Evaluating the Performance of RT-DETR - Real Time DEtection TRansformer for Indvidual Tree Detection in RGB images

Anderson da Rocha Lemos
UnB
Campus Darci Ribeiro - Asa Norte, Brasília - DF, 70910-90
andersonrl66@gmail.com

Abstract

The RT-DETR Real Time DEtection Transformer is a neural network specialized in detecting objects. Its paper has been published in july 2024 and presents results that beat the Yolo series in accuracy and speed. In this work we analyze the performance of RT-DETR first version according to a dataset of individual labeled trees images published in Zamboni et al. [7]. This article also investigated 21 methods of detection which serves as a initial basis of comparison of our results. We used a free Colab environment to train the network with a T4 GPU. This limited our hyperparameter options but allowed us to analyze the results in a restricted environment, similar to the conditions of the models trained in the base article. In this sense, we also trained for five variations of the dataset to perform cross-validation and have a more robust measure of model performance. Finally, we made some small variances on hyper-parameters to verify some possibilities of performance improvement. We were able to increase the number of epochs in the restricted environment up to 100 epochs.

Figure 1. Detect images using RT-DETR. Images source [7]

1. Introduction

Urban trees play an important role in cities. From environmental benefits, like air quality improvement, to social and economic benefits, like mental and physical health, they are essential for maintaining a good quality of life [5].

Zamboni *et al*. [7] investigated methods for detecting and quantifying individual trees in urban areas by comparing 21 different deep learning models of image detection, using anchor-based (one and two-stage) and anchor-free methods.

There are numerous challenges in individual urban tree detection like the diversity of species, differences in crown sizes, overlapping of crowns, the heterogeneity of urban scenarios and the difficulty in identifying small trees [6]. They utilized remote sensing for collecting high-resolution RGB orthoimages and manually annotated them. This dataset is available in https://github.com/pedrozamboni/individual_urban_tree_crown_detection. They applied the images and their annotations to 21 models and assessed the results using the average precision (AP). The true positive was obtained when the Intersection over Union (IoU) reached 0.50 (AP_{50}). The models were divided in two categories: anchor-based (AB one, two and multi-stage) and anchor-free (AF). The five best models were FSAF (AF), Double Heads (AB 2/multi stage), CARAFE (AB 2/multi stage), ATSS (AF), and FoveaBox (AF). The Double Heads achieved the best average (AP_{50}) (0.732).

Recently, end-to-end Transformers Base Detectors (DETRs) [2] are being implemented and achieving great results. In special, RT-DETR (Real Time DEtector TRansformer)[8] has achieved better results than the YOLO Series [4] in real time detection. Our goal is to assess the RT-DETR model and verify its performance in the individual urban tree dataset. We also restricted our work to run in Google Colab https://colab.research.google.com/ to investigate the viability of running this model in a free and limited environment.

1.1. RT-DETR

Real-time detectors aim to identify and localize objects in image sequences. It is important then to balance accuracy and speed in tasks like video surveillance [3] and autonomous driving [1]. Real time detection is typically done with convolutional networks, the most famous are the YOLO series [4]. These methods use a post-

processing technique NMS - Non-maximum suppression (NMS) to eliminate duplicate detection, slowing down inference speed.

While end-to-end Transformer-based detectors (DETRs) [2] eliminated components like NMS, they did not initially improve inference speed. They suffered from problems like slow training convergence, high computational cost, and hard-to-optimize queries. The RT-DETR [8] was created with the goal of beating YOLO in both accuracy and speed (see fig. 2), redesigning DETRs encoding scheme, "decoupling the intra-scale interaction and cross-scale fusion of features with different scales", and proposing a new query selection scheme, which reduces localization uncertainty.

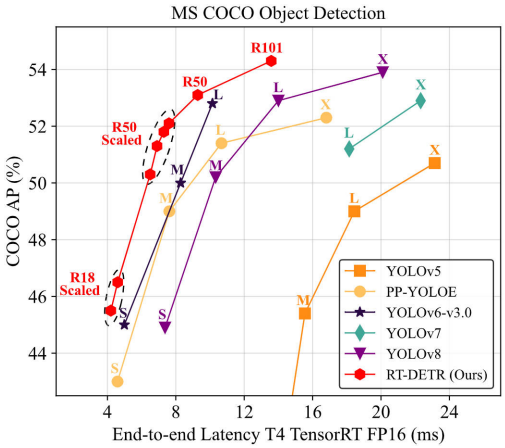

Figure 2. Performance of RT-DETR. Source [8]

1.2. Model Overview

RT-DETR architecture is composed of a CNN backbone, an hybrid encoder, and a transformer decoder, as shown in figure 3. The encoder receives features of the last three layers of the CNN and form a sequence of features through intra-scale and cross-scale feature fusion. The Attention-based Intra-scale Feature Interaction (AIFI) receives inputs only from the last layer of the CNN. The Cross-cale Feature Fusion (CCFF) uses convolution to fuse adjacent features into a new one. The input to the decoder is given by the uncertainty-minimal query selection who selects a fixed number of features from the cross-scale feature fusion, addressing both localization and classification. The decoder then uses the selection of features to predict boxes and categories.

1.3. Model Selection

There are several RT-DETR pre-trainned models available for training as shown in figure 4. For this work we chose the RT-DETR-R50 pre-trainned with COCO + Objects365. We used a free Colab environment with T4-GPU. Due to the limited time and storage we had in this context, we chose a

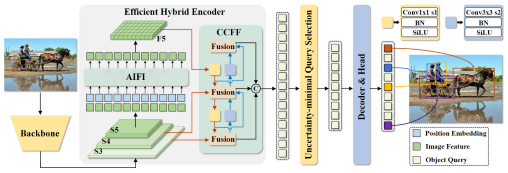

Figure 3. Model Overview. Source [8]

model that has a medium number of parameters and a good (AP_{50}) compared to the others : RT-DETR-R50.

Model	Input shape	Dataset	AP^{val}	AP^{val}_{50}	Params(M)	FLOPs(G)	T4 TensorRT FP16(FPS)
RT-DETR-R18	640	COCO	46.5	63.8	20	60	217
RT-DETR-R34	640	COCO	48.9	66.8	31	92	161
RT-DETR-R50-m	640	COCO	51.3	69.6	36	100	145
RT-DETR-R50	640	COCO	53.1	71.3	42	136	108
RT-DETR-R101	640	COCO	54.3	72.7	76	259	74
RT-DETR-HGNetv2-L	640	COCO	53.0	71.6	32	110	114
RT-DETR-HGNetv2-X	640	COCO	54.8	73.1	67	234	74
RT-DETR-R18	640	COCO + Objects365	**49.2**	**66.6**	20	60	**217**
RT-DETR-R50	640	COCO + Objects365	**55.3**	**73.4**	42	136	**108**
RT-DETR-R101	640	COCO + Objects365	**56.2**	**74.6**	76	259	**74**
RT-DETRv2-S	640	COCO	**48.1** (+1.6)	**65.1**	20	60	217
RT-DETRv2-M*	640	COCO	**49.9** (+1.0)	**67.5**	31	92	161
RT-DETRv2-M	640	COCO	**51.9** (+0.6)	**69.9**	36	100	145
RT-DETRv2-L	640	COCO	**53.4** (+0.3)	**71.6**	42	136	108
RT-DETRv2-X	640	COCO	54.3	**72.8** (+0.1)	76	259	74

Figure 4. Pre-trainned Models. Source https://github.com/lyuwenyu/RT-DETR

2. Material and Methods

Our work used the same dataset of images of individual labeled trees published in Zamboni et al. [7]. The dataset was divided into five different sets to perform cross-validation. Prior to implementing three experiments, as illustrated in Figure 5, the dataset was converted to the COCO format, the default format for RT-DETR. These experiments are described in the following sections.

2.1. Experiment 1

The goal of the first experiment was to run the model with similar hyper-parameters used by Zamboni *et al*. [7] to compare to the others 21 methods. For this experiment, we trained the model on one image set with an initial learning rate of 0.00125 and 24 epochs as shown in figure 6, which plots the loss and (AP_{50}) across the epochs. We also ran with an initial learning rate 0.0001 (figure 7) but the validation AP_{50} was far below the best methods and we decided not to continue with this hyper-parameters.

2.2. Experiment 2

The goal of the second experiment was to evaluate the model using the default hyper-parameters provided in

Figure 5. Experiments

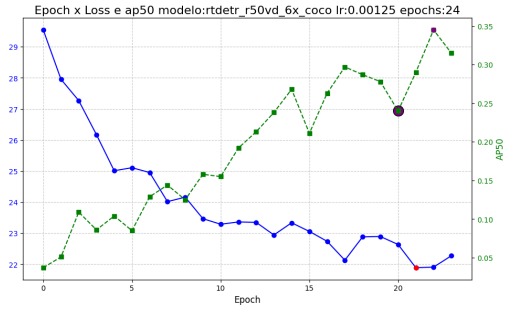

Figure 6. Experiment 1 - Train on dataset of individual trees with initial learning rate of 0.00125 and 24 epochs. Validation loss data was not available in the logs

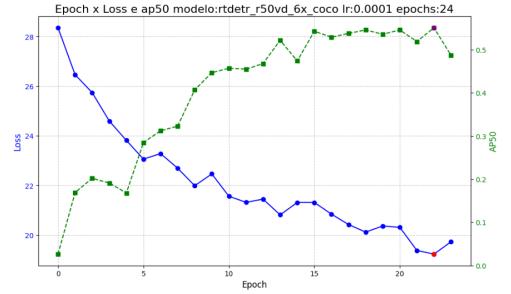

Figure 7. Experiment 1 - Train on dataset of individual trees with initial learning rate of 0.0001 and 24 epochs. Validation loss data was not available in the logs.

the official implementation https://github.com/lyuwenyu/RT-DETR. We used a learning rate of 0.0001 and ran the model for 72 epochs using five different combinations of image sets to perform cross-validation as shown in figures 8 (loss across epochs) and 9 ((AP$_{50}$) across epochs). We also performed a test to simulate early stopping using a "patience" parameter that tells how many iterations training can continue without improving (AP$_{50}$). Just for the sake of exemplification we plot in figure 9 points were training would stop if used patience = 2. No image set trained would stop if we use patience = 3.

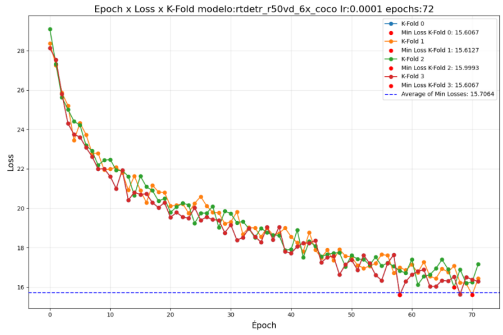

Figure 8. Experiment 2 - Train on 5 image sets of individual trees with initial learning rate of 0.0001 and 72 epochs.

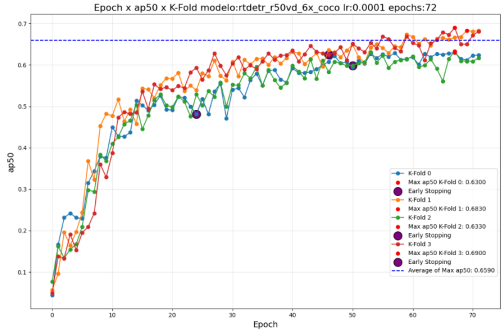

Figure 9. Experiment 2 - Train on 5 image sets of individual trees with initial learning rate of 0.0001 and 72 epochs.

2.3. Experiment 3

In the third experiment we tried to improve AP$_{50}$ performance by changing some hyperparameters. Unfortunately, due to the use of free Colab environment we were limited by time and storage. We were not able to assess larger models with bigger number of backbone layers. The free environment also disconnected with 125 and 150 epochs but could run 100 epochs for all five image sets, as shown in figures 10 and 11. This figure also shows the simulation of early stopping with patience = 3. Now, the training would not stop with patience = 4.

2.4. Results Assessment

We used the same metric used in Zamboni *et al*. to analyze our results and compare to the results published in [7]. The

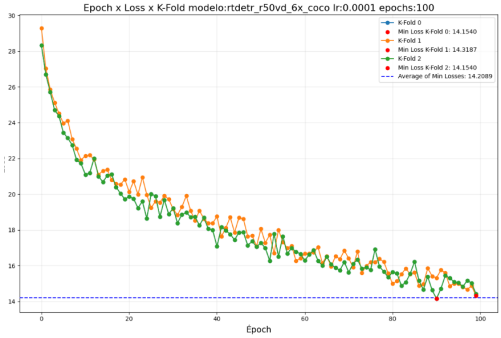

Figure 10. Experiment 3 - Train on 5 image sets of individual trees with initial learning rate of 0.0001 and 100 epochs.

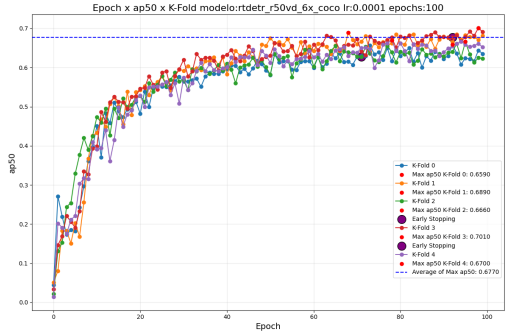

Figure 11. Experiment 3 - Train on 5 image sets of individual trees with initial learning rate of 0.0001 and 100 epochs.

AP_{50} is determined from the precision-recall curve. Precision and recall are based on IoU (Intersection over Union) which classifies a prediction as true positive if the intersection areas of the predicted bounding box and its ground truth is above 0.50. If it is below 0.50 it is considered false positive and if the ground-truth is not detected it is considered false negative.

As mentioned before, we used "Patience" to indicate weather we could improve our results if we raise the number of epochs, as validation loss data was not available in the logs. Patience tells how many iterations training can continue without improving (AP_{50}), simulating early stopping. If the training does not reach some value of patience in any point it can indicate that we can continue raising the number of epochs.

3. Results

Now we present the results of our experiments.

3.1. Experiment 1

Experiment 1 was interrupted after evaluating the best AP_{50} value for the validation dataset. It was 0.35 for initial learning rate = 0.00125 and 24 epochs, and 0.49 for initial learning rate = 0.0001 and 24 epochs, far below from the best methods.

3.2. Experiment 2

The test results for Experiment 2 (lr = 0.0001, epochs = 72) are shown in table 1 and in figure 12. Figure 13 shows an example of an inferred image.

Image set	AP_{50}
0	0.65800
1	0.64400
2	0.67200
3	0.64300
4	0.63400
Average	**0.64400**

Table 1. RT-DETR Test Results for five image sets, lr = 0.0001, epochs = 72

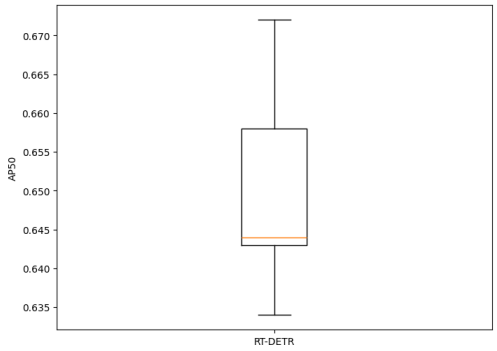

Figure 12. Box-plot for RT-DETR results, lr = 0.0001, epochs = 72

Figure 13. Example of inferred image.

In this experiment we ran with the default hyper-parameters found in https://github.com/lyuwenyu/RT-DETR. The results improved a lot but were still below the best methods tested in [7]. The average AP_{50} was 0.6440, below all anchor-free methods (table 2). It was just above Weight Standardization (table 3) and YoloV3 (table 4).

Model	AP_{50}
ATSS	0.69200
VarifocalNet (1)	0.66400
VarifocalNet (2)	0.68300
FSAF	0.70100
FoveaBox	0.69200

Table 2. Anchor-free test results. Source: [7]

Model	AP_{50}
Faster R-CNN	0.66000
DetecoRS	0.65100
Weight Standardization	0.63100
Deformable ConvNets v2	0.65700
CARAFE	0.69700
Dynamic R-CNN	0.65500
Double Heads	0.69900
Mixed precision training	0.67900
Empirical Attention	0.69000

Table 3. Two and multi-stage performance. Source: [7]

Model	AP_{50}
SABL	0.66100
Generalized Focal Loss	0.67700
Probabilistic Anchor Assignment	0.67700
RetinaNet	0.65000
NAS-FPN	0.65800
YoloV3	0.59100
Gradient Harmonized Single-stage Detector	0.69100

Table 4. One-stage performance. Source: [7]

We then investigated whether increasing the number of epochs could improve performance, as suggested by the difference between experiment 1 and 2. To assess this, we used the patience test. No image set would stop with patience = 3, suggesting that we can raise the number of epochs, as AP_{50} was still improving.

3.3. Experiment 3

The test results for Experiment 3 (lr = 0.0001, epochs = 100) are shown in table 5 and in figure 14. The average AP_{50} was 0.6460, almost the same of experiment 2. If we take away the result from image set 2, which is an outlier, the average goes to 0.6430.

Image set	AP_{50}
0	0.63700
1	0.64700
2	0.68200
3	0.64100
4	0.64600
Average	**0.64600**

Table 5. RT-DETR Test Results for five image sets, lr = 0.0001, epochs = 100

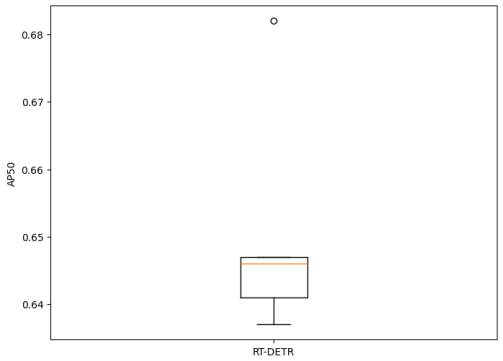

Figure 14. Box-plot for RT-DETR results, lr = 0.0001, epochs = 100

Now, no image set would stop with patience = 4, a higher value than experiment 2 but still suggesting that we can raise the number of epochs. However, the results from test sets didn't show any improvement when we raised the number of epochs to 100.

4. Discussion

The results are summarized in table 6. We discuss here the limitations of our choices and the variations on initial learning rate, number of epochs and minimum value of patience that we experimented.

Exp.	Initial LR	Epochs	AP_{50}	Min. Patience
1	0,00125	24	-	-
2	0,00010	72	0,64400	3
3	0,00010	100	0,64600	4

Table 6. RT-DETR Test Results summary

4.1. Limitations

In this work we used a free colab environment to see the viability and limitations of training state of the art models in a restricted environment. We could not run all the models available, especially those with more backbone layers. Then number of epochs were also limited by time and space, as the environment disconnect after a fix period of time and the storage eventually become full with the recording of checkpoints. However, we could run a representative model with the default hyper-parameters and could also raise the number of epochs and analyze the results. We were also able to run cross-validation for five different image sets to get a more stable result.

4.2. Initial Learning Rate

We first tried to use the same initial learning rate used by Zamboni *et al*. [7] (0.00125) with 24 epochs but the validation AP_{50} was far below the best methods (0.35). When we changed it to 0.0001, AP_{50} raised to 0.49, indicating that it is more appropriate to RT-DETR but still below the best methods.

4.3. Number of epochs

We then raised the number of epochs to the default number found in https://github.com/lyuwenyu/RT-DETR (72) and the results improved a lot, with AP_{50} achieving an average of 0.6440 for five image sets. However, it was below all anchor-free methods and it was just above Weight Standardization and YoloV3. This raise of AP_{50} after 72 epochs indicates that RT-DETR convergence is slower than the others methods analyzed. We used patience to simulate early stopping and found that for every image set AP_{50} did not decrease for more than 2 epochs in sequence, indicating that we could increase the number of epochs. We could raise the number of epochs up to 100 and achieved an average of 0.6460. If we take away the result from image set 2, which is an outlier, the average goes to 0.6430, showing no improvement with 100 epochs.

5. Conclusion

In this work we investigated the performance of the RT-DETR model in the individual urban tree dataset in a free Google Colab environment. We had to choose a representative model among those available due to the limited amount of resources, time and storage. We could train an image set of 220 images up to 100 epochs in one session, without disconnection, but had to adopt an strategy to remove checkpoints to not overload storage. We found that running in a free environment can be useful in a proof-of-concept of the model before running bigger versions, but it is not viable for achieving robust results. The AP_{50} results achieved from this proof-of-concept were not impressive. Compared to the others results found in Zamboni *et al*. [7], RT-DETR could only beat some anchor-free models. The best results were found when we ran the model with the suggested number of epochs for the model chosen and did not improved when we raised to 100 epochs.

References

[1] Daniel Bogdoll, Maximilian Nitsche, and J Marius Zöllner. Anomaly detection in autonomous driving: A survey. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4488–4499, 2022. 1

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 1, 2

[3] Rashmika Nawaratne, Damminda Alahakoon, Daswin De Silva, and Xinghuo Yu. Spatiotemporal anomaly detection using deep learning for real-time video surveillance. *IEEE Transactions on Industrial Informatics*, 16(1):393–402, 2019. 1

[4] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016. 1

[5] S Roy, J Byrne, and C Pickering. A systematic quantitative review of urban tree benefits, costs, and assessment methods across cities in different climatic zones. *Urban Forestry & Urban Greening*, 11(3):351–363, 2012. 1

[6] Felipe Wagner, Marcus Vinicius Ferreira, Andrea Sanchez, Mahesh Hirye, Mateus Zortea, Elizabeth Glorr, Odorico P Carlos Souza Filho, Yosio Edemir Shimabukuro, and Luiz Eduardo OC Aragão. Individual tree crown delineation in a highly diverse tropical forest using very high resolution satellite images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 145:362–377, 2018. 1

[7] Pedro Zamboni, José Marcato Junior, Jonathan de Andrade Silva, Gabriela Takahashi Miyoshi, Edson Takashi Matsubara, Keiller Nogueira, and Wesley Nunes Gonçalves. Benchmarking anchor-based and anchor-free state-of-the-art deep learning methods for individual tree detection in rgb high-resolution images. *Remote Sensing*, 13(13):2482, 2021. 1, 2, 3, 5, 6

[8] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. Detrs beat yolos on real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 1, 2

Source code

The source code for this work is available at https://github.com/andersonrl66/rt-detr