End-to-End Object Detection on the RF100VL Dataset Using YOLO and RF-DETR

Frank Chen
Department of Computer Science, Viterbi School of Engineering, University of Southern California
Los Angeles, United States
chenyinu@usc.edu

Chenxi' Dong
Department of Computer Science, Viterbi School of Engineering, University of Southern California
Los Angeles, United States
chenxido@usc.edu

Darren Cao
Department of Computer Science, Viterbi School of Engineering, University of Southern California
Los Angeles, United States
darrenca@usc.edu

***Abstract*—This paper presents a comprehensive comparison between the CNN-based YOLOv12 detector and the transformer-based RF-DETR model on RF100VL, a multimodal dataset containing 564 classes and 164,149 images from 100 diverse sources. Building upon the reference study "RF-DETR Object Detection vs YOLOv12: A Study of Transformer-based and CNN-based Architectures for Single-Class and Multi-Class Greenfruit Detection in Complex Orchard Environments Under Label Ambiguity," which focused on orchard scenes with limited category diversity, this work examines how both models generalize to a far more heterogeneous environment. We construct full preprocessing, conversion, and evaluation pipelines for both models, including unified COCO/YOLO annotation handling, segmentation processing, zero-shot evaluation, finetuning, and hyperparameter optimization. Experimental results show that RF-DETR achieves higher mAP@50–95 on RF100VL, while YOLOv12 remains competitive at moderate IoUs. These findings highlight how architectural differences manifest under complex multimodal conditions. (*Abstract*) Keywords—YOLOv12, RF-DETR, object detection, multimodal dataset, RF100VL, transformers, CNNs. (**key words**)**

I. INTRODUCTION

Modern object detection has advanced through improvements in both convolution-based and transformer-based architectures. CNN detectors such as the YOLO family emphasize efficiency and strong inductive priors, whereas transformers like DETR and RF-DETR provide global attention mechanisms that capture long-range spatial structures. Understanding how these architectural choices affect performance on diverse real-world data remains an open question.

A prior study titled *"RF-DETR Object Detection vs YOLOv12: A Study of Transformer-based and CNN-based Architectures for Single-Class and Multi-Class Greenfruit Detection in Complex Orchard Environments Under Label Ambiguity"* compared the two models in orchard environments characterized by occlusion, clustered objects, and label ambiguity [1]. While informative, the study was limited by its narrow domain and very small number of object categories [1]. Many real-world settings demand robustness to large category sets, significant domain shift, and multimodal variation.

To address this gap, we evaluated YOLOv12 and RF-DETR on the RF100VL dataset, a considerably more challenging benchmark spanning 564 classes and multiple imaging modalities [2]. This paper presents unified data preprocessing, annotation conversion into COCO and YOLO formats, and reproducible training and evaluation workflows for both detectors. By analyzing both pretrained and finetuned performance, we provide a detailed view of how CNN and transformer-based models adapt to multimodal conditions.

II. RELATED WORK

Object detection has evolved from anchor-based CNN architectures to transformer-based frameworks capable of modeling global spatial relationships. The YOLO series has progressively integrated more advanced backbone and attention components, culminating in YOLOv12. Transformer-based detectors, beginning with DETR and further improved by DN-DETR, DINO, and RF-DETR, have reduced reliance on handcrafted components and enhanced generalization across domains. The reference greenfruit detection study concluded that RF-DETR handles occlusion and ambiguous labels more effectively, whereas YOLOv12 performs efficiently in structured environments [1]. However, its conclusions were constrained by the limited scope of orchard imager, which had just two classes in its multi-class approach, occluded and non-occluded [1]. Unlike this domain-specific dataset, RF100VL provides a broader testbed for examining model robustness and generalization across varied modalities.

III. DATASET OVERVIEW

A. RF100VL Dataset Description

The RF100VL dataset provides the foundation for all experiments in this study. It contains 164,149 images and 564 object categories collected from 100 heterogeneous datasets spanning RGB photography, thermal and infrared imagery, microscopy, X-ray scans, and industrial inspection domains [2]. This broad coverage introduces extensive variation in lighting,

object scale, background complexity, sensor modality, and visual style. Compared with orchard-specific datasets used in prior studies, RF100VL presents far greater multimodal diversity and severe domain shifts, making it an ideal setting for evaluating detector robustness and generalization. The data was downloaded using a RoboFlow API call in CARC.

B. Characteristics and Challenges

RF100VL exhibits a highly imbalanced long-tail distribution, where many categories contain only a limited number of annotated instances. Because the dataset originates from numerous independent sources, annotation density, labeling style, and image resolution differ significantly across subsets [2]. These factors introduce challenges beyond conventional object detection benchmarks, including inconsistent visual patterns, ambiguous boundaries, and nonuniform annotation quality. Such conditions stress-test the ability of detection architectures to learn stable representations under multimodal noise and imbalance.

IV. KEY CONTRIBUTIONS AND ACCOMPLISHMENTS

Our major accomplishments include:

1. Successfully convert the RF100VL annotations into both YOLO and COCO structures

2. Build a zero-shot evaluator to establish baseline performance on the RF100VL dataset, something the reference paper does not do

2. Successfully construct two full-feature training and evaluation pipelines for YOLOv12 and RF-DETR

3. Optimize hyperparameters specific to both YOLOv12 and RF-DETR rather than use same hyperparameters in both as the reference paper does

5. Establish consistent evaluation metrics (mAP@50, mAP@50-95, precision, recall, F1)

V. YOLOv12 APPROACH AND EXPERIMENTS

A. Introduction

The YOLOv12 model is the newest version of the YOLO object detection models. It is a CNN-based object detection model that shares the same backbone, head, and neck architecture as its predecessors [1]. However, YOLOv12 replaces many of the convolutional-heavy architectures of previous versions with attention mechanisms and an R-ELAN backbone [3]. YOLOv12's key innovation is Area Attention, which instead of having the model look at every pixel at once, looks at regions of the image and learns each of the segments, which improves the speed of the attention mechanism [3]. The R-ELAN backbone helps the model learn better by improving the way information flows through the network, making training more stable and efficient, particularly for larger versions of the model [3]. The YOLOv12 model comes in five sizes, YOLOv12-N, YOLOv12-S, YOLOv12-M, YOLOv12-L, and YOLOv12-X. The sizes increase from YOLOv12-N to YOLOv12-X by the number of parameters, so larger models have better mAP measurements while sacrificing speed [3]. Based on the tradeoff between mAP, we determined YOLOv12L was the optimal model for this paper with a mAP:50-95 of 53.7% on the COCO dataset and a FLOPs (G) of 88.9 per image [3].

B. Data Preprocessing

The YOLOv12L model preprocessing consists of two main stages. The first stage takes all the COCO-style JSON annotations files and extracts the image, paths, dimensions, bounding boxes, and segmentation polygons to merge them into a single CSV file. It also builds a JSON label-mapping file to map class names to integer label ids for consistent label usage. Furthermore, a PyTorch dataset converts the COCO boxes into corner coordinates (x_1, y_1, x_2, y_2) and when segmentations are available, converts the polygon shapes into pixel-based masks for the model.

After the COCO CSV annotations have been merged into one file, it can then be converted into the standard YOLO detection format. Each COCO GT (ground truth) bounding box is transformed into YOLO coordinates (x_{center}, yc_{enter}, width, height) with $x_{center} = (x + w / 2)$ and $y_{center} = (y + h / 2)$. w and h are the bounding box width and height. All coordinates are then normalized by being divided by the image width and height. This ensures all coordinates are within a 0 to 1 range, which is what YOLO expects for bounding-box coordinates [4]. These coordinates are written into txt files next to each image so the model can find them for evaluation. The script then preserves the training, validation, and test splits and images from the original directory.

C. Non-Finetuned Evaluator

Unlike the reference paper, this approach establishes baseline metrics for YOLOv12L by evaluating a base YOLOv12L model on the RF100VL without finetuning it to the dataset for zero-shot values. In this evaluation, the normalized YOLO coordinates from data preprocessing are converted into the corner coordinates for the bounding box. First, the normalized coordinates are converted into pixel center coordinates by multiplying them by the image width W and height H. Then, the x-center and y-center coordinates are converted into corner coordinates, with the two corner x-coordinates being the pixel center x-coordinate added or subtracted to the pixel center width coordinate and the two corner y-coordinates being the pixel center y-coordinate added or subtracted to the pixel center height coordinate. Now, with the bounding boxes established, IoU-based comparison can be done by having each image being passed through the model to generate predicted boxes and confidence scores. The evaluator then computes the intersection over union between the predictions and GT boxes to determine precision, recall, F1-score, and mAP across IOU thresholds from 0.50 to 0.95. Note, class labels were not considered for this evaluation due to YOLOv12L not knowing any of the RF100VL class names or ids. Furthermore, Optuna, which is a hyperparameter optimization framework utilizing Bayesian optimization, is done for 20 trials to optimize the confidence threshold and NMS IoU. Confidence threshold is the minimum

score a predicted bounding box must have for the model to keep it while NMS IoU controls how much overlap is allowed between predicted boxes to remove duplicate detections. The optimal confidence threshold found was 0.054 while the optimal NMS IoU was 0.484 as that combination gave the highest validation mAP@50:9 for the model.

D. Finetuned Evaluator

After establishing a zero-shot baseline, the YOLOv12L model was then finetuned on the RF100VL training set. The evaluation metrics used were the same as the ones used for the non-finetuned evaluator except now predicted class labels were considered a part of the evaluation due to the YOLOv12L now knowing the class labels having been finetuned on the dataset. A batch size of 16 was used for a good balance of stability and memory usage. Other traditional model hyperparameters such as initial learning rate, weight decay, and warmup momentum were found utilizing Optuna. However, a large portion of the hyperparameters tuned by Optuna were data augmentation hyperparameters. These hyperparameters helped the model adapt to different color distributions and pixel intensities, the spatial arrangement, orientation, and scale of objects, and even combined different images into one to help with object detection in crowded spaces. The default optimizer, which is AdamW for the first 10,000 iterations before switching to SGD, was used.

The finetuned evaluator consisted of two stages. In the first stage, the first 10 layers, which make up most of the backbone, were frozen for 20 epochs. Freezing these layers was beneficial since the backbone captures low-level visual features such as edges, textures, and basic shapes that are already well learned from the COCO training, so retraining initially was not necessary. Instead, training in stage 1 was largely done on the detection head and neck, which are responsible for learning RF100VL-specific object structures and classes and making the actual predictions. In the second stage, the first 10 layers were unfrozen, and training was done for 60 epochs. This is because once the detection head and neck have been adapted to the RF100VL dataset in stage 1, the backbone can now be safely trained without forgetting what it had learned. Furthermore, due to the RF100VL dataset being larger than the COCO dataset, there are likely new shapes, textures, and overall image patterns specific to RF100VL that need to be learned by YOLOv12L. Overall, this two stage approach ensures YOLOv12L does not forget its pretraining from COCO while adapting to the RF100VL dataset for stronger overall detection performance.

E. Finetuning Hyperparameter Optimization

Optuna hyperparameter optimization was once again done to find the best performing hyperparameter combinations on the validation set to be used for evaluation on the test set. The hyperparameters search space for stage 1 was based on the Ultralytics own tuning guide [5]. The optimal hyperparameters are shown in figures A1 and A2 in the appendix (Fig. A1 and A2). Optuna was done for 10 trials in both stage 1 and stage 2. After the optimal hyperparameters were found in stage 1, eleven of the model and data augmentation hyperparameters were reused in stage 2. The other nine hyperparameters, including the initial learning rate, weight decay, mosaic, degrees, and translate were reoptimized in stage 2. These new hyperparameter values besides warmup epochs were all smaller in stage 2 compared to stage 1 to adapt to finetuning of the full model without destabilizing the pretrained backbone and allowing the network to gradually adjust to the new dataset for improved overall performance.

VI. RF-DETR APPROACH AND EXPERIMENTS

A. Introduction

RF-DETR (Roboflow's Detection Transformer) is a state-of-the-art, real-time object detection and instance segmentation model developed by Roboflow. It is the first real-time model to achieve over 60 mAP (mean Average Precision) on the standard Microsoft COCO benchmark and excels in adapting to diverse, real-world domains. Unlike the traditional CNN-based model, it uses an end-to-end transformer architecture that produces final predictions directly, simplifying the pipeline and improving efficiency. The model is pre-trained with a powerful DINOv2 backbone on a massive, uncurated dataset, giving it a rich and general understanding of visual patterns. This allows it to adapt quickly and effectively to new or specialized datasets with limited fine-tuning data, outperforming models that struggle with out-of-distribution classes [6].

B. Data Preprocessing

RF-DETR requires a very different preprocessing pipeline than YOLO. Before finetuning RF-DETR Medium, we must merge all 100 RF100 sub-datasets into a single, clean COCO-formatted dataset with unified train, valid, and test splits. Each split needs its own images and a single annotation JSON file. To do this, we wrote a script that creates a fresh output directory, removes old annotations, and assigns every image and annotation a new globally unique ID to avoid collisions across datasets. The script scans each RF100 folder, loads its split-specific JSON, copies images into the merged directory, and rewrites annotation IDs and image IDs so they align with the new index space. All annotations for each split are appended into one consolidated COCO file. After processing all datasets, we also build a unified, alphabetically sorted category list and insert it into every final annotation file. Because RF-DETR uses COCO-style labels without internal remapping, it requires category IDs to be zero-indexed and contiguous (0…K-1). If categories use IDs like 1–100, the transformer's output indices no longer match the annotation labels. Therefore, after merging the RF100 dataset, we renumber all categories to start at 0 and increase sequentially to ensure RF-DETR trains correctly.

C. Non-Finetuned Evaluator

To understand how well the pretrained RF-DETR model performs before any finetuning on RF100, we manually evaluated the non-finetuned RF-DETR Medium checkpoint on the full test split of our merged RF100 dataset. We first loaded the official COCO-pretrained RF-DETR model and parsed all

ground-truth annotations from the RF100 COCO JSON file. For each test image, we ran a forward pass with a very low confidence threshold so that every prediction would be included in the evaluation, then converted RF-DETR's outputs into XYXY format and sorted them by confidence. For every predicted box, we computed its IoU with each ground-truth box and matched them using a greedy strategy at a chosen IoU threshold of 0.5, marking true positives only for unique ground-truth matches and counting the remaining predictions as false positives. Any ground-truth boxes left unmatched were counted as false negatives. By accumulating the totals across all images, we calculated the overall precision, recall, and F1-score of the non-finetuned model, and saved these metrics to a JSON file (Table B1) This evaluation gave us a baseline understanding of how much performance improvement the finetuning provided over the raw model.

D. *Finetuning the Model*

The next step is to finetune the model using the RF-DETR Medium pretrained weight [7]. For training, we used a customized RF-DETR Medium configuration designed to balance performance, GPU memory limits, and training speed on CARC. The model was trained for 10 epochs, which is long enough for RF-DETR to stabilize without overfitting or exceeding cluster time limits. A batch size of 2 was selected because transformer-based detectors are memory-intensive, and RF-DETR Medium typically fits only batches of size 1–2 on most NVIDIA GPUs. To maintain an effective batch size of 16, we used gradient accumulation with 8 steps, which improves training stability without increasing GPU memory. The learning rate of 5e-5 is a standard value for finetuning transformer backbones and avoids divergence on large, complex datasets like RF100. We used two data-loading workers to avoid CPU overload on CARC and enabled mixed precision (amp=True) to speed up training while saving memory. Finally, the model evaluated every epoch and automatically stores the highest-performing checkpoint based on validation mAP. Overall, these parameter choices allowed RF-DETR Medium to train efficiently on the full RF100 dataset while maximizing model accuracy within hardware constraints.

VII. EVALUATION

A. Metrics Used

The primary performance metric used was mean Average Precision (mAP), which is the area under the precision-recall curve. The formula for precision is:

$$precision = \frac{TP}{TP + FP} \quad (1)$$

The formula for recall is:

$$recall = \frac{TP}{TP + FN} \quad (2)$$

The standard mAP used by COCO is 101 interpolated AP, which has the following formula:

$$AP = \frac{1}{101}\sum_{k=0}^{100} \max_{r \geq k/100} P(r) \quad (3)$$

F1-score was also calculated for the YOLOv12L model with the equation:

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \quad (4)$$

B. YOLOv12

1) Validation Set

For the validation set, as expected, the finetuned model values improve significantly compared to the non-finetuned model (Table A1 and A2) for all metrics. The mAP@50:95 was roughly ten times higher for the finetuned model (Table A1 and A2).

One interesting trend is that the precision values are consistently higher than the recall values for all IoU for both models for the non-finetuned metrics but the opposite is true for the finetuned. This can be attributed to the confidence threshold being 0.054 and the NMS IoU of 0.484. While that confidence threshold seems low, it is high, especially for the non-finetuned model that will not have a great deal of confidence making predictions. Therefore, a lot of predicted bounding boxes will be removed as most of them will be made with low confidence. This will then increase the number of false negatives due to many objects not having a prediction for them, which leads to low recall. Though not as significant as a factor as the confidence threshold, the somewhat low NMS IoU threshold of 0.484 will remove many overlapping predictions, which will thus decrease the number of false positives, which leads to a higher precision value. After finetuning, recall becomes higher since the model produces more confident detections. This will result in less false negatives, which means a higher recall value in general.

The mAP values for the finetuned model are pretty in line with those found in a YOLOv12 reference paper. The validation mAP for YOLOv12L on COCO was 70.7% for mAP@50, 58.5% for mAP@75, and 53.7% for mAP@50:95 while it was 75.9% for mAP@50, 54.3% for mAP@75 and 49.3% at mAP@50:95 for the RF100VL dataset (Table A2) [3]. Note, the training on the COCO dataset in the paper was done for 600 epochs compared to just 80 epochs in total for the RF100VL dataset due to time and resource constraints [3]. The non-finetuned metrics are class-agnostic

2) Test Set

The test set results are very similar to the validation set results for both finetuned and non-finetuned values (Table A3 and A4). Specifically, for the finetuned model, the mAP@50 is 76.1%, the mAP@75 is 54.3%, and the mAP@50:95 is 49.4% (Table A4). For the test set, in addition to the metrics, images with the predicted and GT boxes, like the ones in figure A3 and A4, were generated for both the finetuned and non-finetuned model. As seen when comparing those images, the finetuned model shows significant improvement in

generating the predicted boxes (Fig. A3 and A4). The non-finetuned model only generates one predicted box over 5.4% confidence and it is not overlapping with any of the GT boxes (Fig. A3 and A4). In contrast, the finetuned model generates three predicted boxes for "plastic", "bottle", and "can" with IoU overlaps of 87.06%, 94.88%, and 88.54% (Fig. A3 and A4). These results confirm that finetuning the model results in significant object detection improvement. Here, the class prediction was considered when finding them.

C. RF-DETR

1) Validation Set

During validation, the RF-DETR Medium model demonstrated strong and stable performance across IoU thresholds and object scales. Based on the summary statistics of the EMA model at the end of the training(also the EMA model that is saved as the best total checkpoint to evaluate on the test split), the final EMA evaluation produced an mAP@50 of 0.7506, showing that the model correctly detects and classifies about 75% of objects under a moderate IoU requirement. The more challenging mAP@50–95 = 0.5121 indicates the model still performs well as the IoU threshold becomes stricter. The precision and recall shown here are the micro-averaged precision and recall across all classes and predictions. (Fig B1)

The COCO-style evaluation breaks average precision down by object *area* and *maxDets*. In COCO terminology, the "area" parameter groups objects by their size in pixels—*small*, *medium*, and *large*—; this allows us to see where the model is strongest. RF-DETR performs best on large objects (AP≈0.574), shows good performance on medium objects (AP≈0.426), and still handles small objects reasonably (AP≈0.295), which is typical for transformer-based detectors. Meanwhile, "maxDets" specifies the maximum number of predicted bounding boxes the evaluator is allowed to consider for each image (e.g., 1, 10, or 100), and the primary COCO Average precision metrics uses maxDets=100 as a standard limit (Table B2). Overall, both the EMA summary and the detailed AP metrics show that the RF-DETR Medium model generalizes well on the validation set, steadily improves across epochs, and it can capture precision in different object sizes—indicating it is well-trained and ready for test-set evaluation.

2) Test Set

The evaluation on the held-out test set shows results that closely mirror the validation performance, confirming that the model did not overfit and generalizes reliably to unseen data. Although the test evaluation only outputs Average Precision (AP) metrics, these are sufficient to compare performance across IoU thresholds and object sizes. The overall mAP@50–95 of 0.519 is slightly higher than the validation score, and the mAP@50 of 0.759 also surpasses the validation value. This pattern indicates that the model continues to perform consistently even on previously unseen images. When broken down by object size, the model achieves AP=0.322 for small objects, AP=0.468 for medium objects, and AP=0.581 for large objects, which is nearly identical to the trends observed during validation. Larger objects remain the easiest for the model to detect, small objects remain the most challenging, and medium objects fall in between—again matching the validation behavior. Overall, these test-set results reinforce that the RF-DETR Medium model trained on RF100 generalizes well and maintains stable accuracy across different object scales and IoU thresholds (Table B3). Finally, we performed a qualitative inference test using the first image from the test split ([8],[7]). The left image shows the ground-truth annotations, while the right displays the model's predicted detections. The predicted bounding boxes align closely with the GT across both large structural elements and smaller components, with confidence scores consistently exceeding 0.8. The model also successfully distinguishes between visually similar parts and remains robust under the scene's uneven lighting and reflective surfaces. This qualitative result is consistent with the quantitative metrics, further confirming that the finetuned RF-DETR model generalizes effectively to unseen samples in the RF100 test set. (Fig. B2)

VIII. CONCLUSION AND NEXT STEPS

Overall, the RF-DETR seemed to do a better job overall adapting to the RF100VL dataset as the finetuned RF-DETR model has a higher mAP@50-95 for the test set compared to the finetuned YOLOv12L model (51.9% vs 49.4%). The YOLOv12L does have a slightly higher mAP@50 (76.1% vs 75.9%), indicating that as the IoU threshold becomes stricter, the transformer-based RF-DETR model tends to perform better. In contrast, for the greenfruit object multi-class detection, the opposite trend occurs as the mAP@50 was higher for RF-DETR (0.946 vs 0.934) while the mAP@50:95 was higher for YOLOv12L (0.760 vs 0.743) [1]. The overall mAP values were higher on the Greenfruit dataset, but that is expected given there are only two classes in that dataset versus the 564 classes of the RF100VL dataset.

The challenges for the YOLOv12L implementation came from the computational resources used. Specifically, all the training and evaluation were done in CARC on whatever GPU was available (preferred was NVIDIA A100). The main challenge was the 48-hour SLURM job configuration limit. The training, especially for the 60 epochs in the second stage, often took longer than 48 hours, so we had to implement conditions in the code that would resume training from the last epoch if another SLURM job had to be submitted. Furthermore, 10 Optuna trials would take too long to run one after another, so they were run in parallel, with the selected hyperparameters and validation nresults per trial saved to a SQLite database. Here, we faced an issue of having two trials fail to run due to GPU limitations on the GPU its SLURM job had and parallel trials occasionally attempting to write to the SQLite database at the same time, causing lock errors that needed to be addressed by adding retry logic to the script. There were many other conditions added to the script as well to ensure all the training,

evaluation, and image generation worked correctly. The challenges for RF-DETR implementation were quite similar to those for YOLO. Apart from the GPU and CPU constraints when training the model on CARC, the challenge also occurred during the data preprocessing step when we dealt with the inconsistent dataset structures, ID collisions across datasets and category inconsistency problems, and it took us multiple attempts to work out the correct script.

Future steps for the YOLOv12L model including addressing class imbalance in the RF100VL dataset, using more trials for the Optuna optimization and ensuring all trials run, and coming up with more class-specific metrics. For RF-DETR, we could try to integrate a stronger visual backbone such as the most recent DINOv3 and also explore longer training schedules or curriculum style finetuning to see how that might affect the results.

References

[1] R. Sapkota, R. H. Cheppally, A. Sharda and M. Karkee, "RF-DETR Object Detection vs YOLOv12: A Study of Transformer-based and CNN-based Architectures for Single-Class and Multi-Class Greenfruit Detection in Complex Orchard Environments Under Label Ambiguity," *arXiv:2504.13099*, Apr. 17, 2025.

[2] P. Robicheaux, M. Popov, A. Madan, I. Robinson, J. Nelson, D. Ramanan, and N. Peri, *"Roboflow100-VL: A multi-domain object detection benchmark for vision-language models,"* RF100VL, 2025. [Online]. Available: https://RF100VL.org/.

[3] Y. Tian, Q. Ye and D. Doermann, "YOLOv12: Attention-Centric Real-Time Object Detectors," *arXiv:2502.12524*, Feb. 18, 2025.

[4] "Oriented Bounding Boxes (OBB) — Train," Ultralytics, online documentation. [Online]. Available: https://docs.ultralytics.com/tasks/obb/#train. (Accessed: Dec. 7, 2025).

[5] Ultralytics, *Ultralytics YOLO Hyperparameter Tuning Guide*, available: https://docs.ultralytics.com/guides/hyperparameter-tuning/ (Accessed Dec. 7, 2025).

[6] I. Robinson, P. Robicheaux, M. Popov, D. Ramanan, and N. Peri, "RF-DETR: Neural Architecture Search for Real-Time Detection Transformers," arXiv:2511.09554, Nov. 12, 2025. [Online].

[7] Roboflow, "How to Finetune RF-DETR on a Detection Dataset" (Colab notebook), GitHub/Google Colaboratory, [Online]. Available: https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/how-to-finetune-rf-detr-on-detection-dataset.ipynb. (Accessed: Dec. 6, 2025.)

[8] James Gallagher, Piotr Skalski, "How to Train RF-DETR on a Custom Dataset," Roboflow Blog, Mar. 20, 2025. [Online]. Available: https://blog.roboflow.com/train-rf-detr-on-a-custom-dataset/ (Accessed Dec. 6, 2025).

APPENDIX A

*NOTE: Table A1 and A3 corrected after presentation

Table A1

Zero-Shot Validation Results (Class-Agnostic)

(maxDets = 100, conf_thr = 0.054, NMS_IoU = 0.484)

Metric	IoU	Value
Precision	0.5	0.382
	0.75	0.239
Recall	0.5	0.202
	0.75	0.126
F1	0.5	0.265
	0.75	0.165
mAP	0.5	0.105
	0.75	0.049
	0.5:0.95	0.055

Table A2

Finetuned Validation Results

(maxDets = 100, conf_thr = 0.054, NMS_IoU = 0.484)

Precision	0.5	0.720
	0.75	0.525
Recall	0.5	0.893
	0.75	0.651
F1	0.5	0.797
	0.75	0.581
mAP	0.5	0.759
	0.75	0.543
	0.5:0.95	0.493

Fig. A1. Optimal hyperparameters for Stage 1 found utilizing Optuna optimization for 10 trials ran in parallel

Fig. A2. Optimal hyperparameters for Stage 2 found utilizing Optuna optimization for 10 trials ran in Parallel. Note that lr0, lrf, weight_decay, degrees, translate, scale, mosaic mixup were re-optimized in Stage 2 with smaller values to ensure stable full model training

Table A3

Zero-Shot Test Results (Class-Agnostic)

(maxDets = 100, conf_thr = 0.054, NMS_IoU = 0.484)

Metric	IoU	Value
Precision	0.5	0.359
	0.75	0.229
Recall	0.5	0.182
	0.75	0.116
F1	0.5	0.242
	0.75	0.154
mAP	0.5	0.088
	0.75	0.043
	0.5:0.95	0.047

Table A4
Finetuned Test Results
(maxDets = 100, conf_thr = 0.054, NMS_IoU = 0.484)

Precision	0.5	0.723
	0.75	0.535
Recall	0.5	0.891
	0.75	0.659
F1	0.5	0.798
	0.75	0.591
mAP	0.5	0.761
	0.75	0.543
	0.5:0.95	0.494

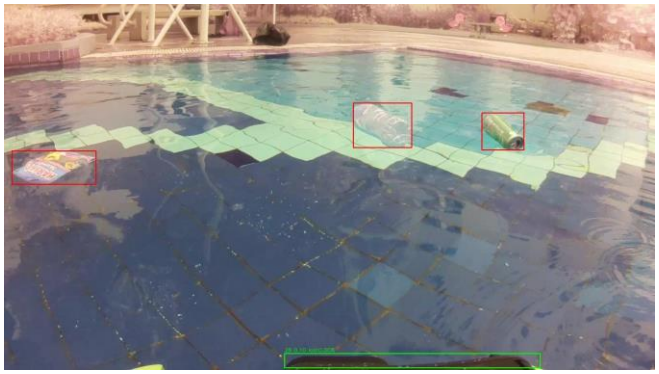

Fig. A3. Ground Truth (red) and Prediction (green) bounding boxes generated for non-finetuned evaluator. The prediction box is not around the right objects.

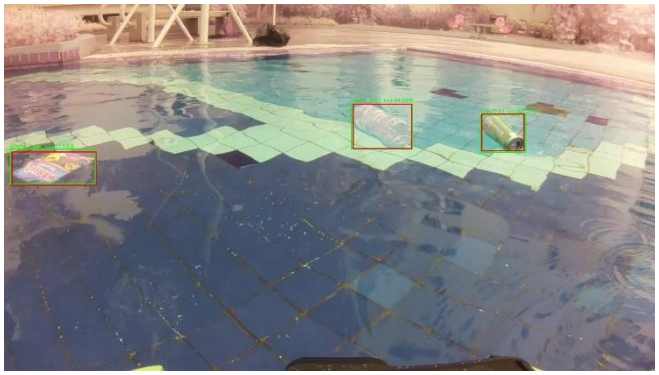

Fig. A4. Ground Truth (red) and Prediction (green) bounding boxes generated for finetuned evaluator. The prediction box is now around the right 3 objects, which are correctly labelled plastic, bottle and can from left to right and give IoU overlaps with the ground truth box of 87.06%, 94.88%, and 88.54% from left to right

APPENDIX B
Table B1
Non-Finetuned Results

Non-Finetuned result	
IoU_thr	0.5
Conf_thr	0.01
tp	73267
fp	4061795
fn	48246
precision	0.018
recall	0.60
f1	0.034

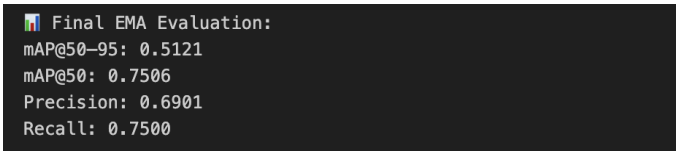

Fig B1. Final EMA summary statistics

Table B2
Finetuned validation results

	IoU	area	maxDets	
Average precision	0:50:0.95	all	100	0.512
	0.50	all	100	0.751
	0.75	all	100	0.559
	0.50:0.95	small	100	0.295
	0.50:0.95	medium	100	0.426
	0.50:0.95	large	100	0.574

Table B3
Finetuned test results

	IoU	area	maxDets	
Average precision	0:50:0.95	all	100	0.519
	0.50	all	100	0.759
	0.75	all	100	0.570
	0.50:0.95	small	100	0.322
	0.50:0.95	medium	100	0.468
	0.50:0.95	large	100	0.581

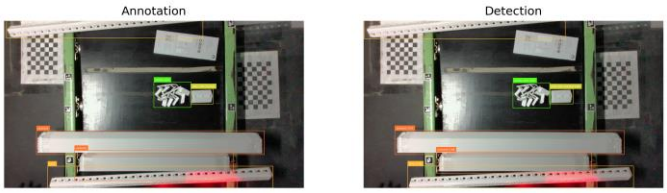

Fig B2. Image result after running inference on the RF100vl test set