AlphaRotate: A Rotation Detection Benchmark using TensorFlow

Xue Yang¹  
Yue Zhou²  
Junchi Yan¹,*

¹Department of Computer Science and Engineering, MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai, China, 200240  
²Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China, 200240

Abstract

AlphaRotate is an open-source TensorFlow benchmark for performing scalable rotation detection on various datasets. It currently provides more than 18 popular rotation detection models under a single, well-documented API designed for use by both practitioners and researchers. AlphaRotate regards high performance, robustness, sustainability and scalability as the core concept of design, and all models are covered by unit testing, continuous integration, code coverage, maintainability checks, and visual monitoring and analysis. AlphaRotate can be installed from PyPI and is released under the Apache-2.0 License. Source code is available at https://github.com/yangxue0827/RotationDetection.

Keywords: Rotation Detection, Convolutional Neural Network, TensorFlow,

1. Introduction

Despite the rich literature (Girshick, 2015; Ren et al., 2015; Dai et al., 2016; Lin et al., 2017a, b) in visual object detection in computer vision, existing models are mostly agnostic to the object orientation, which only output a horizontal bounding box. This can be restricted in real-world settings whereby either the rotation information itself can be critical e.g. for aerial observation, or the rotated detecting bounding box can better align the object for more accurate recognition, especially for small and densely arranged objects.

For the above reasons, recently rotation detectors emerge with the development in terms of both backbone and loss design. The applications are arranged across aerial images (Yang et al., 2018a, b, 2019, 2021b), scene text (Zhou et al., 2017; Jiang et al., 2017; Liu et al., 2018; Ma et al., 2018; Liao et al., 2018), faces (Shi et al., 2018), 3D objects (Zheng et al., 2020), and retail scenes (Chen et al., 2020; Pan et al., 2020) etc.

However, there lacks an open-source benchmark integrating recent advance rotation detection models for evaluation and use. The most popular object detection benchmarks, e.g. MMDetection (Chen et al., 2019a), Detectron2 (Wu et al., 2019), SimpleDet (Chen et al., 2019b), are all focused on horizontal detection. AerialDetection¹ is an earlier rotation detection benchmark based on MMDetection. However, it only provides some baselines and few methods, and it lacks maintenance and integration of new methods. Moreover, all these

¹. Junchi Yan is the correspondence author.

¹. https://github.com/dingjiansw101/AerialDetection
To fill this gap, we propose and implement AlphaRotate – a TensorFlow based framework which consists of state-of-the-art detection techniques and models (keep on extension) – mostly from our previous works in recent literature (Yang et al., 2018a, b; 2019; Yang and Yan, 2020; Qian et al., 2021; Yang et al., 2021a, b, c, d). It is user-friendly to both industry and academic community, with the following features, as shown in Figure 1 and Table 1.

1) AlphaRotate is one of the first rotation detection benchmarks (based on TensorFlow) and it supports training and testing on various datasets including aerial images, scene text, and face. It supports state-of-the-art rotation detectors (including hybrid methods), and provides comprehensive evaluation on DOTA (Xia et al., 2018) dataset.

2) A clean and modular implementation, which eases future integration of new methods and facilitates assembling different modules, such as Backbone, Neck, Loss, etc.

3) AlphaRotate includes a detailed installation and tutorials across all models for clarity and ease of use, and all models are covered by unit testing, continuous integration, code coverage, maintainability checks, and visual monitoring and analysis.

4) AlphaRotate supports multi-GPU training and multi-process testing, and provides commonly used techniques such as data augmentation, multi-scale training and cropping, stochastic weights averaging, etc. to further boost model performance.

---

Table 1: Accuracy comparison of rotation detectors on DOTA (v1.0, v1.5 and v2.0) (Xia et al., 2018). O.C., L.E., and Quad. respectively denotes OpenCV Definition ($\theta \in [-90^\circ, 0^\circ]$), Long Edge Definition ($\theta \in [-90^\circ, 90^\circ]$), and quadrilateral box. ‘H’ and ‘R’ represent the horizontal and rotating anchors, respectively. All models are initialized by ResNet50 (He et al., 2016) without using data augmentation and multi-scale training and testing. mAP$_{50}$ represents mean of Average Precision calculated under the IoU threshold of 0.5 (like-wise for mAP$_{75}$ and mAP$_{50:95}$).

<table>
<thead>
<tr>
<th>Baseline</th>
<th>Improvement modules adding</th>
<th>Box Def.</th>
<th>v1.0</th>
<th>v1.0 train/val</th>
<th>v1.5</th>
<th>v2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>O.C.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>L.E.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IoU-Smooth L1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Yang et al., 2019)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSDet</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Qian et al., 2021)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSDet</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Qian et al., 2021)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Yang and Yan, 2020)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCL (BCL)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Yang et al., 2021a)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GWD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Yang et al., 2021c)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KLD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Yang et al., 2021d)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RetinaNet-H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Lin et al., 2017b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RetinaNet-R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Tian et al., 2019)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R$^3$Det</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Yang et al., 2021b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Lin et al., 2017a)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

benchmarks are based on Pytorch (Paszke et al., 2017) which can be less efficient than TensorFlow (Abadi et al., 2016) in terms of industrial deployment.
2. Project Focus

**Modular implementation:** As shown in Figure 1, the detectors are organized by eight components: *data, backbone, detector, neck, anchor head, roi extractor, bbox head, loss*. The core of AlphaRotate is a set of base classes and functions that is designed to allow for rapid and easy development of such models. Developers only need to add/delete/improve specific modules to build a new rotation detection model. The modular design improves code utilization and also helps developers to debug and troubleshoot. Through our test procedure, all functions and classes are tested, with a line coverage of over 92% of the code.

**Rich models and tools:** AlphaRotate supports more than 18 state-of-the-art rotation detection methods (keep on extension), including single-stage/two-stage methods, anchor-based/anchor-free methods, and supports training and testing on nearly ten datasets such as aerial images, scene text, and face, as shown in Figure 1. Besides, it provides fair comparisons of all methods (including hybrid methods) on representative DOTA dataset (Xia et al., 2018) to provide researchers with an accurate and comprehensive baseline, as listed in Table 1. Unless otherwise specified, all models are trained on the trainval set by default, and the result file of the test set is submitted to the official evaluation server\(^2\) to obtain the final evaluation result. We have provided download links of all baseline model weights.

**Open and collaborative development:** We have implemented AlphaRotate with an open and collaborative spirit and placed it under Apache-2.0 License. AlphaRotate is hosted on GitHub\(^3\), and developers can consult and discuss issues through the platform. In addition, external contributions and requests are encouraged and we enforce a strict rule on providing several tests for every new contribution and detected bug. When writing this paper, about 620 stars and 112 forks have been created and 47 issues have been resolved.

**Installation and tutorials:** AlphaRotate provides installation instructions, including dependent libraries and platform environment. We also provide developers with the Docker

---

\(^2\) https://captain-whu.github.io/DOTA/evaluation.html

\(^3\) https://github.com/yangxue0827/RotationDetection
image matched by the benchmark. We also provide detailed tutorials \(^4\), including how to convert dataset, compile files, train and test, and use visualization tools, etc.  

**Dependencies:** AlphaRotate is built as a Python application on Tensorflow (Abadi et al., 2016). It allows to use the benchmark on various platforms and devices like CPUs and GPUs. AlphaRotate relies on open source libraries such as NumPy, SciPy, Cython, OpenCV-Python, Matplotlib, Shapely, which are commonly used in literature and community.

3. **Toolbox Usage**

**Data pre-processing:** The input of the model mainly includes the image \(I\), the four-point coordinates of the object \((x_1, y_1, \ldots, x_4, y_4)\) and the category corresponding to the object \(L\). Then, dataset should be converted into tensorflow’s exclusive data reading format, tfrecord.

**Model training:** First, select the detector and dataset, and create the corresponding configuration file according to the plan under configs. Then run the training script under tools until the training is complete. During the training process, tensorboard can be used to observe the training status, including the loss curve of each stage and the image visualization. All trained models and log files are saved in the output. In the following example, the training process is easily implemented in a couple lines of code:

```python
import alpharotate
from alpharotate.libs.models.detectors import retinanet
from retinanet.build_whole_network import DetectionNetworkRetinaNet
from configs import cfgs
# init detector (e.g. RetinaNet)
model = DetectionNetworkRetinaNet(cfgs=cfgs, is_training=True)
# training
_, loss = model.build_whole_detection_network(input_img_batch=img, gt=gt)
```

**Model testing and evaluation:** Under the tools, it provides test and evaluation scripts for evaluation (e.g. mAP, F-measure), visualizing results, and generating result files.

```python
# init detector (e.g. RetinaNet)
model = DetectionNetworkRetinaNet(cfgs=cfgs, is_training=False)
# inference
bbox, score, category = model.build_whole_detection_network(input_img_batch=img)
```

4. **Conclusion**

With the practical importance and academic emergence for visual rotation detection, AlphaRotate is a deep learning benchmark for visual object rotation detection in Tensorflow under the Apache-2.0 license. The architecture is designated for both flexibility and ease of use, with the goal of facilitating the deployment of rotation detection in diverse domains, both in industrial applications and in academic research. We will continue to improve the entire optimized benchmark and support representative detection methods in the future. We also welcome the community to participate in the development.

\(^4\) https://alpharotate-tutorial.readthedocs.io/
References


