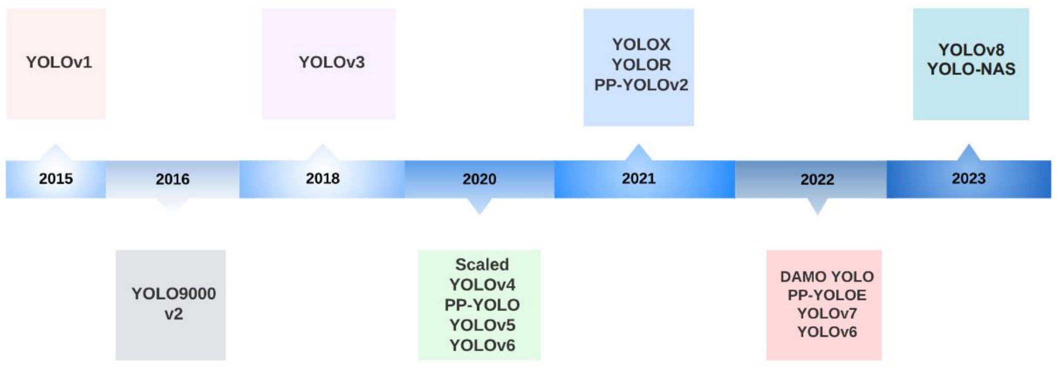


Figure 2. *A timeline of YOLO versions. [1]*

All the YOLO models share the same underlying conceptual pipeline (also displayed in Figure 3), which can be broke down in the following steps:

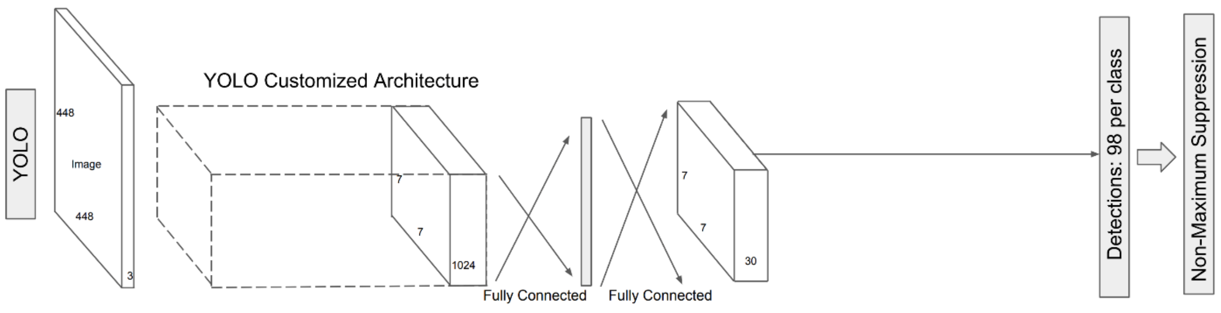


Figure 3. *Conceptual pipeline of the YOLO architecture. [8]*

1. The input image is first processed by a convolutional neural network (CNN), which extracts relevant features from the image.
2. These extracted features are then passed through a series of fully connected layers, which predict the bounding box coordinates and class probabilities. The bounding box predictions consist of four values, each normalized to the range [0,1]: the x and y coordinates of the bounding box center and the lengths of the bounding box along the x-axis and y-axis, normalized by the image width and height, respectively. The class probabilities include k values, where the i-th value indicates the likelihood of the bounding box containing an object of class i.
3. A post-processing step called non-maximum suppression (NMS) is applied to filter out overlapping bounding boxes. This step retains the bounding box with the highest confidence score for each object, as illustrated in Figure 4.
4. The result consists of a set of bounding boxes and their corresponding class labels, representing the detected objects in the image.

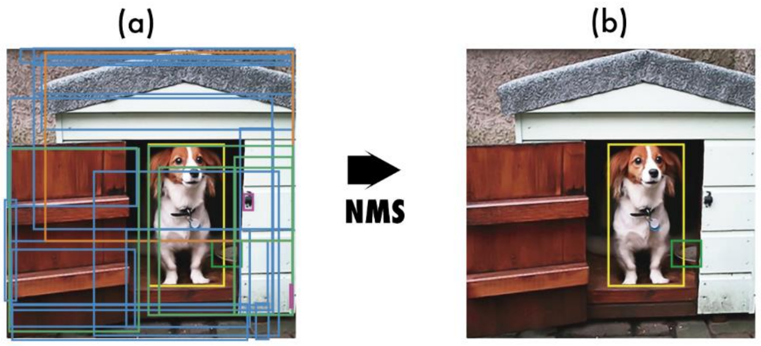


Figure 4. *Non-maximum suppression (NMS) [1]*

2.2.2. *Learning Task*

The loss functions used by YOLO are the localization loss, the classification loss, and confidence estimation loss. These loss functions are designed to ensure better alignment between predicted and ground truth bounding boxes, while also minimizing errors in object classification and detection confidence. The formulas for each loss function can be seen in Figure 5.

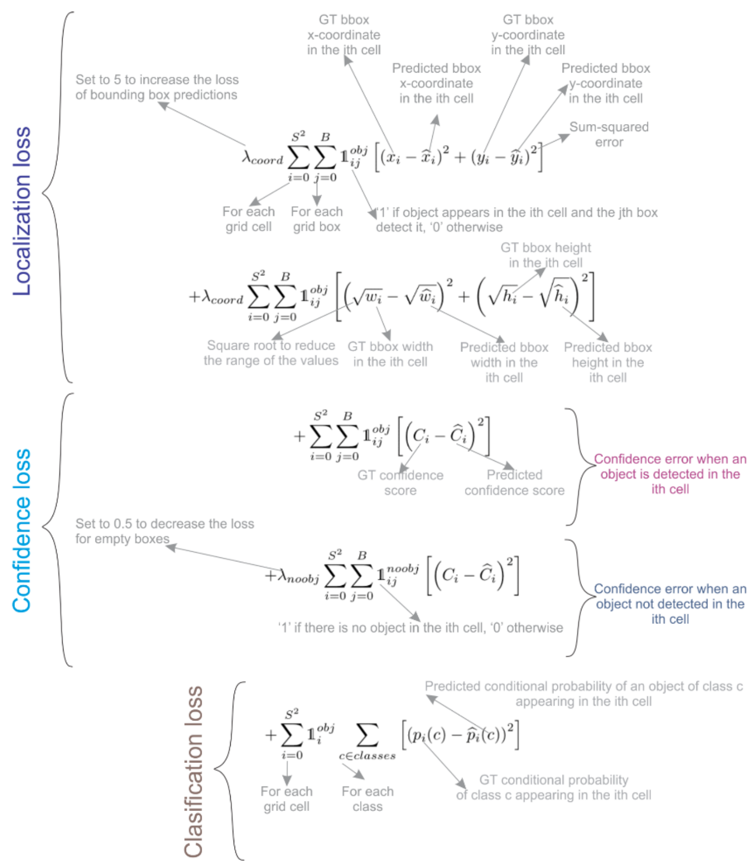

Localization loss
Confidence loss
Clasification loss
GT bbox x-coordinate in the ith cell
GT bbox y-coordinate in the jth cell
Predicted bbox x-coordinate in the ith cell
Predicted bbox y-coordinate in the ith cell
Set to 5 to increase the loss of bounding box predictions
Sum-squared error
$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$
For each grid cell
For each grid box
'1' if object appears in the ith cell and the jth box detect it, '0' otherwise
GT bbox height in the jth cell
$+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$
Square root to reduce the range of the values
GT bbox width in the ith cell
Predicted bbox width in the ith cell
Predicted bbox height in the ith cell
$+\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[\left(C_i - \hat{C}_i \right)^2 \right]$
GT confidence score
Predicted confidence score
Confidence error when an object is detected in the ith cell
Set to 0.5 to decrease the loss for empty boxes
$+\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} \left[\left(C_i - \hat{C}_i \right)^2 \right]$
'1' if there is no object in the ith cell, '0' otherwise
Confidence error when an object not detected in the ith cell
Predicted conditional probability of an object of class c appearing in the ith cell
$+\sum_{i=0}^{S^2} \mathbb{1}_{i}^{obj} \sum_{c \in classes} \left[(p_i(c) - \hat{p}_i(c))^2 \right]$
For each grid cell
For each class
GT conditional probability of class c appearing in the ith cell

Figure 5. YOLO cost functions, which include localization loss for bounding box coordinates, confidence loss for object presence or absence, and classification loss for category prediction accuracy [1].

- **Localization Loss** (noted as L_{box}). This loss calculates the error in predicting the position (x, y) and dimensions (width *w* and height *h*) of the bounding boxes. This loss is only applied to grid cells and bounding boxes that contain objects, as indicated by the 1_{ij}^{obj} term. This ensures that the model is penalized only when it misplaces or incorrectly sizes the bounding boxes for objects. To stabilize the optimization for width and height, the square root of their predicted and ground truth values is used, reducing the magnitude of the loss for larger objects and avoiding bias toward smaller boxes.
- **Confidence Loss** (or **Distributional Focal Loss**, noted as L_{dfl}). This loss evaluates the model's ability to predict whether or not a bounding box contains an object. It has two parts:
    - **Positive Confidence Loss:** This applies when an object is detected in a box (1_{ij}^{obj}), penalizing the difference between the predicted and true confidence scores. This ensures the model becomes confident when it detects actual objects.
    - **Negative Confidence Loss:** This applies when no object is present (1_{ij}^{obj}). Since there are usually many empty grid cells, this loss is scaled down by the factor λ_{noobj} to prevent the model from disproportionately focusing on these empty areas. This adjustment ensures balanced learning across present and absent object cases.
- **Classification Loss** (noted as L_{cls}). This loss calculates the error in predicting the correct category for an object within a grid cell. It evaluates the difference between the predicted and true class probabilities for the detected object, but only if an object is present in the grid cell (1_{ij}^{obj}). This component ensures that the model accurately identifies the object type while focusing only on cells where objects exist.

3. METHODOLOGY

3.1. Datasets

3.1.1. SkyFusion: Aerial Object Detection Dataset [4]

This dataset is specifically designed for training and evaluating models on the task of detecting small objects in various contexts. It includes high-resolution images annotated for small and often hard-to-detect objects, which make it suitable for scenarios like aerial surveillance, industrial inspections, or specialized applications requiring detailed feature recognition. Some samples of this dataset can be seen in Figure 6.

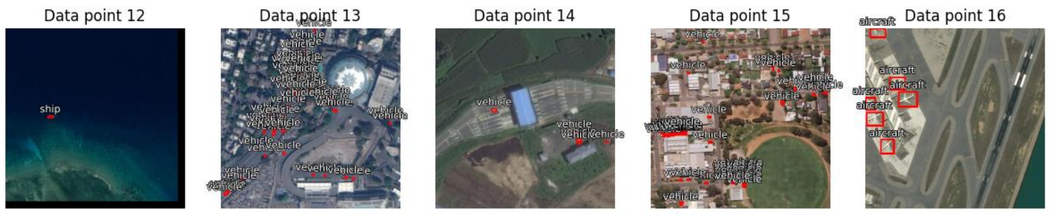


Figure 6. *Samples of the SkyFusion dataset [4]. Sample 12 presents a ship, samples 13-15 presents some vehicles, and samples 16 presents some aircrafts.*

The dataset contains 2094 training samples and 450 validation samples with the resolution 640x640x3 and has 3 classes: *aircraft*, *ship* and *vehicle*. By analysing the bounding boxes details visualizations in Figure 7, we observe that the dataset contains a significant class imbalance, with the "vehicle" class dominating and fewer instances of "aircraft" and "ship," potentially biasing detection models toward better performance on the dominant class. Most objects share similar sizes and aspect ratios, as indicated by the dense overlap of bounding boxes. The heatmap shows an even spatial distribution of annotations, avoiding positional bias and aiding model generalization. However, the scatter plot reveals that most objects are very small, emphasizing the dataset's focus on tiny object detection and the need for precise localization methods to handle this challenge effectively.

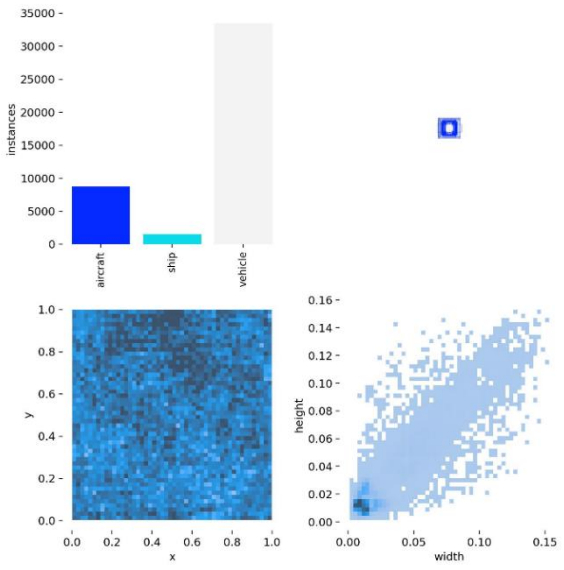


Figure 7. *SkyFusion [4] bounding boxes details visualization. Distribution bar graph (top-left) overlapped bounding boxes (top-right), positions heatmap (bottom-left), lengths scatter (bottom-right).*

3.1.2. Traffic Signs Detection Dataset for Self-Driving Cars [5]

This dataset is focused on identifying and classifying traffic signs, making it valuable for traffic analysis and autonomous driving applications. It includes images of commonly encountered signs such as traffic lights (e.g., red and green signals), speed limit signs (ranging from 10 km/h to 120 km/h), and Stop signs, which collectively represent a wide spectrum of real-world road scenarios. These characteristics make this dataset useful for training object detection models for detecting and recognizing traffic signs in various conditions. Some samples of this dataset can be viewed in Figure 8.

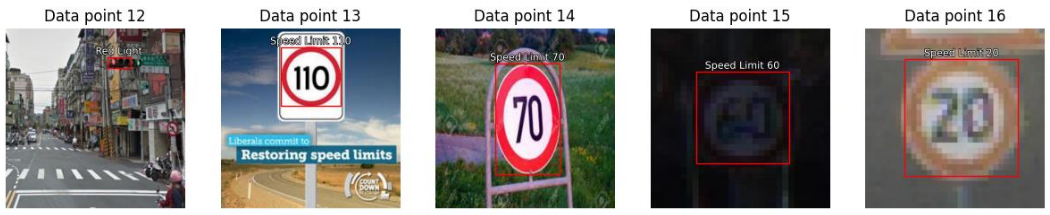


Figure 8. *Samples of the Traffic Signs Detection dataset [5]. Sample 12 presents a red-light semaphore, while samples 13-16 presents some speed limits signs with the limits 110, 70, 60, respectively 20.*

The dataset contains 3530 training samples and 801 validation samples with the resolution 416x416x3 and has 15 classes: *red semaphore light*, *green semaphore light*, *speed limit 10-110* and *stop sign*. By analysing the bounding box details visualized in Figure 9, we can see that the dataset shows significant class imbalance, with "Green Light" dominating and underrepresented classes like "Speed Limit 10" potentially biasing detection models. Most traffic signs share similar sizes and consistent aspect ratios, as revealed by the overlapping bounding boxes and scatter plot. The heatmap indicates a central bias in object annotations, with a higher density near the image center. The small object sizes emphasize the need for high precision and fine-grained feature extraction to effectively detect traffic signs.

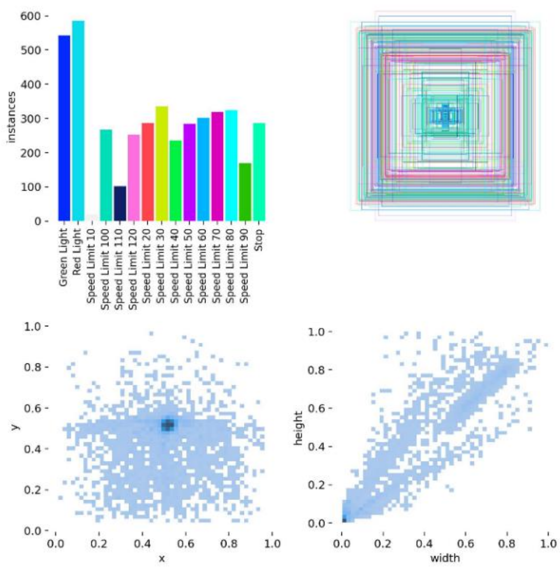


Figure 9. *Traffic Signs Detection [5] bounding boxes details visualization. Distribution bar graph (top-left) overlapped bounding boxes (top-right), positions heatmap (bottom-left), lengths scatter (bottom-right).*

3.1.3. Multilabel Fruits Detection Dataset [6]

The dataset's diverse fruit classes cover a range of real-world scenarios with varying shapes, sizes, and appearances. This diversity allows for robust model training, enabling effective detection under different conditions, such as lighting or occlusions. By focusing exclusively on fruit types, it provides a targeted dataset for applications in agriculture, food supply chain monitoring, or inventory management. Some samples can be seen in Figure 10.

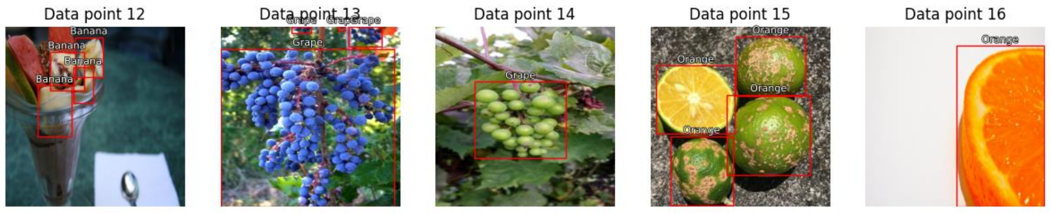


Figure 10. *Samples of the Multilabel Fruits Detection dataset [6]. Sample 12 presents a photo containing pieces of banana, samples 13 and 14 presents photos with grapes and samples 15-16 presents photos with oranges.*

The dataset contains 7108 training samples and 914 validation samples with the resolution 640x640x3 and has 6 classes: *apple*, *banana*, *grape*, *orange*, *pineapple*, and *watermelon*. By analysing the bounding box details visualizations in Figure 11, we can say that the dataset shows significant class imbalance, with "Orange" being the most represented and "Pineapple" and "Watermelon" underrepresented, potentially biasing the model toward better performance on dominant classes. Bounding boxes indicate most objects share similar sizes and aspect ratios, with a dense cluster near smaller dimensions, highlighting the focus on detecting small objects. The heatmap reveals a central bias in object placement, which could reduce the model's effectiveness in detecting fruits in peripheral areas. Overall, the consistent aspect ratios and size variability emphasize the need for high precision in object detection methods.

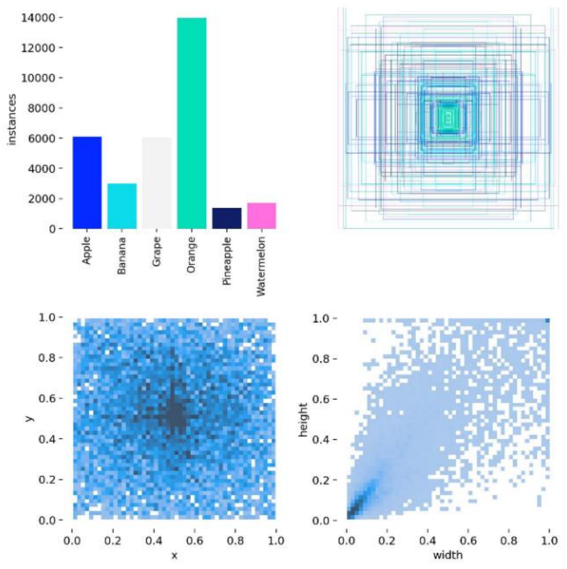


Figure 11. *Multilabel Fruits Detection [6] bounding boxes details visualization. Distribution bar graph (top-left) overlapped bounding boxes (top-right), positions heatmap (bottom-left), lengths scatter (bottom-right).*

3.2. Evaluation Metrics

3.2.1. Intersection over Union

The **Intersection over Union (*IoU*)** measures the overlap between the predicted bounding box (B_p) and the ground truth bounding box (B_g). It is defined as:

$$IoU = \frac{|B_p \cap B_g|}{|B_p \cup B_g|},$$

where $|B_p \cap B_g|$ is the area of overlap between the predicted and ground truth boxes and $|B_p \cup B_g|$ is the area of their union. *IoU* is used to determine if a predicted box is a true positive (e.g., if $IoU \geq 0.5$) or a false positive. Figure 12 illustrates how this metric is computed for 2 bounding boxes.

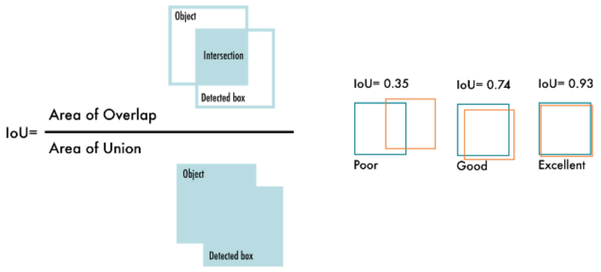


Figure 12. *The computation of the Intersection over Union (IoU) metric [1].*

3.2.2. Classification based Metrics

For object detection, these metrics are defined based on the predicted and ground t ruth bounding boxes at a chosen *IoU* threshold:

- **Precision**: The proportion of correctly predicted objects among all the predictions.

$$Precision = \frac{TruePositives(TP)}{TruePositives(TP)+FalsePositives(FP)}$$

- **Recall**: The proportion of correctly predicted objects among all ground truth objects.

$$\boldsymbol{Recall} = \frac{TruePositives(TP)}{TruePositives(TP)+FalseNegatives(FN)}$$

- **F1 Score**: The harmonic mean of precision and recall.

$$\boldsymbol{F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}}$$

These metrics are typically computed at a specific *IoU* threshold, such as $IoU \geq 0.5$

3.2.3. Classification Curves

Classification curves evaluate how performance metrics change as a function of the model's confidence thresholds (or against other metrics).

- **Precision-Confidence Curve**:

$$Precision\left(C\right) = \frac{TP(c)}{TP(c)\ +\ FP(c)}$$

- **Recall-Confidence Curve**:

$$Recall\left(C\right) = \frac{TP(c)}{TP(c)\ +\ FN(c)}$$

- **F1-Confidence Curve**:

$$F1(C)= 2 \times \frac{Precision(c)\times Recall(c)}{Precision(c)+Recall(c)}$$

- **Precision-Recall Curve**: This is constructed by plotting Precision against Recall at different thresholds. The area under this curve (**AUC**) is used to calculate the **Average Precision** (**AP**). This curve provides a detail view of the trade-off between false positives and false negatives.

3.2.4. Mean Pixel Accuracy

The **Mean Average Precision** (**mAP**) is the average of the **Average Precision** (**AP**) values computed over different object classes. AP is computed as the area under the **Precision-Recall curve** (Section 3.2.3) for a given threshold. The general formula is:

$$\int_0^1 Precision(r)dr,$$

where *r* is the recall. The **mAP50** is one of the most common metrics for evaluating the accuracy of object detectors and is the **mAP** computed only for $IoU \geq 0.5$. The **mAP50-95** provides a more comprehensive view of the model's performance across different levels of detection difficulty by averaging AP across multiple *IoU* thresholds (from 0.5 to 0.95, in steps of 0.05):

$$mAP_{50-95} = \frac{1}{10} \sum_{t=0.5}^{0.95} AP_t,$$

where AP_t represents the *AP* at a specific *IoU* threshold *t*.

3.3. *Procedure*

This Ultralytics Python library [3] provides various sizes for the YOLOv8 model (as seen in Figure 13), but the medium version was chosen to keep a balance between the model's inference time and accuracy. This model has 295 layers and was initially trained on the COCO MS dataset. Some of the last layers are removed in order to match the number of classes for each dataset. One of those layers was frozen to be able to stabilize the DFL loss function (Section 2.3).

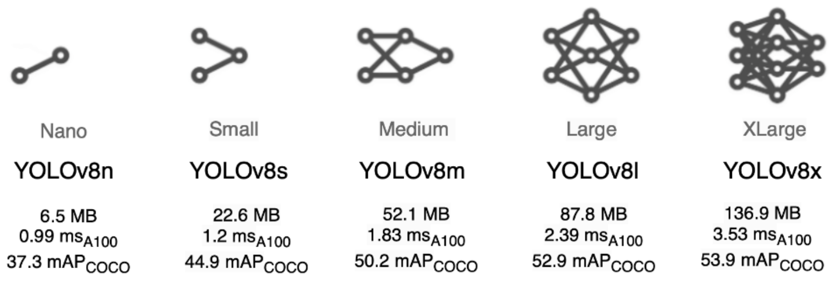


Figure 13. *Comparison between various YOLOv8 sizes [3].*

For each experiment, the model was trained for 100 epochs on a device with NVIDIA RTX 4090 GPU with 24GB VRAM. The resolution of the images during the training was 640x640x3, the batch size was 32, the learning rate was 0.01, and the optimizer used was Adaptive Moment Estimation (Adam). The Mosaic data augmentation technique [9] was applied during training for all the epochs except the last 10 to improve the stabilization of the model [9].

After each epoch, L_{box}, L_{cls}, and L_{dfl} losses (Section 2.3) were measured on both the training and the validation dataset, as well as the precision, recall (Section 3.2.2), mAP50 and mAP50-95 (Section 3.2.4) metrics were measured for the validation set.

At the end of the epochs, the Precision-Confidence, the Recall-Confidence, the F1-Confidence, respectively the Precision-Recall curves (Section 3.2.3) were measured on the validation dataset, the metrics along the epochs were plotted, and the confusion matrix for each class was plotted.

4. RESULTS AND ANALYSIS

4.1. *SkyFusion Results*

4.1.1. *Qualitative Results*

Some qualitative results can be seen in Figure 14. Predictions for "aircraft" and "vehicles" often include overlapping boxes, some misaligned with the ground truth, leading to false positives, while some ground truth objects, like "vehicles" or "ships," are missed, resulting in false negatives. Confidence values indicate the model's certainty, with higher scores reflecting greater prediction confidence.

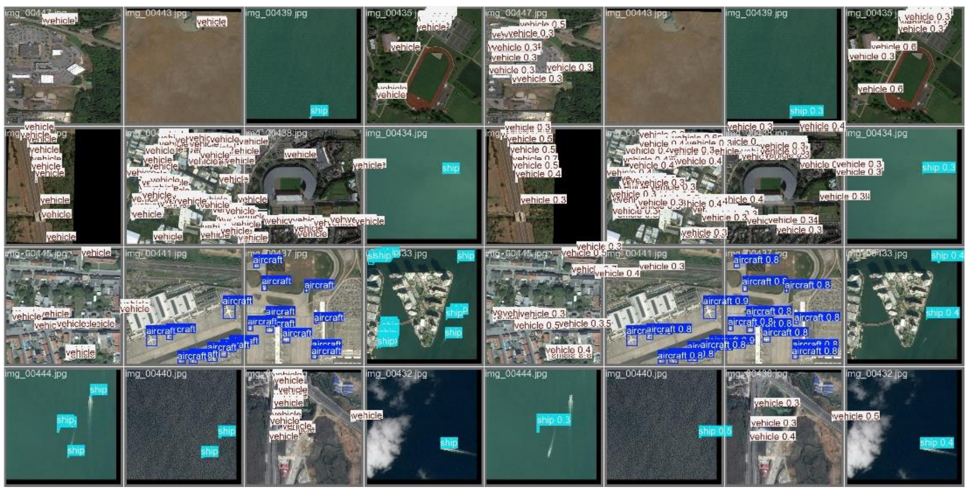


Figure 14. *SkyFusion ground truth data from validation set (left) vs predictions made by YOLOv8m model with their corresponding confidence levels (right).*

4.1.2. Quantitative Results

In Figure 15 and Figure 16 are plotted the classification curves. It can be observed that Precision increases with confidence, peaking at 1.0 at a threshold of 0.876, while recall decreases, with aircraft maintaining high recall and ships and vehicles showing significant losses at moderate thresholds. F1 scores peak for aircraft, indicating a good balance of precision and recall, whereas ships and vehicles yield weaker performance. Overall, aircraft performs excellently (AUC 0.957), but the mAP50 of 0.610 highlights room for improvement in underperforming classes.

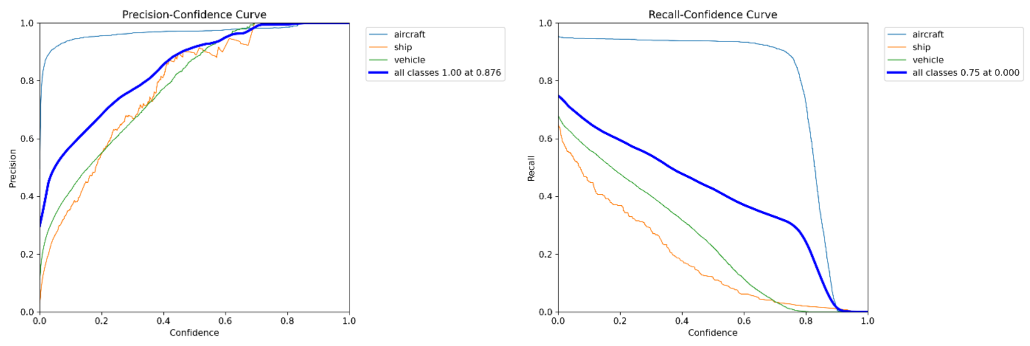


Figure 15. *SkyFusion Classification Curves: Precision-Confidence (left) and Recall-Confidence Curve (right).*

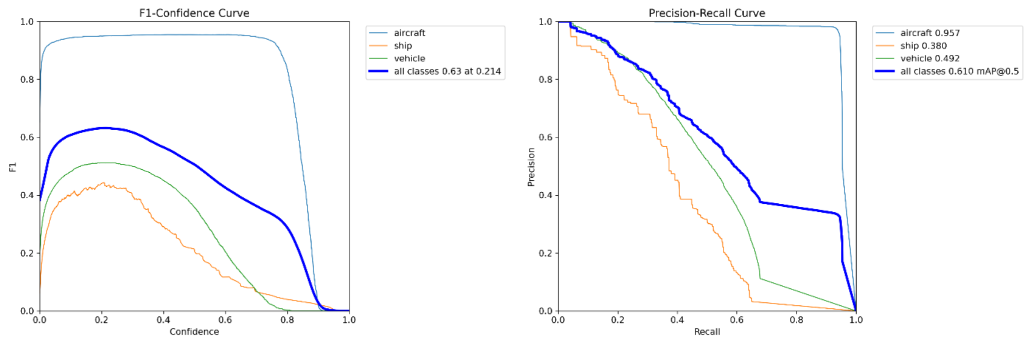


Figure 16. *SkyFusion Classification Curves: F1-Confidence (left) and Precision-Recall Curve (right).*

In Figure 17 the evolution of the losses (on both the training set and the validation set), and the evolution of the metrics (computed on the validation set) across the training epochs were plotted. We can observe that train and validation losses decrease steadily, indicating effective learning, while precision and recall improve over time despite occasional fluctuations due to false positives or negatives. Mean Average Precision (mAP) shows consistent growth, with mAP50-95 (a stricter metric) increasing, reflecting the model's enhanced ability to localize objects with higher confidence.

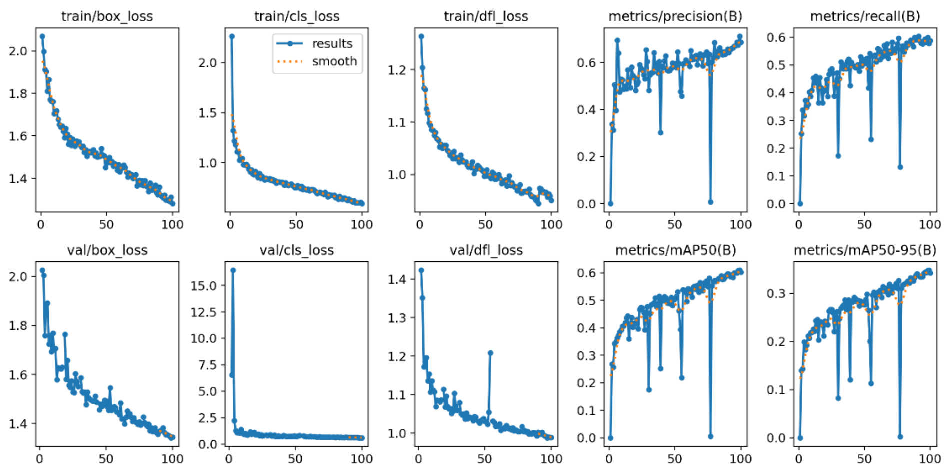


Figure 17. *Skyfusion losses and metrics evolution across the training epochs*.

In Figure 18, the confusion matrix on the validation set was plotted at the end of the training. Aircraft detection is strong, with high precision and minimal misclassifications, while ship detection is the weakest, showing low true positives and significant confusion with vehicles and background. Vehicle detection achieves high true positives but suffers from notable misclassifications as background, highlighting the need for improved recall. Background classification is generally accurate but shows some overlap with vehicles and ships, suggesting a need for better separation.

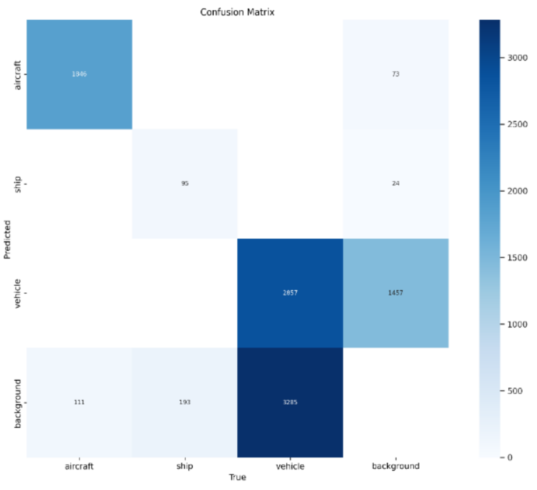


Figure 18. *SkyFusion Confusion Matrix*.

4.2. Traffic Signs Detection Results

4.2.1. Qualitative Results

From a qualitative point of view, some conclusions that we can draw from the Figure 19 are the fact that the model performs well overall, accurately detecting and classifying most traffic signs with high confidence scores (e.g., 0.9 or 1.0) and well-aligned bounding boxes. However, challenging conditions like low lighting, occlusions, or overlapping signs cause lower confidence or slight inaccuracies, particularly for cases like "Red Light" and "Green Light."

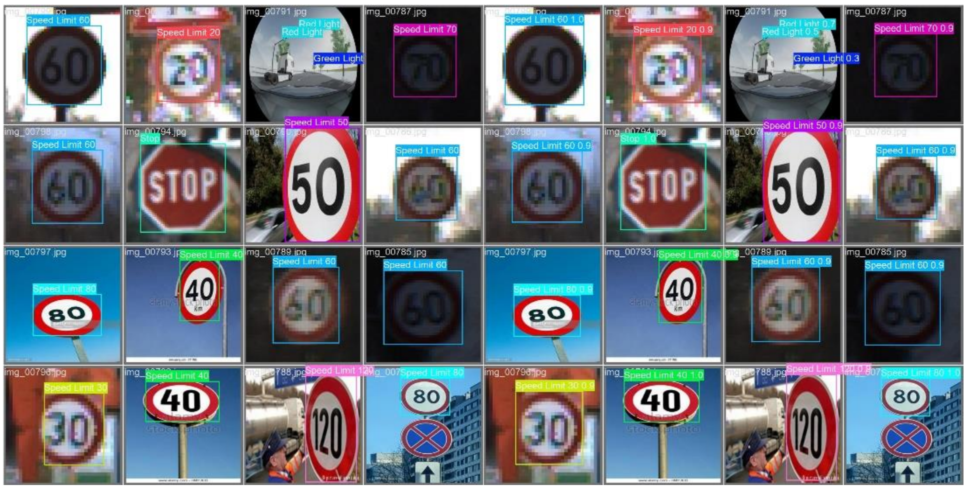


Figure 19. *Traffic Signs Dataset ground truth data from validation set (left) vs predictions made by YOLOv8m model with their corresponding confidence levels (right).*

4.2.2. Quantitative Results

For the Classification Curves, some conclusions that we can observe from the graphs in Figure 20 and Figure 21 are the fact that the precision improves steadily with increasing confidence thresholds, peaking at 1.0 for all classes, with "Speed Limit" performing consistently well, while "Red Light" and "Green Light" show more variability. Recall decreases at higher thresholds but remains high across most classes, supporting good detection coverage. F1 scores peak at moderate thresholds (~0.6), reflecting a balance between precision and recall, with strong overall performance (mAP50 of 0.965) but slightly lower results for "Red Light" and "Green Light" compared to "Speed Limit" and "Stop" classes.

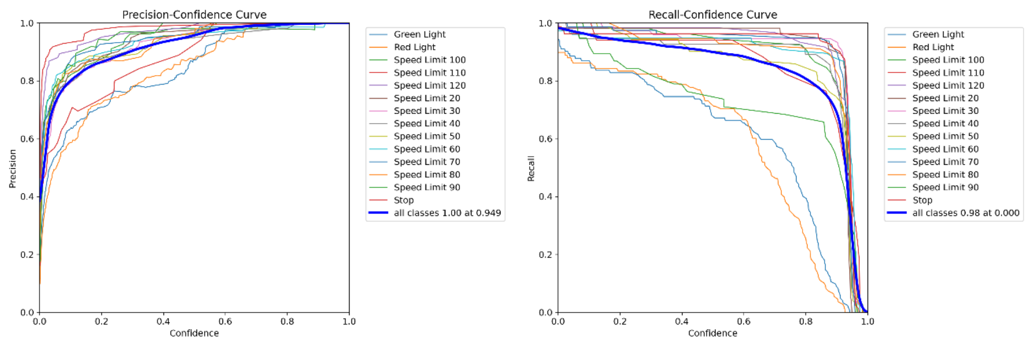


Figure 20. *Traffic Signs Dataset Classification Curves: Precision-Confidence (left) and Recall-Confidence Curve (right).*

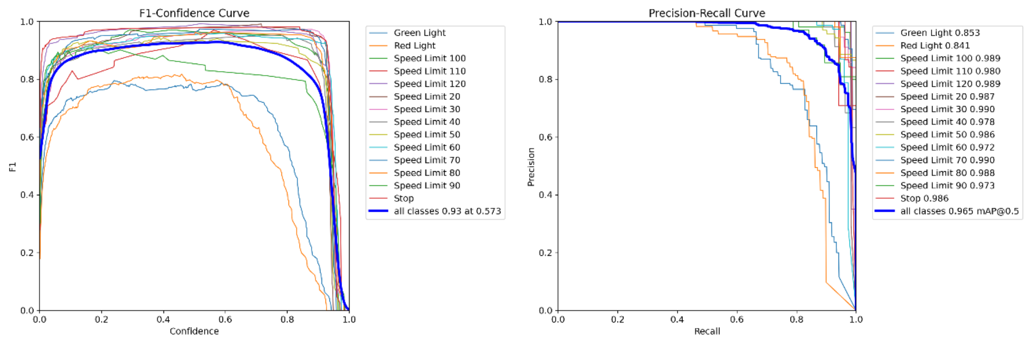


Figure 21. *Traffic Signs Dataset Classification Curves: F1-Confidence (left) and Precision-Recall Curve (right).*

For the evolution of the losses (for training and validation datasets) and the metrics (for the validation dataset) seen in Figure 22, we can conclude that the training and validation losses, including box, classification, and DFL losses, decrease steadily, indicating consistent learning and stable optimization without significant fluctuations. Precision and recall improve consistently over epochs, with mAP50 nearing 1.0 and mAP50-95 steadily rising, reflecting strong overall detection performance.

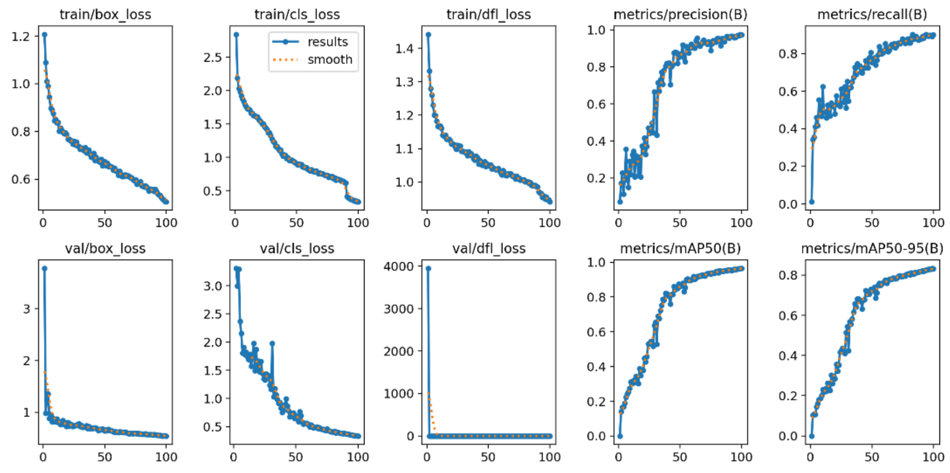


Figure 22. *Traffic Signs Dataset losses and metrics evolution across the training epochs*.

Lastly, the confusion matrix from Figure 23 reveals about the model's performance that the model performs strongly for classes like "Green Light," "Red Light," and most "Speed Limit" signs (e.g., "Speed Limit 30" and "Speed Limit 70") with high true positive counts and minimal confusion. However, it struggles with misclassifications among visually similar classes, such as "Speed Limit 50" and "Speed Limit 40," and occasionally misclassifies "Green Light" and "Red Light" as background under challenging conditions like poor lighting or occlusion. Rare classes, such as "Speed Limit 10" and "Speed Limit 80," show higher misclassification rates, reflecting the impact of dataset imbalance.

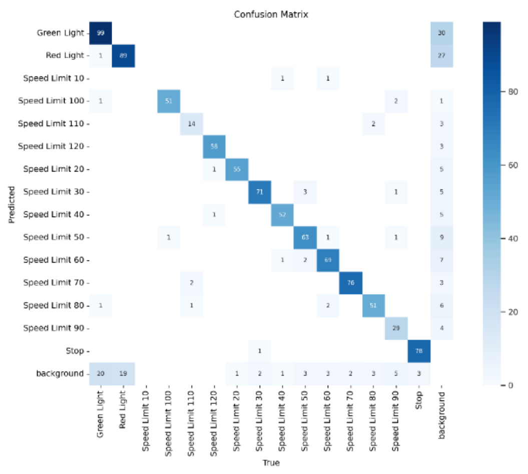


Figure 23. *Traffic Signs Dataset Confusion Matrix*.

4.3. Multilabel Fruits Detection Results

4.3.1. Qualitative Results

Some qualitative conclusions can be observed between the ground truth and predictions in the two images from Figure 24 are the fact that the model performs well for dominant classes like "Orange" and "Banana," accurately detecting most instances with high confidence, though confidence drops in cluttered or complex scenarios. Rare classes like "Pineapple" and "Watermelon" show inconsistent predictions and missed detections due to dataset imbalance. While bounding boxes are generally well-aligned with the ground truth, the model struggles in crowded or occluded scenes, such as overlapping "Grape" clusters, indicating a need for improved feature extraction.

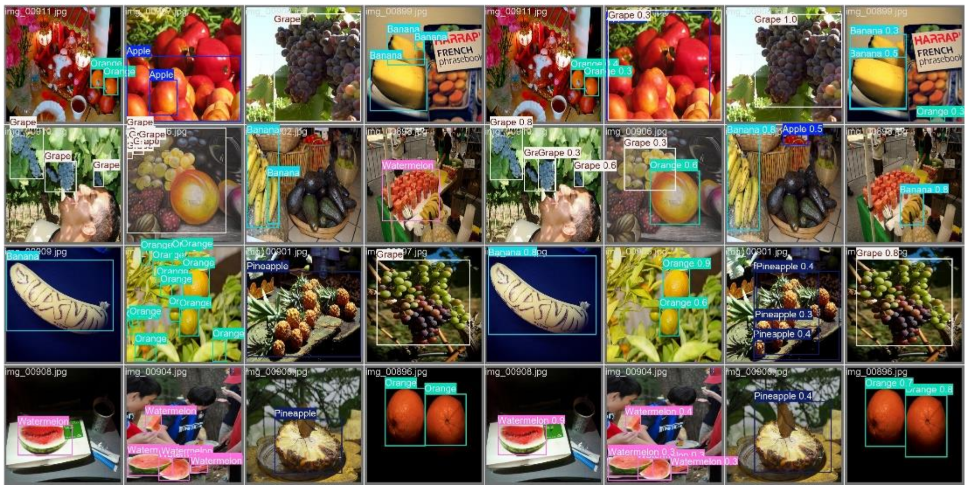


Figure 24. *Multilabel Fruits Dataset ground truth data from validation set (left) vs predictions made by YOLOv8m model with their corresponding confidence levels (right).*

4.3.2. Quantitative Results

For the quantitative analysis of the Classification Curves from Figure 25 and Figure 26, some the insights can be concluded are that the precision improves steadily with higher confidence thresholds, peaking near 1.0 for all classes, particularly "Watermelon," which also maintains relatively high recall at stricter thresholds. F1 scores peak at moderate thresholds (~0.5), with "Watermelon" achieving the best balance of precision and recall, while rare classes like "Pineapple" and "Grape" show significant drops in performance. The overall mAP50 of 0.493 reflects moderate performance, with stronger results for prominent objects like "Watermelon" (0.645) and weaker outcomes for rare or visually similar classes like "Grape" (0.395).

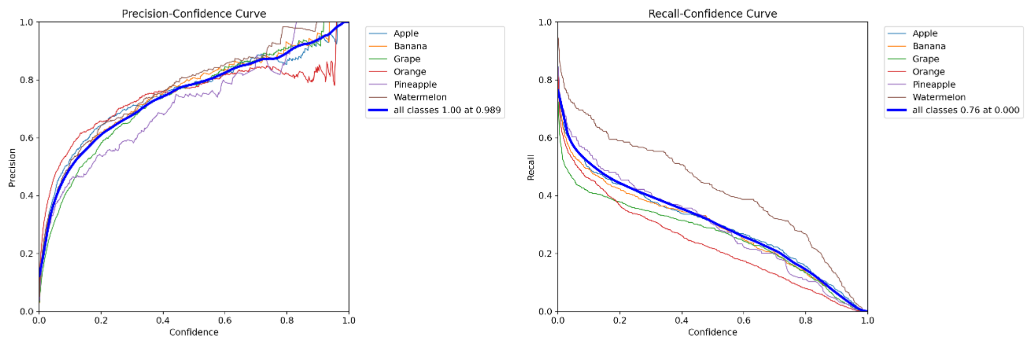


Figure 25. *Multilabel Fruits Dataset Classification Curves: Precision-Confidence (left) and Recall-Confidence Curve (right).*

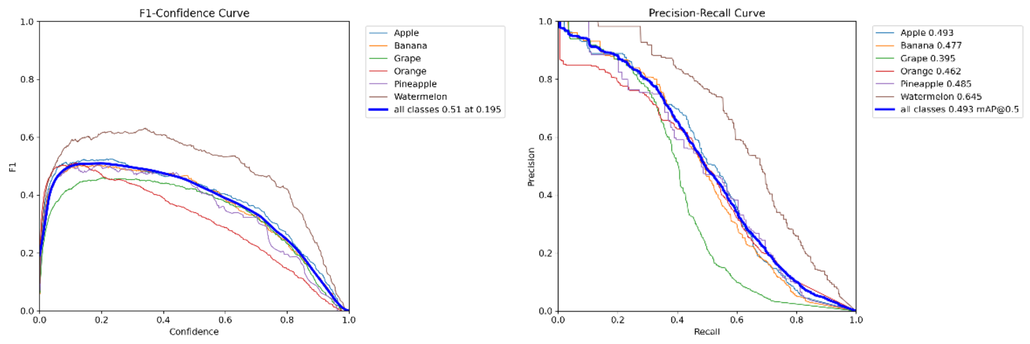


Figure 26. *Multilabel Fruits Dataset Classification Curves: F1-Confidence (left) and Precision-Recall Curve (right).*

By looking at the losses (computed across the training dataset and the validation dataset) in Figure 27, some conclusions that can observed are that the Training and validation losses, including box, classification, and DFL losses, decrease steadily over epochs, indicating effective learning and stable optimization despite slight fluctuations in validation data. Precision and recall improve consistently, with mAP50 and mAP50-95 steadily increasing, reflecting enhanced detection capability. However, the moderate mAP values highlight challenges in achieving uniformly high performance across all classes.

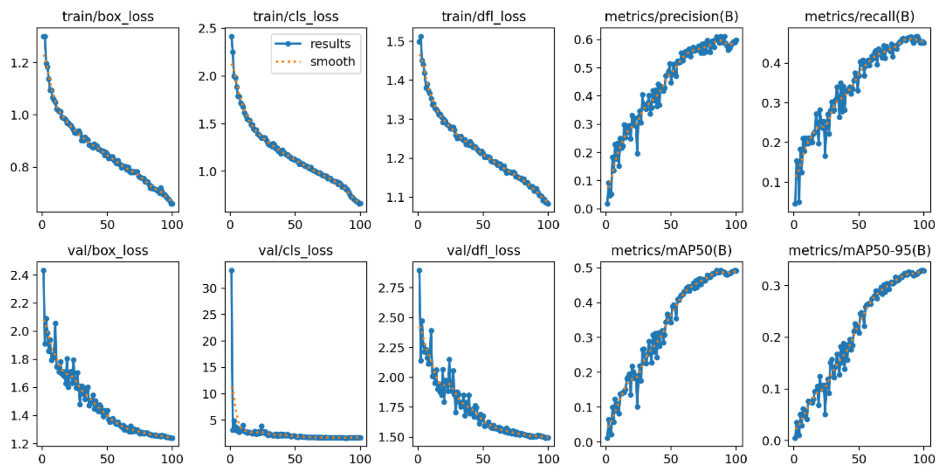


Figure 27. *Multilabel Fruits Dataset losses and metrics evolution across the training epochs*.

Some insights captured by the Confusion Matrix in Figure 28 are that the model performs well for dominant classes like "Orange" and "Grape," with high true positives and minimal misclassifications, reflecting their overrepresentation in the dataset. Rare classes like "Pineapple" and "Watermelon" show higher misclassification rates and lower true positives, highlighting the impact of dataset imbalance. Additionally, significant background confusion, particularly for "Apple" and "Grape," and occasional inter-class misclassifications (e.g., "Banana" as "Apple") suggest a need for more refined feature extraction to better distinguish visually similar objects.

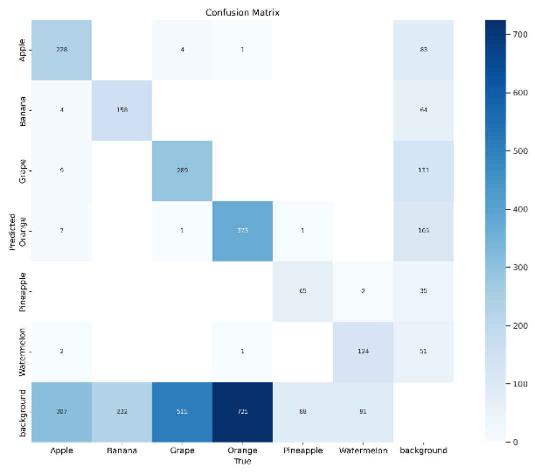


Figure 28. *Multilabel Fruits Dataset Confusion Matrix*.

4.4. Comparison between the results of each dataset

From Table 1 where the results of all the studied datasets are presented, we can conclude that, keeping in mind the distributions of the data represented in the figures in Section 3.1, the **SkyFusion** dataset suffers from significant class imbalance, with the "vehicle" class dominating and "ship" being underrepresented, contributing to moderate precision (0.686) and recall (0.588). The small object sizes and narrow bounding boxes further challenge precise detection, resulting in a low mAP50+90 (0.342), though the uniform spatial distribution of objects avoids positional bias. In contrast, the **Traffic Signs** dataset benefits from a relatively balanced class distribution, enabling high precision (0.972) and recall (0.899). Moderate object sizes and centralized positions simplify detection, reflected in strong mAP50 (0.965) and mAP50+90 (0.832), while the distinct visual features of traffic signs enhance model performance. The **Multilabel Fruits** dataset, however, faces severe class imbalance, with dominant classes like "Orange" overshadowing rare classes such as "Pineapple" and "Watermelon," negatively affecting precision (0.598) and recall (0.450). The broad range of object sizes, including very small fruits, poses additional challenges, leading to lower mAP50 (0.491) and mAP50+90 (0.329), though centralized objects provide some consistency in spatial localization.

Dataset	Classes	Training Samples	Validation Samples	Precision	Recall	mAP50	mAP50+90
SkyFusion	3	2094 (82.3%)	450 (17.7%)	0.686	0.588	0.601	0.342
Traffic Signs	15	3530 (81.5%)	801 (18.5%)	**0.972**	**0.899**	**0.965**	**0.832**
Fruits	6	7108 (88.6%)	914 (11.4%)	0.598	0.450	0.491	0.329

Table 1. *Comparison of the metrics between all the 3 studied datasets*.

5. CONCLUSIONS AND FUTURE WORK

The performance of YOLOv8 varied across the three datasets, highlighting dataset-specific strengths and challenges. On the **SkyFusion dataset**, the model struggled with class imbalance and the small size of objects, leading to moderate precision and recall, while underrepresented classes like "ship" were particularly challenging. In contrast, the **Traffic Signs dataset** showcased YOLOv8's strongest performance, achieving high precision, recall, and mAP50 due to balanced class distribution and distinct visual features. The **Multilabel Fruits dataset** posed greater difficulties, with severe class imbalance and varying object sizes leading to lower precision and recall, especially for underrepresented classes like "Pineapple" and "Watermelon."

The study demonstrated YOLOv8's strengths in datasets with moderate object sizes, centralized spatial distribution, and balanced classes, as seen in traffic sign detection. However, it revealed limitations in handling tiny objects, class imbalances, and visually similar classes. Future research should focus on addressing these challenges through techniques like oversampling or data augmentation for underrepresented classes to improve detection robustness and generalization across diverse datasets.

Acknowledgement:
This work is the result of my own activity, and I confirm I have neither given, nor received unauthorized assistance for this work. I declare that I did not use generative AI or automated tools in the creation of content or drafting of this document.

REFERENCES

[1] Terven, J., Córdova-Esparza, D. M., & Romero-González, J. A. (2023). A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine Learning and Knowledge Extraction*, 5(4), 1680-1716.

[2] Redmon, J. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[3] Jocher, G., Qiu, J., & Chaurasia, A. (2023). Ultralytics YOLO (Version 8.0.0) [Computer software]. https://github.com/ultralytics/ultralytics

[4] Sudheer K.., Selvakumaran R. (2021). *SkyFusion: Aerial Object Detection* [Dataset]. Kaggle. https://www.kaggle.com/datasets/kailaspsudheer/tiny-object-detection

[5] Karimi D. (2023). *Traffic Signs Detection for Self-Driving Cars* [Dataset]. Kaggle. https://www.kaggle.com/datasets/pkdarabi/cardetection

[6] Tyagi L., Gunjal N. (2022). *Multilabel Fruits Detection* [Dataset]. Kaggle. https://www.kaggle.com/datasets/lakshaytyagi01/fruit-detection

[7] Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2019). *Object detection with deep learning: A review*. IEEE transactions on neural networks and learning systems, 30(11), 3212-3232.

[8] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). *Ssd: Single shot multibox detector*. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14 (pp. 21-37). Springer International Publishing.

[9] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). *Yolov4: Optimal speed and accuracy of object detection*. arXiv preprint arXiv:2004.10934.