[Midterm Project Report]
Rotated Object Detection

Ashu Kumar, Neeraj Menon, Pooja Yadav, Pratik Sanghavi*
Group Name: Rotated Squad

Introduction

Overview

Rotated object detection, also known as arbitrary-oriented object detection, aims to identify and locate objects in images with arbitrary orientation. In this task, the oriented directions of objects differ significantly across different images, with multiple orientations possible within a single image. This inherent complexity poses a challenge for standard backbone network models to extract high-quality features of these arbitrarily oriented objects. In addition, complex backgrounds, dense distributions, and heavy occlusions in images make accurate identification even more difficult.

Problem Statement

In this project, we intend to address the aforementioned challenges by proposing a generalizable approach for accurate and efficient rotated object detection.

Significance

In contrast to generic object detection, where object instances are assumed to be aligned with the image axes, objects within the natural scenes often exhibit arbitrary orientations. Despite the significant differences between the images with generic items and those with oriented objects, the design of conventional visual backbones has largely overlooked the inherent characteristics. As a result, the architecture of these models may be sub-optimal in the rotated object detection task.

Moreover, the need for rotated object detection is commonly observed in several fields, including remote sensing detection, text detection ‘in the wild’, face detection, retail scene detection, medical physics, industrial inspection, etc.

Hence, addressing the challenge of detecting rotated objects is crucial, as it will strictly be superior to axis-aligned detection as there is less background within the box and the orientation can potentially convey object pose information.

Objectives & Goals

➔ Develop an approach that results in an improved Mean Average Precision (mAP) value

➔ Experiment with different backbones using the Oriented R-CNN [4] algorithm

➔ Evaluate the generalizability of the proposed approach on different datasets

*in alphabetical order

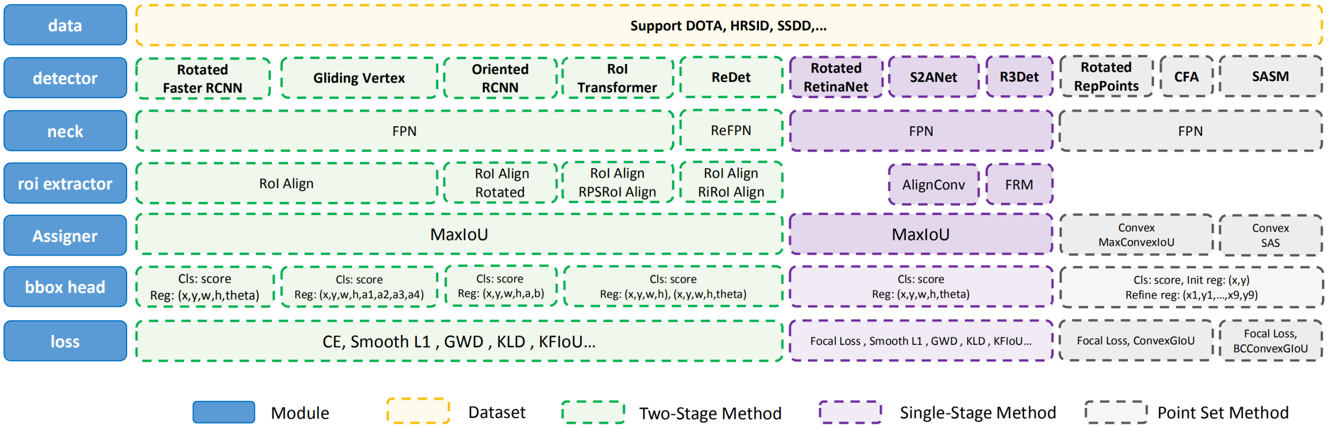


Figure 1: Module design of MMRotate

Progress Made So Far

Dataset Identification

After reviewing various options, we chose the DOTA-v1.0 dataset [1]. This dataset was selected based on its relevance to rotated object detection and the quality of data it provides. DOTA (Table 1) is a large-scale dataset for object detection in aerial images. The dataset is annotated with 15 categories and bounding box coordinates (x_i,y_i), i=1,2,3,4.

Model Identification

In line with our objectives, we identified a suitable model to serve as the backbone of our experiments. The model chosen for this purpose was the standard ResNet architecture. We have set our baseline algorithm as Oriented R-CNN with ResNet as the backbone. This combination brings the advanced rotated object detection capabilities of Oriented R-CNN, alongside the robust feature extraction provided by ResNet.

Environment Setup

We opted to utilize Google Colab, given its accessibility and the computational resources it offers. However, to ensure compatibility with our chosen dataset and model, it was necessary to make specific adjustments to the environment. This involved downgrading the version of CUDA and PyTorch on Colab to a compatible version that meets our requirements. We also resolved several dependency issues to ensure that the code was running. Additionally, we purchased Google Colab Pro due to the computing and memory requirements of the model being trained.

Baseline

With the prepared dataset, we began baseline training, utilizing the updated model framework of Oriented R-CNN with ResNet as the backbone. This stage is pivotal for setting a benchmark for our project, allowing us to evaluate the performance of other backbones with Oriented R-CNN. It also serves as a foundation for identifying potential improvements and making necessary adjustments to our approach. We achieved an mAP of 0.43 on the DOTA-v1.0 dataset on running the baseline model for 11 epochs.

Methodology

Data Preprocessing

To be able to compare our experiments fairly with the reference baseline, we use the same train-validation-test split ($\frac{1}{2}:\frac{1}{6}:\frac{1}{3}$) on DOTA 1.0 dataset. We augmented the sparse categories to obtain a more balanced dataset. Finally, we divided the high-resolution images into smaller, overlapping patches of 1024 × 1024 in order to comply with the expected input sizes for the reference backbone in the Oriented R-CNN paper.

Sr.No	Class	Ground Truths
1	Plane	6878
2	Baseball Diamond	485
3	Bridge	2479
4	Ground Track Field	237
5	Small Vehicle	35582
6	Large Vehicle	26963
7	Ship	43719
8	Tennis Court	3094
9	Basketball Court	417
10	Storage Tank	1012
11	Soccer Ball Field	182
12	Roundabout	333
13	Harbor	10517
14	Swimming Pool	2216
15	Helicopter	109

Table 1: DOTA-v1.0 training data with number of instances per class

Approach

Our objective is to improve the mAP value over the DOTA dataset by modifying the existing state-of-the-art models. We are using MMRotate [5], a popular framework based on PyTorch for a unified implementation of rotated object detection. It consists of various modules (Figure 1) for data loading and pre-processing, model definitions, evaluation tools, and more.

We envision modifying the backbone of Oriented R-CNN to improve the performance on the mAP metric while slightly compromising on the inference speed. To establish the baseline, we ran experiments using the ResNet50 backbone, the results of which are detailed in the Initial Results section.

Evaluation Metrics

Mean Average Precision (mAP) is a standard metric used to measure the performance of object detection models. It is equal to the average of the Average Precision metric across all classes in a model. It is measured between 0 and 1.

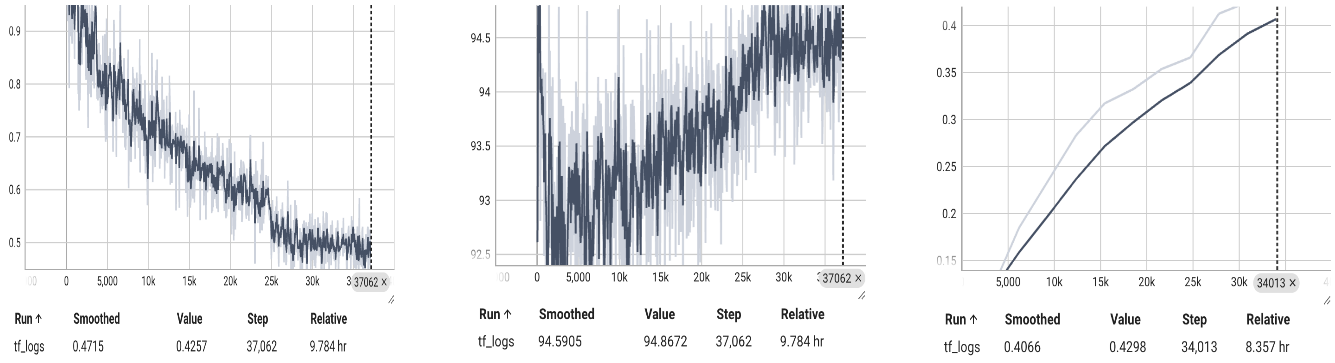
mAP encapsulates the tradeoff between precision and recall and maximizes the effect of both metrics. The metric is particularly useful for assessing performance in scenarios where detection accuracy across multiple categories is crucial.

The primary benchmark for assessing the model's performance will be the mAP value obtained on the validation set. Additionally, we will evaluate the memory usage and inference speed to understand the compromises made to enhance mAP. This multifaceted approach allows for a balanced evaluation of the model's efficiency, accuracy, and operational feasibility.

Initial Results

We used the ResNet50 backbone described in the Oriented R-CNN paper and used the same data preparation and hyperparameters. We used stochastic gradient descent with an initial learning rate of 0.005 and momentum of 0.9. We recorded the results of the best training run after 11 epochs, which are summarized in Table 2.

As shown in Table 1, the dataset is extremely imbalanced. We need to resample the categories to get an approximately equitable distribution. Next, we need to achieve the baseline mAP value of 0.75 with the configuration described. We suspect this difference may have been caused by a difference in the size of the dataset being used as well as the augmentations being done.



Run ↑	Smoothed	Value	Step	Relative
tf_logs	0.4715	0.4257	37,062	9.784 hr

(a) train/loss

Run ↑	Smoothed	Value	Step	Relative
tf_logs	94.5905	94.8672	37,062	9.784 hr

(b) train/acc

Run ↑	Smoothed	Value	Step	Relative
tf_logs	0.4066	0.4298	34,013	8.357 hr

(c) validation/map

Figure 2: Training loss, Accuracy and Validation mAP

We observe an increase in the validation mAP value as the training progresses. Though extending the training for additional epochs could enhance the validation mAP, we aim to match and contrast with the baseline established by the Oriented R-CNN paper. Continuing the training for too many epochs could result in overfitting, a situation we intend to prevent under any circumstances.

Challenges

We aimed to evaluate three algorithms: Adaptive Rotated Convolution (ARC) [3], Oriented RepPoints [2], and Oriented R-CNNs [4]. We encountered significant challenges with ARC and Oriented RepPoints due to unmitigable dependency conflicts and computational constraints, respectively. The setup for Oriented R-CNNs was also fraught with difficulties, including compatibility issues with CUDA and PyTorch, PyTorch's mmcv library versions. To address these challenges for future research, we will release a comprehensive setup guide and a Docker file. Additionally, our efforts to prepare and train on a large dataset (approximately 70GB uncompressed) were hindered by exceedingly slow processes and frequent session timeouts on Colab. To mitigate this, we invested in additional compute units and plan to shift to Vertex AI notebooks for subsequent project phases. The dataset's size posed further complications, as we struggled to manage it on limited storage capacities, even after employing data splitting and soft-linking strategies. This limitation notably impacted data read/write speeds, further exacerbating delays in data preparation and training.

Following the modifications made to the folder structure, we were required to update various configuration files. The dataset's imbalance presented considerable challenges, as evidenced by an initial mean Average Precision (mAP) of approximately 0.33. To mitigate these issues, we explored oversampling techniques; however, the disparity in dataset size continues to influence our training outcomes adversely. Moving forward, upon transitioning to Vertex AI and utilizing Google Cloud Storage (GCS) for data management, our primary focus will be to address and resolve the aforementioned challenges.

Planned Project Activities

In the final stages of our project, we aim to refine our approach and validate our findings through a series of planned activities focused on enhancing robustness and accuracy. We have the following next steps planned for the project:

- **Improve validation accuracy:** As mentioned above, our current implementation of the MMRotate rotated object detection is resulting in an mAP of about 0.43 even after multiple changes and modifications. We plan on first figuring out the root cause of this issue and ensuring its accuracy matches the ones reported in papers for the DOTA dataset. Next, we intend to work on improving the detection accuracy.

- **Dataset preparation:** Although we augmented the dataset to account for sparse categories, we did not obtain substantial gains. We, therefore, plan on balancing the training set to obtain more accurate model predictions.

- **Model Backbone:** We wish to experiment with different model backbones in MMRotate to identify the most effective architecture for our specific dataset and the DOTA dataset. The existing set of backbones already used

Class	**Detections**	**Recall**	**Avg. Precision**
Plane	26703	0.902	0.833
Baseball-diamond	11471	0.466	0.167
Bridge	29607	0.397	0.255
Ground Track Field	415	0.025	0.045
Small Vehicle	108994	0.821	0.715
Large Vehicle	66115	0.907	0.818
Ship	76388	0.824	0.778
Tennis Court	12104	0.887	0.813
Basketball Court	2294	0.175	0.017
Storage Tank	13354	0.668	0.432
Soccer Ball Field	39	0.011	0.030
Roundabout	2304	0.171	0.073
Harbor	35921	0.604	0.463
Swimming Pool	8662	0.925	0.793
Helicopter	288	0.385	0.213
mAP			**0.430**

Table 2: Validation Metrics after 11 epochs

in MMRotate are unable to detect objects with an mAP of more than 0.76 on the DOTA dataset, and we think we can work on a model that can give us better results than the current ones.

- **Different Datasets:** If time permits, we plan on finding and training our refined MMRotate rotated object detection model on a dataset different from the one used in the original paper i.e. DOTA.

Timetable

Timeline	**Milestone**	**Key Tasks**
Mar 28 - Apr 15	Project Phase 2	- Continue experiments - Improve validation accuracy - Draft final presentation - Improving Dataset imbalance - Model Backbone - Different Datasets
Apr 16 - May 3	Project Webpage	- Design webpage layout. - Include overview, methodologies, results, and conclusions. - Embed code snippets, demos, and source code links.

Member Contributions

Starting from the initial setup, configuring the environment, experimenting with models and configurations, monitoring the executions, and through all tasks in all stages of the project, we, the undersigned, state and affirm that all members of the group contributed fairly and equally to this assignment:

Ashu Kumar, Neeraj Menon, Pooja Yadav, Pratik Sanghavi
Signed on March 26, 2024

References

[1] Jian Ding et al. “Object Detection in Aerial Images: A Large-Scale Benchmark and Challenges”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), pp. 1–1. DOI: `10.1109/TPAMI.2021.3117983`.

[2] Wentong Li et al. *Oriented RepPoints for Aerial Object Detection.* 2022. arXiv: `2105.11111 [cs.CV]`.

[3] Yifan Pu et al. "Adaptive Rotated Convolution for Rotated Object Detection". In: *arXiv preprint arXiv:2303.07820* (2023).

[4] Xingxing Xie et al. "Oriented R-CNN for Object Detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 3520–3529.

[5] Yue Zhou et al. "MMRotate: A Rotated Object Detection Benchmark using PyTorch". In: *Proceedings of the 30th ACM International Conference on Multimedia*. 2022.