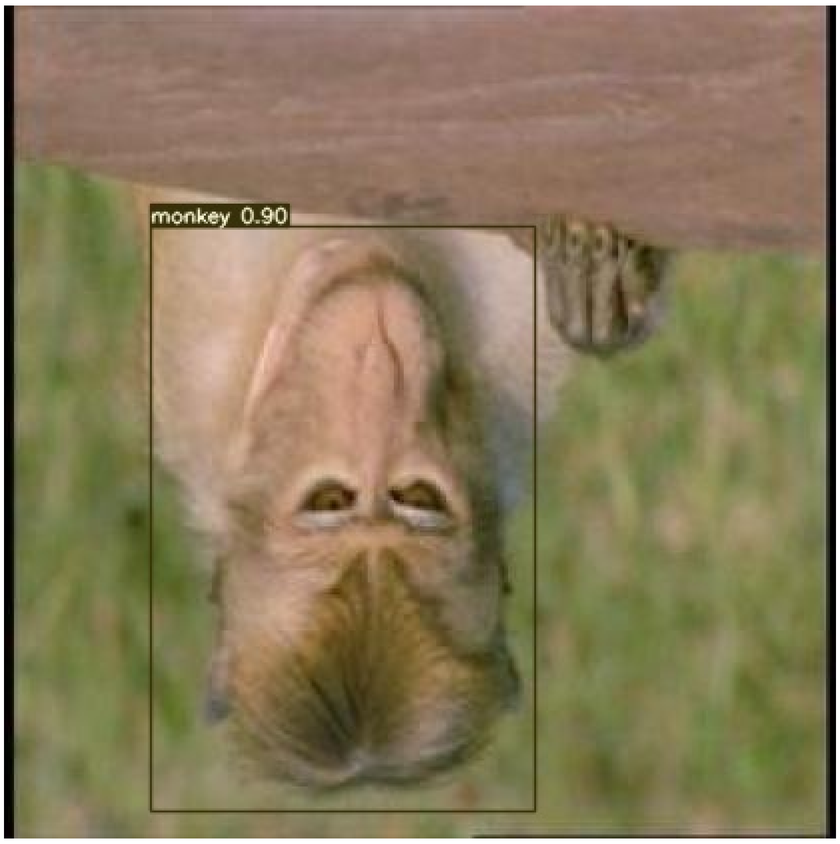
INDIAN INSTITUTE OF TECHNOLOGY
BOMBAY

ASSIGNMENT-REPORT

Submitted by – VENKATARAMI REDDY

Monkey Detection using Computer Vision:

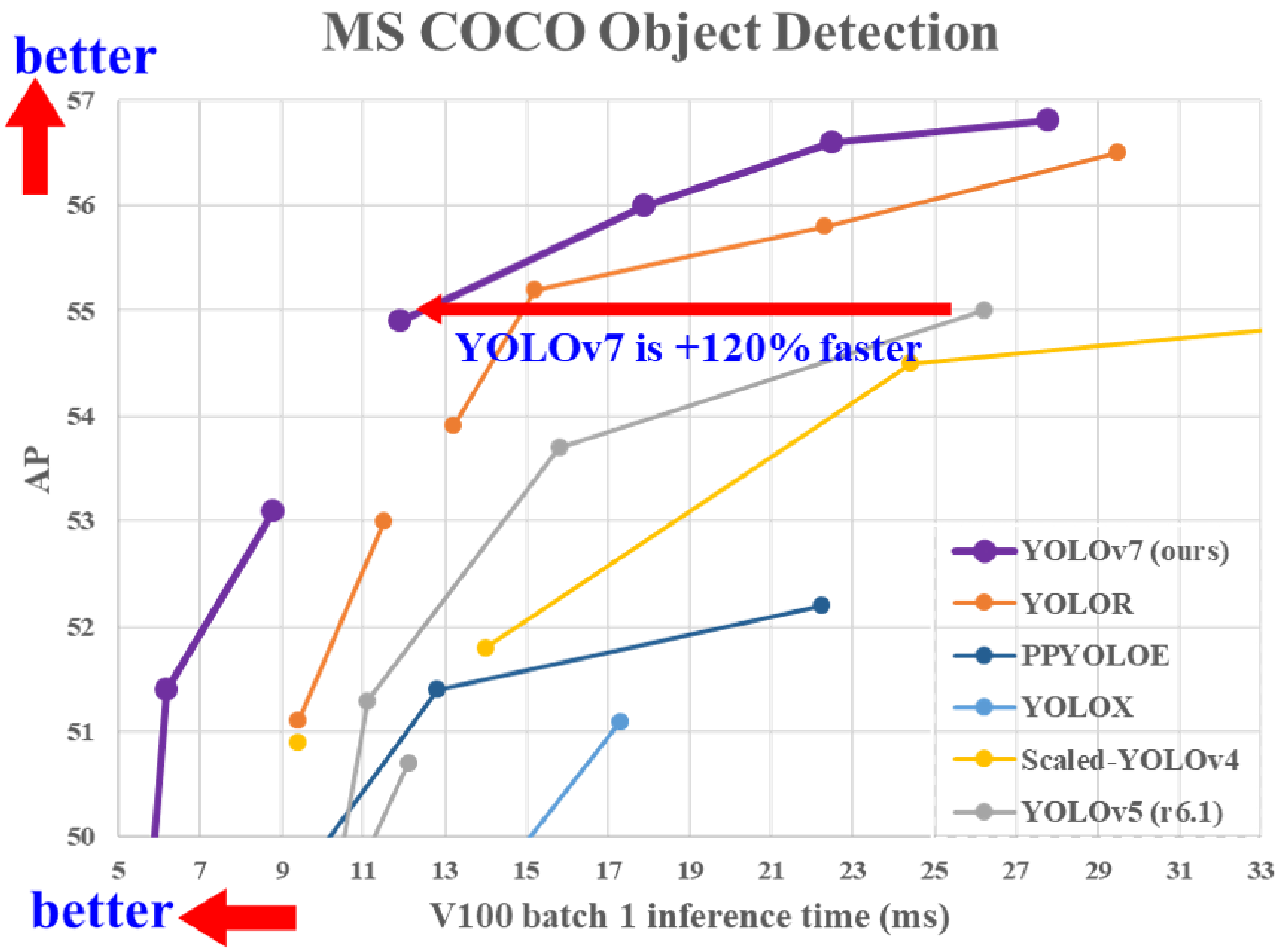


Introduction-

This is an Assignment in the process of Machine Learning Internship Recruitment by IIT Bombay

Monkey Detection using Computer Vision using AI. It uses Live videos from camera feed or rtsp streaming from IP camera or CCTV or Input image and use object detection to detect Monkey in these feeds. In this project I used the YOLOV7 model to detect Monkey in the video feed.

YOLOV7:



Performance:

MS COCO

Model	Test Size	AP^{test}	AP_{50}^{test}	AP_{75}^{test}	batch 1 fps	batch 32 average time
YOLOv7	640	51.4%	69.7%	55.9%	161 *fps*	2.8 *ms*
YOLOv7-X	640	53.1%	71.2%	57.8%	114 *fps*	4.3 *ms*
YOLOv7-W6	1280	54.9%	72.6%	60.1%	84 *fps*	7.6 *ms*
YOLOv7-E6	1280	56.0%	73.5%	61.2%	56 *fps*	12.3 *ms*
YOLOv7-D6	1280	56.6%	74.0%	61.8%	44 *fps*	15.0 *ms*
YOLOv7-E6E	1280	56.8%	74.4%	62.1%	36 *fps*	18.7 *ms*

Installation of YOLOV7 :

Docker environment (recommended)

```
# create the docker container, you can change the share memory size if you have more.
nvidia-docker run --name yolov7 -it -v your_coco_path/:/coco/ -v your_code_path/:/yolov7 --shm-size=64g
nvcr.io/nvidia/pytorch:21.08-py3

# apt install required packages
apt update
apt install -y zip htop screen libgl1-mesa-glx

# pip install required packages
pip install seaborn thop

# go to code folder
cd /yolov7
```

Testing :

```
python test.py --data data/coco.yaml --img 640 --batch 32 --conf 0.001 --iou 0.65 --device 0 --weights yolov7.pt --name yolov7_640_val
```

You will get the results :

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.51206
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.69730
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.55521
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.35247
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.55937
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.66693
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.38453
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.63765
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.68772
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.53766
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.73549
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.83868
```

To measure accuracy, download COCO-annotations for Pycocotools to the ./coco/annotations/instances_val2017.json

Data Preprocessing :

Data preprocessing is an important step for the creation of a machine learning model. Initially, data may not be clean or in the required format for the model which can cause misleading outcomes. In pre-processing of data, we transform data into our required format. It is used to deal with noises, duplicates, and missing values of the dataset. Data pre-processing has the activities like importing datasets, splitting datasets, attribute scaling, etc. Preprocessing of data is required for improving the accuracy of the model.

Requirements :

System:

System requirements depends on complexity or size of object detection model, larger model will require more compute power and will be good at detection. I have used this in Google Colab with Tesla T4 GPU & Kaggle with Tesla p100 GPU which is a good device. With my Nvidia Geforce 330 MX GPU based system it is not sufficient to train a YOLOV7 model.

Minimum of 4GB NVIDIA Graphics Card is required to train the YOLOV7 Model.

Python :

Python 3.6 or higher. Tested with Python 3.6, 3.7,3.8,3.9,3.10 in windows 11 and linux.

Packages :

1. Pytorch
2. Numpy
3. OpenCV-Python

The Implementation is tested with Pytorch gpu in Windows 11 and Linux, Google Colab and Kaggle.

Installation :

Prerequisites :

All the dependencies and required libraries are included in the file requirements.txt See here

Install Python:

There are two ways to install python in windows using Python 3 installer or Anaconda. Installing python with anaconda or miniconda is recommended. In linux Python 3 is installed by default but we can also install miniconda or conda into linux.

Creating Virtual Environment:

Create a new python virtual environment using conda or venv and activate it. If Anaconda or Miniconda is installed use `conda` else use `venv` to create virtual environments.

1. Using Conda

```
conda create --name security_cam
conda activate security_cam
conda install pip
```

2. Using Venv in Linux

```
Python3 -m venv path/to/create/env/security_cam
Source path/to/create/env/security_cam/bin/activate
```

3. Using Venv in Windows

```
Python -m venv path\to\create\env\security_cam
path\to\create\env\security_cam\Scripts\activate
```

Installing Dependencies:

The command below will install all the required dependencies from requirements.txt :

```
pip install -r requirements.txt
```

Setup (On Production Machine) :

Download the YOLOV7 model from here and place it in the root directory of the project. Then copy the dataset you downloaded from Roboflow to the YOLOV7-MAIN Folder and add the data.yaml file in the main directory of the project. The data.yaml file should contain the following information: (The dataset folder name is monkey_data)

Note: Clone the YOLOV7 repository and do not download as a zip since a .git folder is required for training and create a "static" and "results" named empty folder after copying templates folder to YOLOV7.

```
git clone https://github.com/WongKinYiu/yolov7.git
```

The data.yaml file should contain the following information:

```
train: monkey_data/train/images
val: monkey_data/valid/images
nc: 1
names: ["monkey"]
```

1. Change your directory to the cloned repo

```
cd "yolov7-main"
```

2. Also copy the files from this repository to the YOLO Folder same as the structured folder in the given drive link.

https://drive.google.com/file/d/1fL9VM5kW1jm0_m8rXufzoNuznVLG4fQD/view?usp=sharing

3. I used "Flask" as an API to connect the model with the web application.
4. Check out the flask file here app.py
5. Then I created a Docker image using Dockerfile and pushed it to the Docker Hub.
6. We can also use Docker-compose to run the application.

How to Run :

1. Open terminal. Go into the cloned project directory and type the following command:

```
python3 train.py --epochs 128 --workers 8 --device 0 --batch-size 32
--data data.yaml --img 416 416 --cfg cfg/training/yolov7.yaml --weights
'' --name Monkey-Model --hyp data/hyp.scratch.p5.yaml
```

2. After training is done, run the following command to test the model on a video/image:

```
py detect.py --weights best.pt --project results --name results
--source static/img.jpg --save-txt --save-conf
```

3. If you want to try the Project then run the flask file app.py

```
python app.py
```

Additional Guides :

If get stuck in installation part follow these additional resources

- [Python Installation](https://www.youtube.com/watch?v=YYXdXT2l-Gg&list)
- [pip and usage](https://www.youtube.com/watch?v=U2ZN104hIcc)
- [Anaconda installation and using conda](https://www.youtube.com/watch?v=YJC6ldI3hWk)

Documentation :

Input Parameters :

The Input Parameters used in this project are

1. --epochs: Number of epochs to train the model.
2. --workers: Number of workers to use for data loading.
3. --device: Device to use for training.(0 for GPU and 1 for CPU)
4. --batch-size: Batch size for training.
5. --data: Path to data.yaml file.
6. img: Image size.
7. --cfg: Path to model.yaml file.
8. --weights: Path to weights file. (Leave blank for training from scratch)
9. --name: Name of the model.
10. --hyp: Path to hyperparameters file.

Error Handling :

Mostly you are supposed to get errors regarding cv2.imwrite() to a particular folder, At that time you need to create respected folders in the main directory.

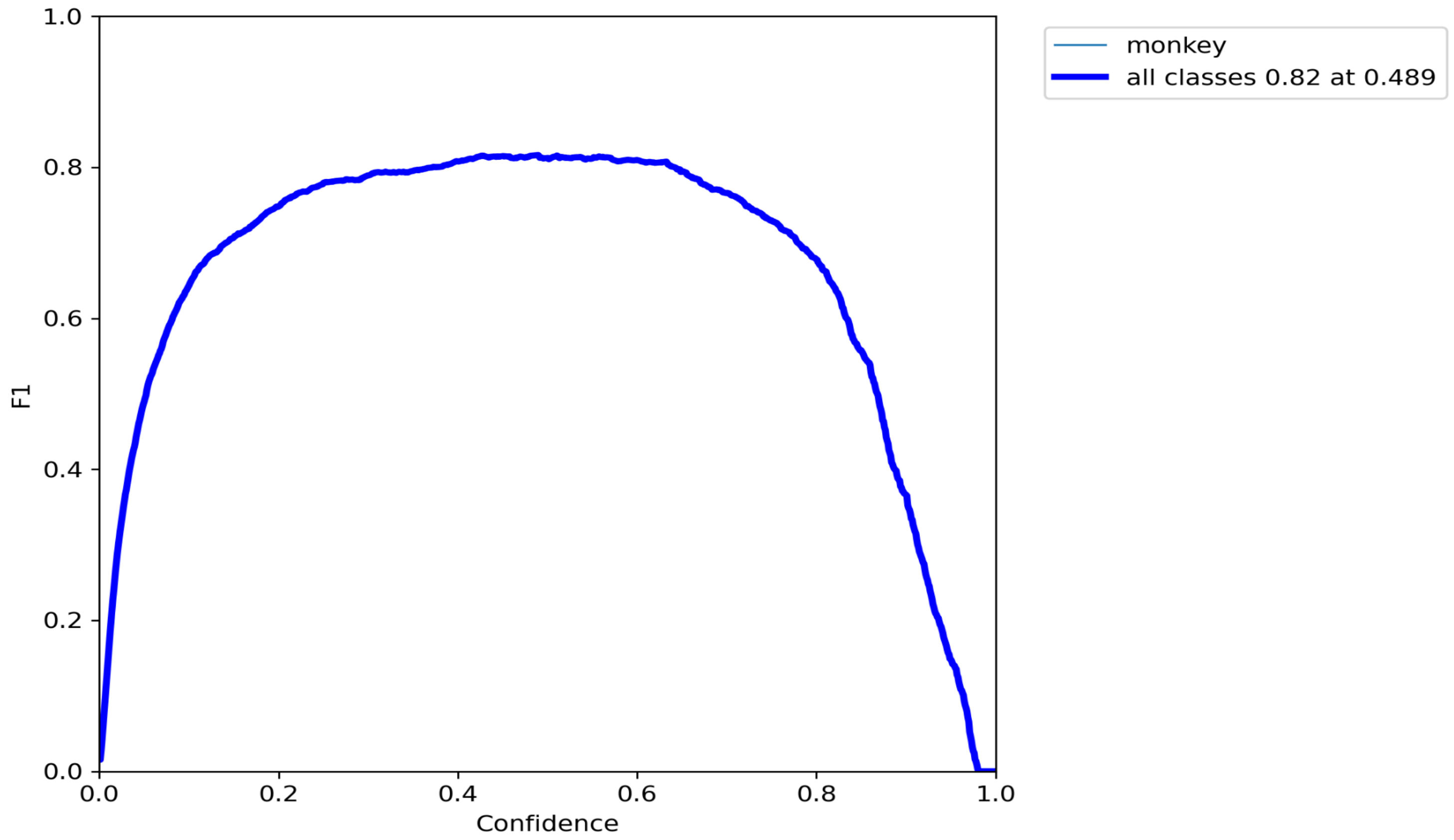
Directory Tree :

```
├───cfg
│   ├───baseline
│   ├───deploy
│   └───training
├───data
├───deploy
│   └───triton-inference-server
│       └───data
├───models
│   └───__pycache__
├───monkey_data
│   ├───test
│   │   ├───images
│   │   └───labels
│   ├───train
│   │   ├───images
│   │   └───labels
│   └───valid
│       ├───images
│       └───labels
├───results
├───runs
│   └───train
│       └───Monkey-Model
│           └───weights
├───scripts
├───static
├───templates
├───tools
└───utils
    ├───aws
    ├───google_app_engine
    ├───wandb_logging
    └───__pycache__
```

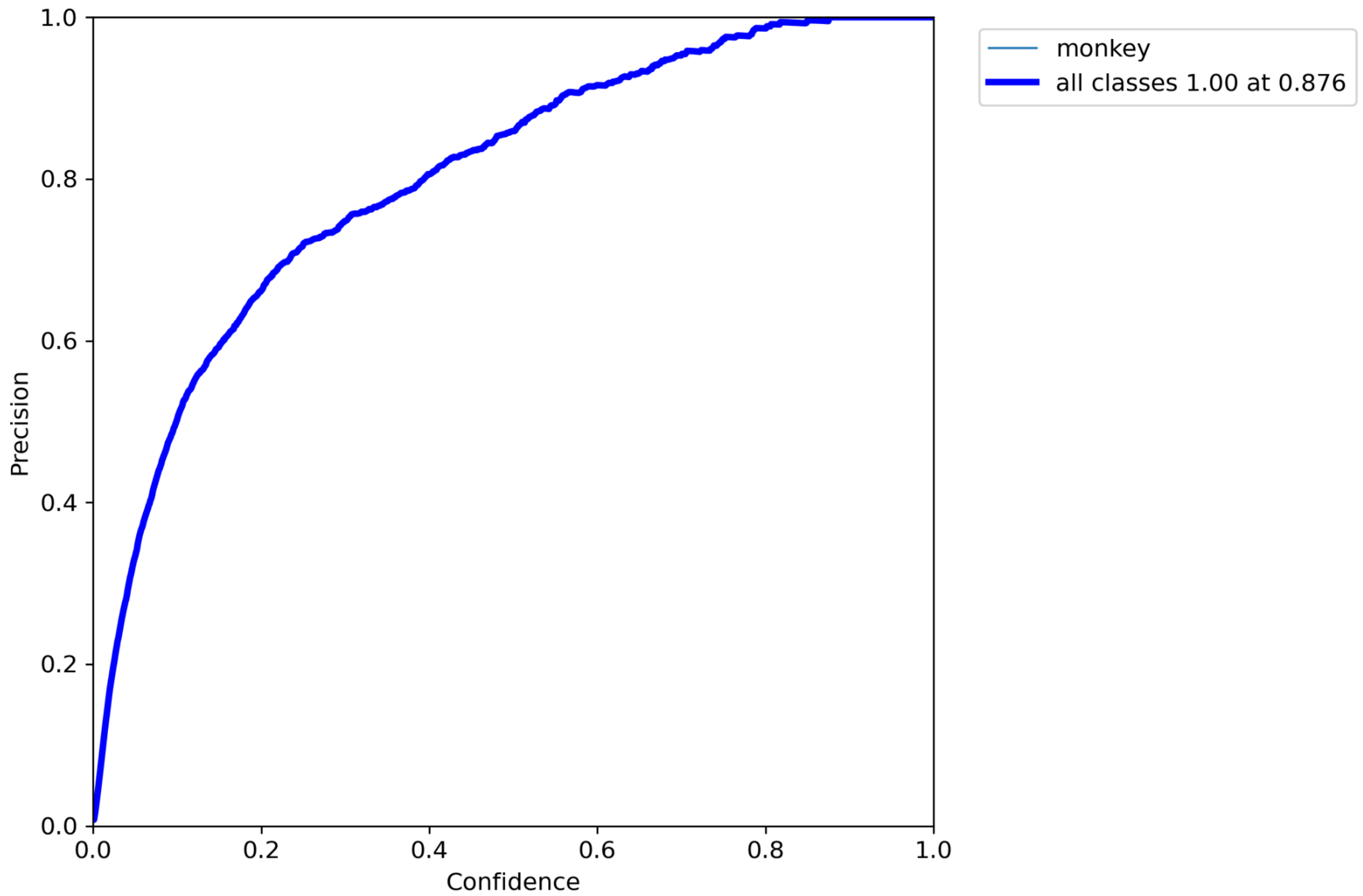
Results :

Our YOLOV7 Model gives the best accuracy and some good F1 Scores etc., & you can see below :

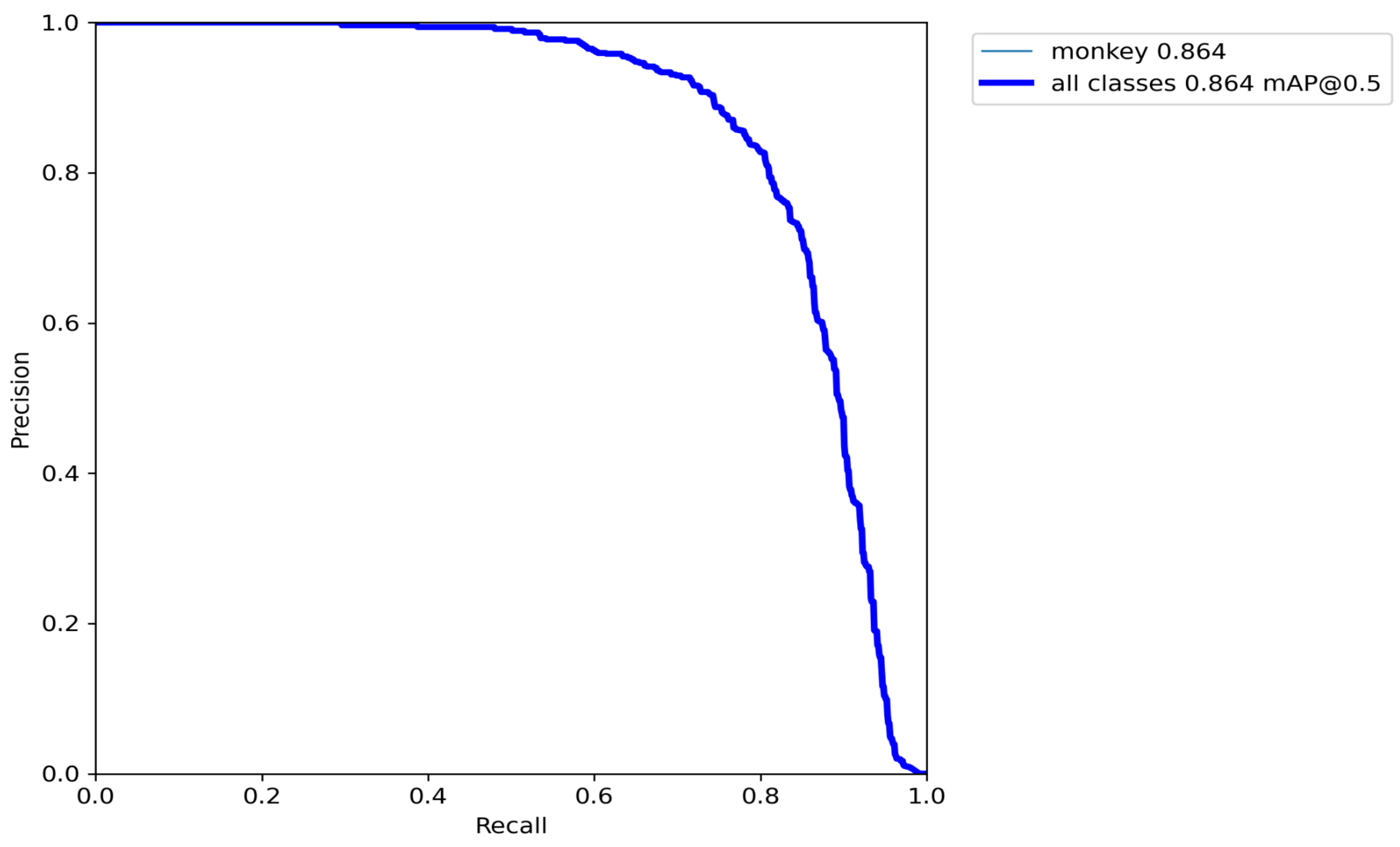
F1 Score:



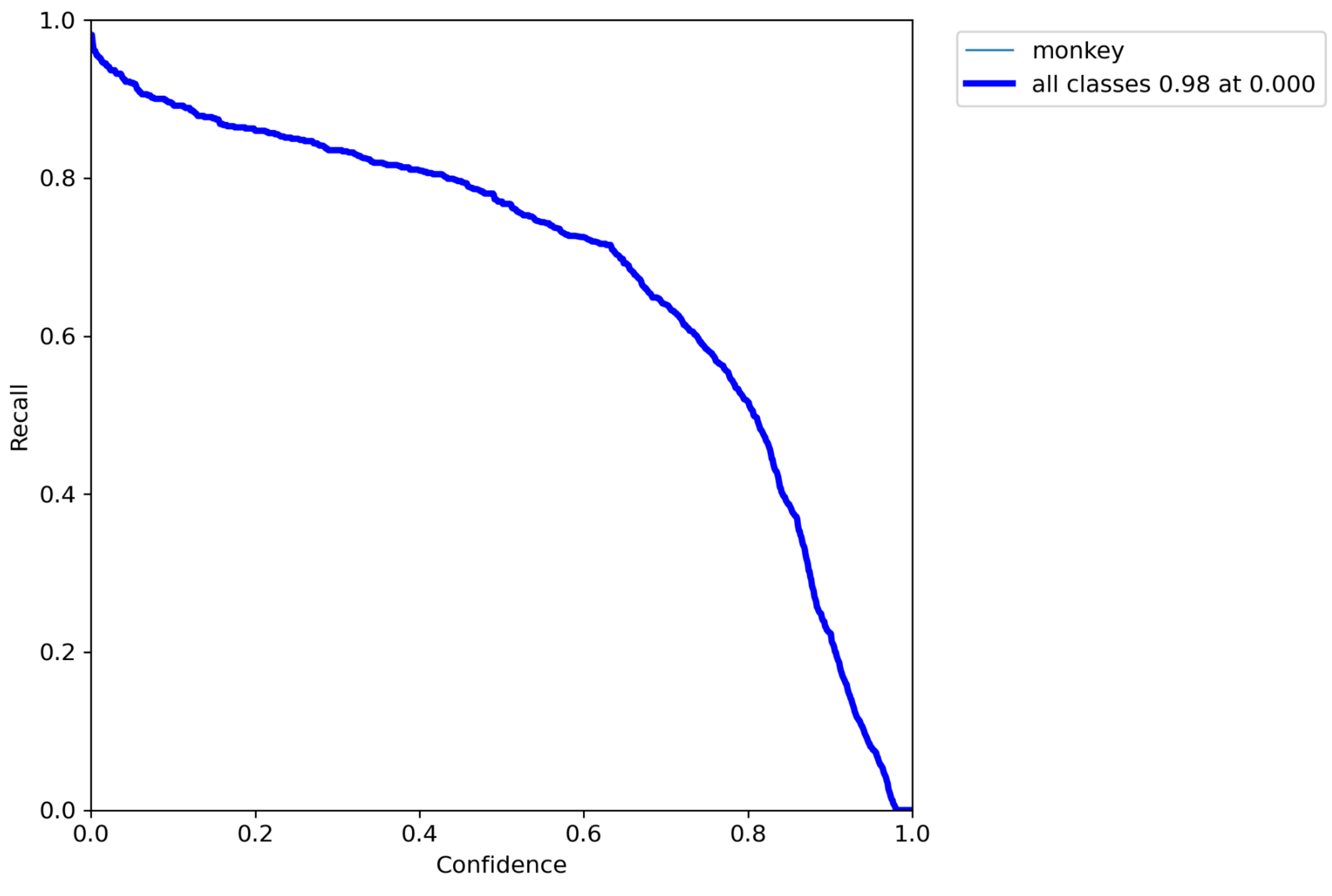
P Curve:



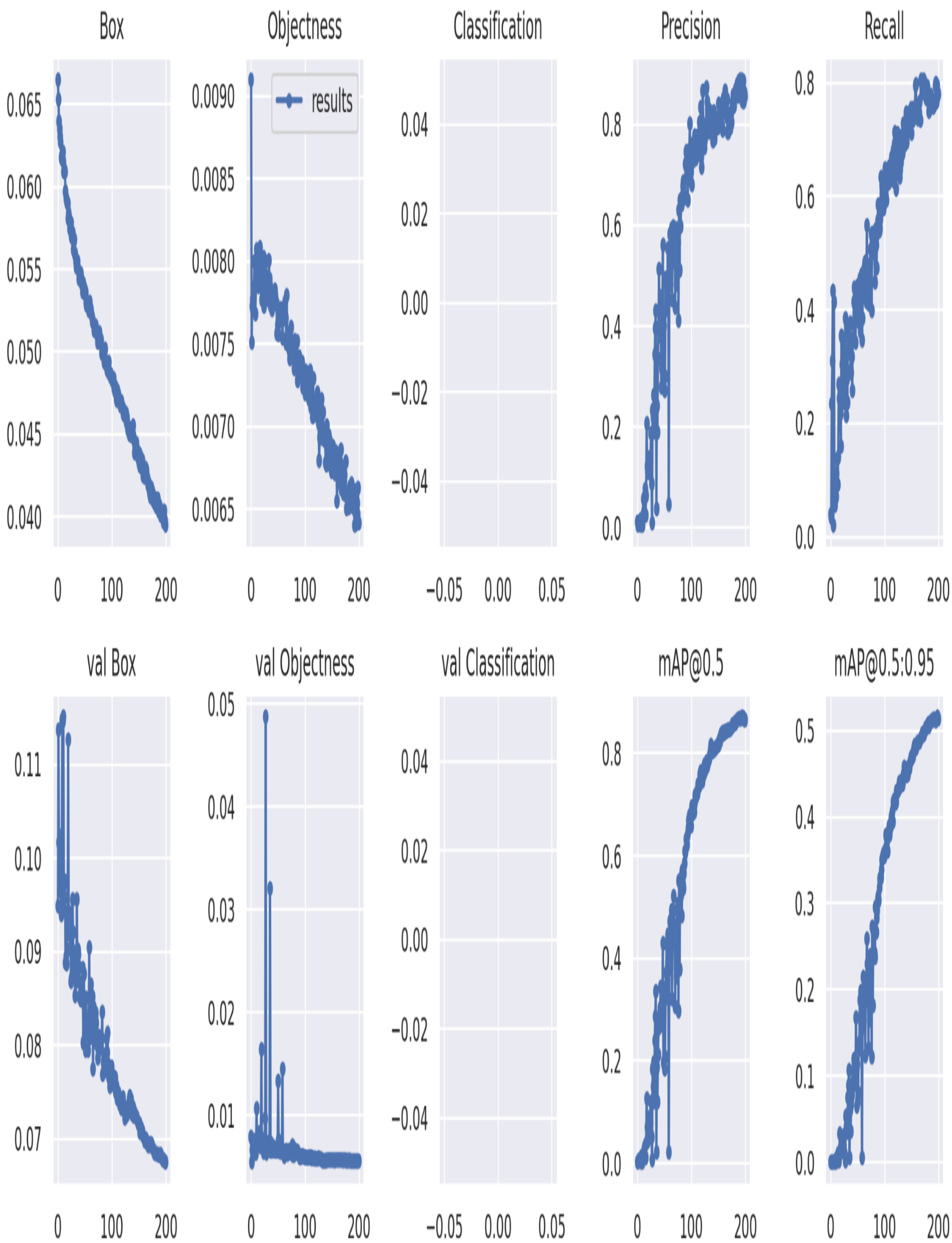
PR Curve:



R Curve:



Results:

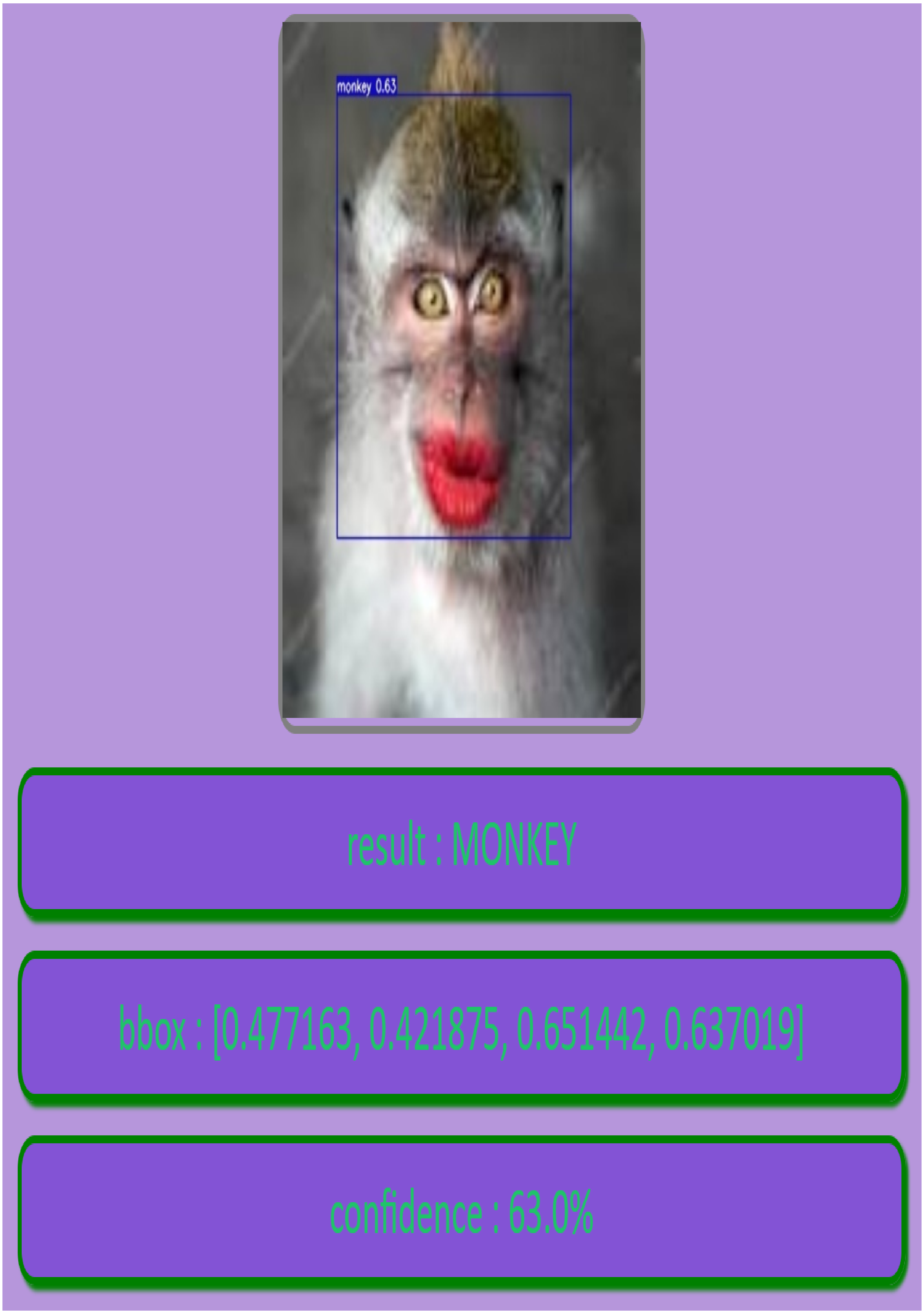


Box
Objectness
results
Classification
Precision
Recall
val Box
val Objectness
val Classification
mAP@0.5
mAP@0.5:0.95

Test Cases

Test Case 1:

Test Case 2:


monkey 0.63
result : MONKEY
bbox : [0.477163, 0.421875, 0.651442, 0.637019]
confidence : 63.0%

Test Case 3:


monkey 0.71
result : MONKEY
bbox : [0.38101, 0.582933, 0.65625, 0.675481]
confidence : 71.0%

References :


```
@article{wang2022yolov7,
  title={{YOLOv7}: Trainable bag-of-freebies sets new state-of-the-art
for real-time object detectors},
  author={Wang, Chien-Yao and Bochkovskiy, Alexey and Liao, Hong-Yuan
Mark},
  journal={arXiv preprint arXiv:2207.02696},
  year={2022}
}
```

```
@article{security-camera,
  title={{security-camera}: Intelligent security camera using object
detection},
  author={Tarun Bisht},

journal={application-of-object-detection-in-home-surveillance-system},
  year={2022}
}
```