AHMEDABAD UNIVERSITY

SCHOOL OF ENGINEERING AND APPLIED SCIENCE

Winter Semester 2024

CSE-541 Computer Vision

Team Number: 3

Members:

Khwahish Patel	**Krishang Shah**	**Sachin Dindor**	**Dhruvesh Panchal**

Project 6: Explore oriented object detection (OOD) models. Create our own AU drone

- **dataset for such a model and then test/validate trained models.**

WEEKLY REPORT

(Week 6)

(11/03/2024 - 17/03/2024)

Tasks Completed:

- ➔ Explored other datasets like VISDrone, HRSID, etc., to meet our requirements to be fulfilled.
- ➔ We have explored various codes to be implemented to get the bounding box across the dataset.
- ➔ We have tried replacing datasets other than DOTA to get faster results and more efficient computing.
- ➔ Implemented certain models that include the labeling bounding box.

Challenges Faced:

- ➔ Implementing various versions of YOLO did not work out.
- ➔ Not having access to GPUs hindered our progress a lot.
- ➔ More processing time is involved due to the lack of more accessible resources.
- ➔ Implementing every code that we found for bounding boxes on the dataset, did not yield efficient results.

Code that worked out:

```
# Check nvcc version
!nvcc -V
# Check GCC version
!gcc --version

# install dependencies: (use cu111 because colab has CUDA 11.1)
!pip install torch==1.9.0+cu111 torchvision==0.10.0+cu111 -f
https://download.pytorch.org/whl/torch_stable.html

# install mmcv-full thus we could use CUDA operators
!pip install mmcv-full -f
https://download.openmmlab.com/mmcv/dist/cu111/torch1.9.0/index.html

# Install mmdetection
!pip install mmdet

# Install mmrotate
!rm -rf mmrotate
!git clone https://github.com/open-mmlab/mmrotate.git
%cd mmrotate
!pip install -e .

# switch branch
!git checkout dev
!git branch -a

from mmcv import collect_env
collect_env()

# Check Pytorch installation
import torch, torchvision
print(torch.__version__, torch.cuda.is_available())

# Check MMRotate installation
import mmrotate
print(mmrotate.__version__)
```

```
# Check MMDetection installation
import mmdet
print(mmdet.__version__)

# Check mmcv installation
from mmcv.ops import get_compiling_cuda_version, get_compiler_version
print(get_compiling_cuda_version())
print(get_compiler_version())

# We download the pre-trained checkpoints for inference and finetuning.
!mkdir checkpoints
!wget -c
https://download.openmmlab.com/mmrotate/v0.1.0/oriented_rcnn/oriented_rcnn_r50_
fpn_1x_dota/oriented_rcnn_r50_fpn_1x_dota_20200909-fab9f4fb.pth \
      -O checkpoints/oriented_rcnn_r50_fpn_1x_dota_20200909-fab9f4fb.pth

import mmcv
from mmcv.runner import load_checkpoint

from mmdet.apis import inference_detector, show_result_pyplot
from mmrotate.models import build_detector

# Choose to use a config and initialize the detector
config_path = 'mmrotate/configs/oriented_rcnn/oriented_rcnn_r50_fpn_1x_dota.py'
# Setup a checkpoint file to load
checkpoint = 'checkpoints/oriented_rcnn_r50_fpn_1x_dota_20200909-fab9f4fb.pth'

# Set the device to be used for evaluation
device = 'cuda:0' if torch.cuda.is_available() else 'cpu'

# Load the config
config = mmcv.Config.fromfile(config_path)
# Set pretrained to be None since we do not need pretrained model here
config.model.pretrained = None

# Initialize the detector
model = build_detector(config.model)

# Load checkpoint onto CPU if CUDA is not available
checkpoint = load_checkpoint(model, checkpoint, map_location=device)

# Set the classes of models for inference
model.CLASSES = checkpoint['meta']['CLASSES']
```

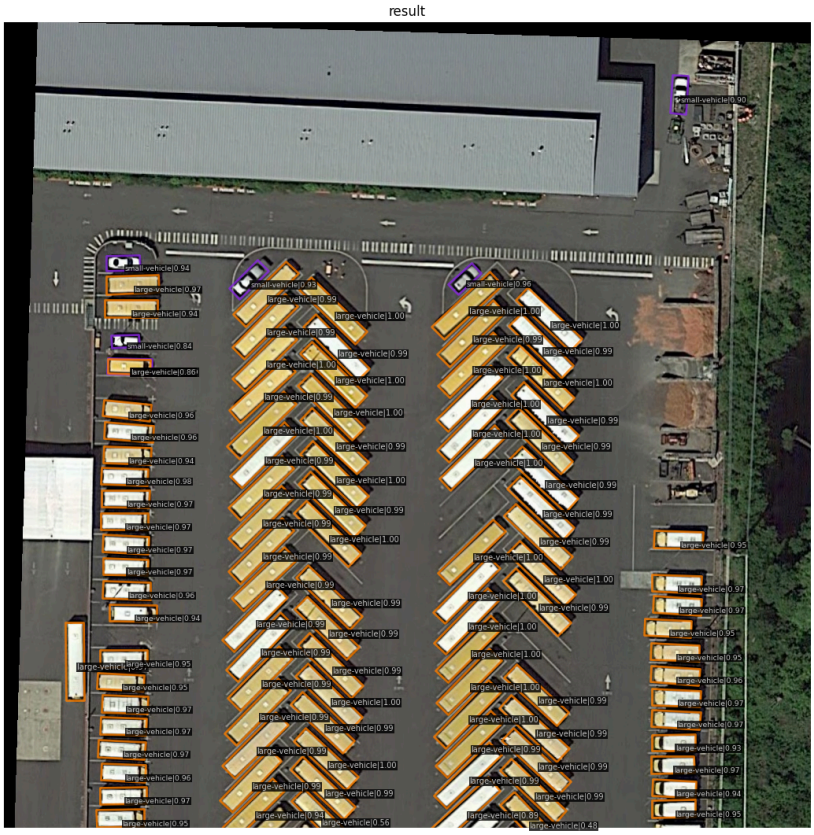
```
# We need to set the model's cfg for inference
model.cfg = config

# Convert the model into evaluation mode
model.eval()

# Use the detector to do inference
img = 'demo/demo.jpg'
result = inference_detector(model, img)

# Let's plot the result
show_result_pyplot(model, img, result, score_thr=0.3, palette='dota')
```

Output:



Next steps:

- ➔ We are focusing on implementing R-Box CNN, R-CNN, and Faster R-CNN models for oriented bounding boxes.
- ➔ We are studying the methodology or features involved in the model that enables the model to create oriented bounding boxes.
- ➔ We are studying how the model understands that the vehicle is making a turn and is accordingly making a bounding box around it.