Enhancing Small Object Detection in Aerial Imagery: A Comparative Study of YOLO and RT-DETR Models Using Slicing Aided Hyper Inference

Brief Overview of the Project

This project focuses on **Enhancing small object detection in aerial imagery** using advanced deep learning models, specifically **YOLO (You Only Look Once)** and **RT-DETR (Real-Time Detection Transformer)**. The research addresses the significant challenges posed by low resolution and complex backgrounds in aerial images, which make traditional detection methods ineffective. By employing the **Slicing Aided Hyper Inference (SAHI)** technique, the project aims to improve detection accuracy for small objects, which is crucial for applications in urban planning, traffic monitoring, and disaster management. The models are trained on the VisDrone dataset over 200 epochs, and their performance is compared against standard inference and SAHI inference techniques to evaluate improvements in detection metrics.

System Requirements

For this project, **AWS EC2 instance** is used, below is the specification

EC2 Instance type	g5.xlarge
AMI name	NVIDIA GPU Cloud VMI Base 2024.05.1 x86_64-676eed8d-dcf5-4784-87d7-0de463205c17
Operating system	Ubuntu 22.04
CPU RAM memory	16 GB
Storage	378 GB
Storage Type	SSD
GPU memory	24 GB

Essential packages and their versions

nvidia-cublas-cu12	**12.4.5.8**
nvidia-cuda-cupti-cu12	**12.4.127**
nvidia-cuda-nvrtc-cu12	**12.4.127**
nvidia-cuda-runtime-cu12	**12.4.127**
nvidia-cudnn-cu12	**9.1.0.70**
nvidia-cufft-cu12	**11.2.1.3**
nvidia-curand-cu12	**10.3.5.147**
nvidia-cusolver-cu12	**11.6.1.9**
nvidia-cusparse-cu12	**12.3.1.170**
nvidia-nccl-cu12	**2.21.5**
nvidia-nvjitlink-cu12	**12.4.127**
nvidia-nvtx-cu12	**12.4.127**
sahi	**0.11.19**
torch	**2.5.1**
torchvision	**0.20.1**
ultralytics	**8.3.31**
ultralytics-thop	**2.0.11**

Software Installation

Once you done the AWS EC2 setted up, connect to the instance using the .pem file with below command.

```
ssh -i "ResearchProject.pem" ubuntu@ec2-3-133-88-41.us-east-2.compute.amazonaws.com
```

Once you are connected to your instance from your local, run the following commands to install the packages which we needed for our project.

```
sudo apt update
sudo apt upgrade
nvidia-smi (info)
nvcc -v (cuda toolkit version info)
sudo apt install nvidia-cuda-toolkit
sudo reboot now
```

After running the above commands, the ec2 instance will restart to make the cuda library available for our development. Now we need to setup our python environment. Run the below commands

```
mkdir Research_Project
cd Research_Project/
python3 -m venv venv
source venv/bin/activate
```

The above command will create a folder called Research_Project and also a python environment inside the folder. Now we are going to install the python packages needed for our project using the below command.

```
pip install numpy pandas matplotlib seaborn
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu124
pip install ultralytics sahi
```

Now clone the github url to download the project files from github

```
HTTPS: https://github.com/ajaykkumar-nci/Research_Project.git
or
Download .zip : https://github.com/ajaykkumar-nci/Research_Project/archive/refs/heads/main.zip
```

Now the project directory is ready, download the Visdrone image dataset, the below commands will download the train,test-dev,val datasets of Visdrone images

```
pip3 install gdown

mkdir Visdrone_images
cd Visdrone_images/

Train dataset
gdown https://drive.google.com/file/d/1a2oHjcEcwXP8oUF95qiwrqzACb2YlUhn/view?usp=sharing
--fuzzy
```

Test dataset
gdown https://drive.google.com/file/d/1PFdW_VFSCfZ_sTSZAGjQdifF_Xd5mf0V/view?usp=sharing --fuzzy

Val dataset
gdown https://drive.google.com/file/d/1bxK5zgLn0_L8x276eKkuYA_FzwCIjb59/view?usp=sharing --fuzzy

Configuration settings

The configuration file for running the project cfg/Visdrone.yaml

path: ../ (Specify the absolute path of the Visdrone_images dataset)

Below are the paths to train, test, val datasets from Visdrone_images

train: VisDrone2019-DET-train/images
val: VisDrone2019-DET-val/images
test: VisDrone2019-DET-test-dev/images

6 Distinct Classes of the dataset
names:
0: people
1: bicycle
2: car
3: truck
4: bus
5: motor

Apart from the above file, there is a utils.py file, in which

```
convert2yolo("./Visdrone_images", ['VisDrone2019-DET-train', 'VisDrone2019-DET-test-dev', 'VisDrone2019-DET-val/images'])
```

Make sure the path is configured correctly because the above file converts the Visdrone annoation format to YOLO supported format.

There is another file vis_to_coco_util.py file in which

```
visdrone_to_coco('./Visdrone_images/VisDrone2019-DET-test-dev', './Visdrone_images/VisDrone2019-DET-test-dev/coco_annotations.json')
```

The above file will convert the yolo to coco annotation format to calculate the metrics while running test dataset with SAHI testing.

The above are the places where absolute path of the dataset has to be checked and corrected if necessary.

Data Preparation

Step 1

Run the utils.py file to convert the Visdrone annotation format to YOLO supported annotation format

```
python3 utils.py
```

Step 2

Run the vis_to_coco_util.py file to convert the test dataset to COCO annotation format

```
python3 vis_to_coco_util.py
```

Running the Experiment

Step 1

Train the YOLO and RT-DETR models for 200 epochs with our Visdrone Train dataset. Run the following train commands one by one and run it each gets completed.

```
cd src/

python3 yolov8-train.py

python3 yolov11-train.py

python3 rt-detr-train.py
```

Once all these files are done, the results of these training datasets are saved in results folder inside a folder is created for each model and their weights are saved.

Note: You can also download the already Pre-trained model results into the project, as the github has the file size limitation, the results were not present in github. Use the below command to download the results from google drive

```
gdown https://drive.google.com/file/d/1Si2C6F4s0pjmi1KalrIhiva8MDF8pT02/view?usp=drive_link --fuzzy
```

After downloading the results, put the results folder inside the project Research_Project/result

Step 2

Now to test the model, execute the following commands

```
cd ../test/

python3 test.py
```

The above command will execute Standard testing inference for all 3 models , followed by SAHI testing for each of them. Once the testing is done, the results are saved in a **visdrone_model_evaluation.csv**

Below is the output of Standard testing of the models:

Model	mAP50	mAP50-95	Average Precision	Average Recall
Yolov8	0.3665160074	0.2410608118	0.2334197125	0.2811547576
Yolov11	0.3719482585	0.2436852602	0.2355937776	0.282450114
RT-DETR	0.4142389391	0.2695400586	0.2334009745	0.2983507726

Below is the output of SAHI testing of the models:

Model	mAP50	mAP50-95	Average Precision	Average Recall
Yolov8	0.4573050473	0.2924227389	0.2776737485	0.357048117
Yolov11	0.463131706	0.2973265082	0.2810449035	0.3598532199
RT-DETR	0.4716393693	0.3024496859	0.2674128088	0.354603568

From above results, the outputcan be cross verified. ## References

https://github.com/VisDrone/VisDrone-Dataset (Visdrone dataset github url)

https://docs.ultralytics.com/models/ (Ultralytics Model documentation for Yolov8, Yolov11 & RT-DETR)

https://github.com/obss/sahi (SAHI github url)

https://www.youtube.com/watch?v=RVMAyyVGAC4 (Install Nice DCV for AWS EC2 Instance which is a remote desktop for the instance)