SkyGrove

1st Eric Tom Mathews
Stirling, Scotland
Student ID: 3058481
etm00024@students.stir.ac.uk

2nd Mayank Harbola
Edinburgh, Scotland
Student ID: 2730828
mah00177@students.stir.ac.uk

I. Introduction

Tree detection in images is an important task in various applications, such as forest management, environmental monitoring, and urban planning. Object detection is an extension of the Image classification task. We not only assign a label to the image but also localize the object in the image by drawing a bounding box around it.

Object detection models, such as Faster R-CNN [1] and YOLO, have shown great success. Faster R-CNN is a region-based object detection model that uses a two-stage approach to detect objects, while YOLO is a single-stage object detection model that directly predicts object bounding boxes and class probabilities from the image. In this research, we compare the performance of Faster R-CNN with different backbones and YOLO for tree detection. We evaluate the models on a dataset of labeled images containing trees and also through the test dataset from another city. Our experiments' results provide insights into the strengths and weaknesses of our models and give an understanding of the standard procedures needed to follow in object detection tasks.

II. Proposed Solution with Justifications

A. Data Acquisition

The images were taken manually by both the authors from Stirling and Edinburgh. 53 images were collected per city which was then augmented into 212 images for each city. We used Computer Vision Annotation Tool (CVAT) [8] for annotating trees in our dataset. We did a random split such that 50% is allocated to training and the rest 50% to validation/testing of our models.

B. Data Augmentation with Dataloaders

We resized the raw images to a maximum resolution of 1024 pixels. The resulting images from Stirling and Edinburgh were 1024*768 and 1024*680 pixels in dimension.

Instead of using normal random flips and filters available in pytorch, we used 'imgaug' Python package to augment our images. We transformed our images with climate filters such as fog and snow along with Gaussian noise, brightness, hue, saturation adjustments, and flipping to make the data more robust portrayed in Fig1.

We prepared annotations in YOLO which were then converted into COCO format with the help of a YOLO to COCO format converter [9] to build and evaluate our YOLO and Faster RCNN models. To speed up the data retrieval for training our models we used a batch size of 4 and then iterate over all the batches.

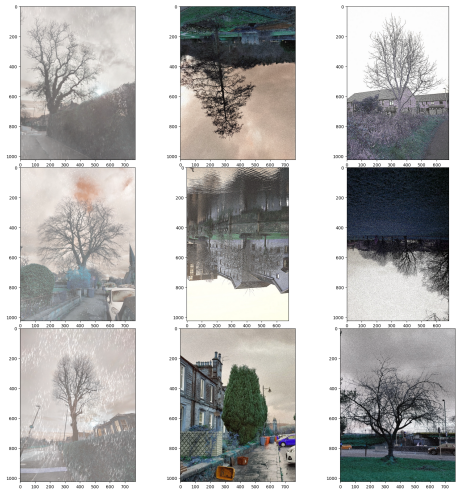


Fig. 1: Augmented Samples

C. Model training and Optimization

The object detection methodology involves the identification of instances and assigning class labels and also giving spatial location and drawing a bounding box to give a measure of its extent. [2]

We used a region proposal-based technique called Faster-RCNN for building our model for object detection. We also compared our models with the state-of-the-art YOLO v8. object detection algorithm.

The architecture of Faster-RCNN consists of a backbone network, a region proposal network(RPN), and a ROI pooling layer. The image is passed on to the backbone network to get a tensor feature map. This backbone network can be any dense convolutional network such as VGG-Net, AlexNet, MobileNet, or ResNet50. The RPN finds the region which contains more likelihood of an object by generating the objectness score. The ROI pooling layer gives the constant size feature vector for each of the generated region proposals. The last regression and classification layer is used to classify it as foreground or background and to correct the position of the bounding box. In contrast to Faster-RCNN, in YOLO a single convolutional network predicts both bounding boxes and class probabilities for the boxes.

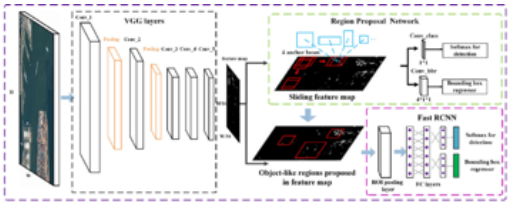


Fig. 2: Faster RCNN architecture [3]

We build two Faster RCNN models with pre-trained MobileNet, and ResNet50 as the backbone and passed them to the evaluation stage. We also built one model based on YOLO v8 for comparison purposes. Considering we had a finite amount of data it was important to reduce the training error by updating the weights of CNN and minimizing the overall loss for every epoch using optimization algorithms. Since computational time was an important factor we used Stochastic Gradient Descent Algorithm [4] for optimization. The hyperparameters selected for the optimization process of our two Faster-RCNN models are given in the table below.

Epochs	Learning Rate	Batch Size
10	0.01	4

TABLE I: Hyperparameters

D. Model Evaluation

It is important to find metrics that will compare the performance of the object detection model. The IoU value helps us to know the tightness of the bounding box of our detection while mAP will tell us the correctness of our predictions.

IoU is defined as the ratio of the area of intersection and area of the union of ground truth and the predicted bounding box. Usually, we define IoU > 0.5 to consider the predicted bounding box to be true positive. False positives are considered when IoU is less than 0.5 or there is also a duplicate bounding box. We also consider the false negative cases when there is no detection at all or misclassification of the bounding box. Mean average precision(mAP) [5] compares ground truth object annotations of the image data with the object detections from the model and computes an accuracy score. To find average precision(AP) first, we generate a precision-recall curve for each object class using the IoU. The IoU are thresholds from 0.5 to 0.95 and the mean score for all classes gives the value of mAP. We refer to the traditional mAP50 value when IoU threshold is taken as 0.5. It is more likely that the true positive detections are the ones with high confidence, therefore we set the value of confidence to 0.8. Setting this confidence value too high would have resulted in an overly optimistic model. [6]

III. Results

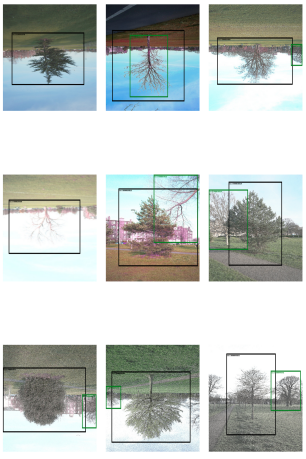
Since we had training data from each city i.e. Stirling and Edinburgh, we built a total of six models consisting of Faster R-CNN with pre-trained MobileNet and ResNet50 as the backbone and YOLO v8. We evaluated the model with

(a) Faster RCNN ResNet50 Stirling model on Edinburgh Dataset.

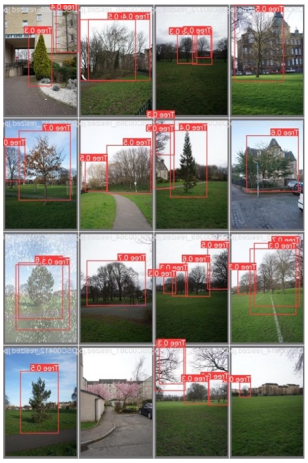
(b) Faster RCNN ResNet50 Stirling model on Stirling Dataset.

(c) YOLO Stirling model on Stirling Dataset.

(d) Faster RCNN ResNet50 Edinburgh model on Stirling Dataset.



(e) Faster RCNN ResNet50 Edinburgh model on Edinburgh Dataset.



(f) YOLO Edinburgh model on Edinburgh Dataset.

Fig. 3: Comparison of Faster RCNN ResNet50 and YOLO models on datasets.

our validation data from the same city and then calculated the evaluation metrics which are summarized in the table below.

We selected the Faster R-CNN with ResNet50 as our final model from both cities since it showed good evaluation metrics with limited epochs even though there is a scope for improvement in all the models. We further evaluated this model with the test dataset of the other city i.e. the model built on the Stirling dataset was tested with the test dataset from Edinburgh and vice versa. We summarised the results in the tables below.

TABLE II: Model Results: Stirling

	Faster R-CNN with MobileNet	***Faster R-CNN with ResNet50***	***YOLO v8***
IoU	0.47	0.49	_
mAP50	0.75	0.88	0.86

TABLE III: Model Results: Edinburgh

	Faster R-CNN with MobileNet	***Faster R-CNN with ResNet50***	***YOLO v8***
IoU	0.33	0.37	_
mAP50	0.4	0.7	0.83

TABLE IV: Experiment 3: Faster RCNN with ResNet 50 Stirling model

	Stirling Dataset	***Edinburgh Dataset***
IoU	0.49	0.34
mAP50	0.88	0.37

TABLE V: Experiment 3: Faster RCNN with ResNet 50 Edinburgh model

	Edinburgh Dataset	***Stirling Dataset***
IoU	0.37	0.43
mAP50	0.7	0.68

IV. Discussion

From the performed experiments we were able to demonstrate the use of object detection algorithms such as Faster-RCNN and YOLO for object detection. Both the Faster R-CNN(ResNet50) models were able to detect trees in the images. One key finding from our results (Tables IV and V) was that both the models suffered in accuracy(mAP50) in detection in the Edinburgh dataset. Since the majority of our images in the Edinburgh dataset consisted of multiple overlapping trees or small trees, our models struggled to detect them as individual trees or there was no detection at all. This was a bit improved by the ROI pooling layer of the Faster R-CNN, however, due to annotations complexity, the algorithm still struggled to detect trees on different scales or aspect ratios.

The performance of the model is highly correlated with the 'perfect' annotations of the custom dataset. The latency of the models can be considered in future works when deploying these models in real-life scenarios. Since the model relies on the data, other variations arising due to lighting conditions, perspective, and tree conditions in different seasons should also be considered. If the purpose is tree counting or avoiding false positives, to calculate average precision, we need to consider results with low confidence as well. With low confidence, the mAP/AP values are immune to false positives if we have already covered ground truth with high confidence. [7]

V. Conclusion

We implemented state-of-the-art Faster-CNN with different backbones and YOLO v8 algorithms for building our Object detection model for the identification and localization of trees in the image. The experiment demonstrated that the models built on these algorithms were good enough for tree detection. We also utilized different metrics to compute the performance of our models. The industry standard for IoU is, IoU>=0.95. Although our models were nowhere close to it, however, there is scope for improvement in our models. Due to the limited time availability of the project, the models can be improved by increasing the size of the dataset, more exploration of model hyperparameters search space, and improving the quality of annotations with adequate ground truth examples and with accurate annotated bounding boxes.

References

[1] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," 2015.

[2] Shet Reshma Prakash, Paras Nath Singh, "Object detection through region proposal based techniques," Materials Today: Proceedings, vol. 46, 2021.

[3] Zhipeng Deng, Hao Sun, Shilin Zhou, Juanping Zhao, Lin Lei, "Multi-scale object detection in remote sensing imagery with convolutional neural networks," ISPRS Journal of Photogrammetry and Remote Sensing, 2018.

[4] 12.4. Stochastic Gradient Descent," [Online]. Available: https://d2l.ai/chapter_optimization/sgd.html.

[5] R. J. Tan, "Breaking Down Mean Average Precision (mAP)," [Online]. Available: https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52.

[6] E. Hofesmann, "IoU a better detection evaluation metric," [Online]. Available: https://towardsdatascience.com/iou-a-better-detection-evaluation-metric-45a511185be1.

[7] "https://www.objectivity.co.uk/blog/comparing-object-detection-models/," [Online]. Available: Comparing object detection models.

[8] "CVAT," www.cvat.ai. https://www.cvat.ai/

[9] T. Kim, "Yolo-to-COCO-format-converter," GitHub, Apr. 11, 2023. https://github.com/Taeyoung96/Yolo-to-COCO-format-converter