

6 yolov4-tiny

6 yolov4-tiny

- 6.1 Introduction
- 6.3 folder structure
- 6.4 Environmental requirements
- 6.5 Custom training data set
 - 6.5.1 Making a dataset
 - 6.5.2 Add weight file
 - 6.5.3 Make label file
 - 6.5.4 Modify the train.py file
 - 6.5.5 Model checking

yolov4-tiny official website: <https://github.com/AlexeyAB/darknet>

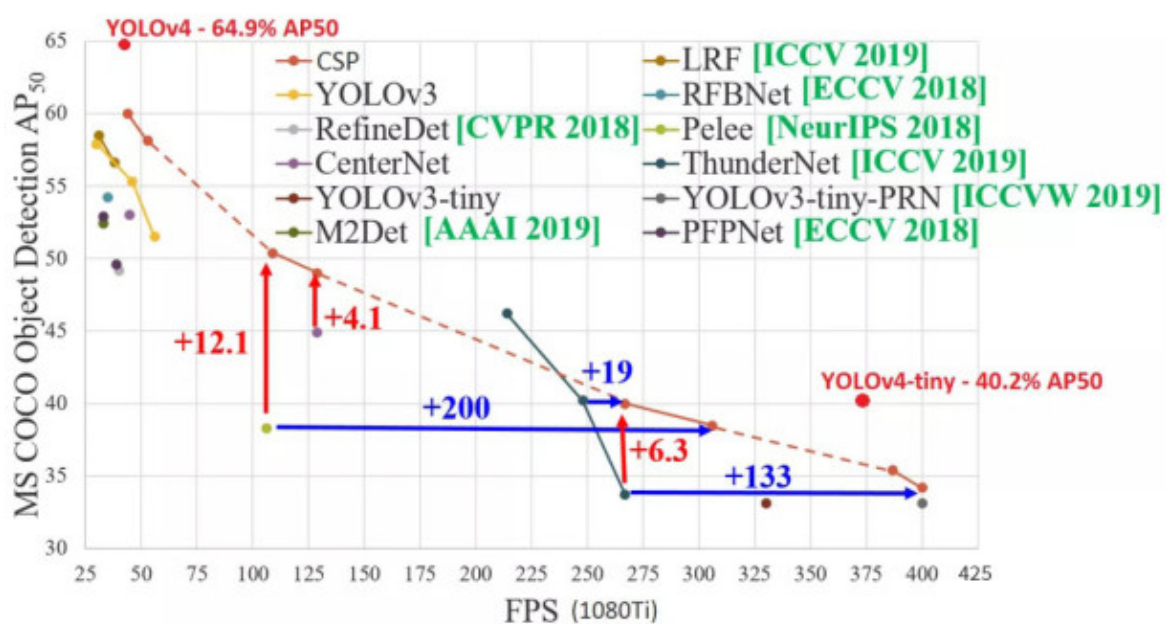
Source code: <https://github.com/bubbliiiing/yolov4-tiny-tf2>

6.1 Introduction

release time point

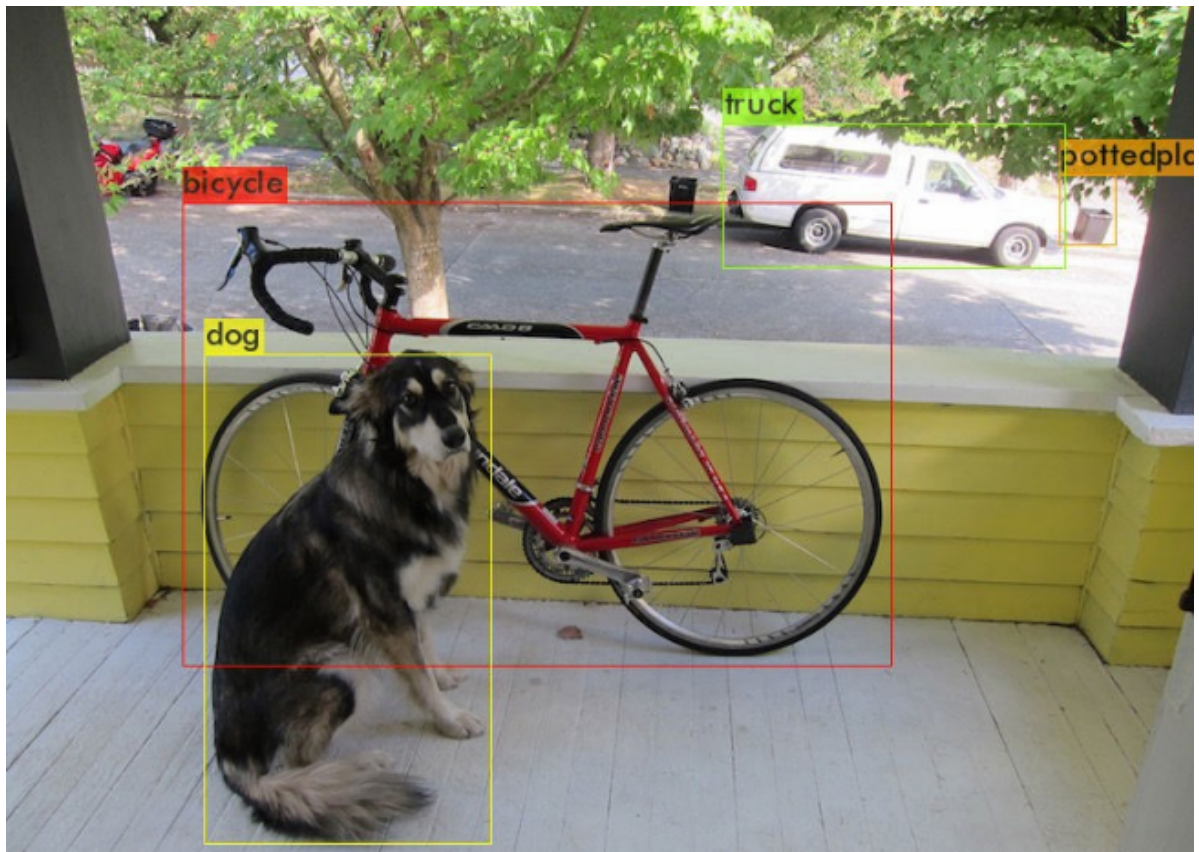
- 2020.04: YOLOv4 officially released
- 2020.06: YOLOv4-Tiny is officially released

The performance of YOLOv4-Tiny on COCO: **40.2% AP50, 371 FPS(GTX 1080 Ti)** Whether it is AP or FPS performance, it is a huge improvement compared to YOLOv3-Tiny, Pelee, and CSP, as shown in the figure below :



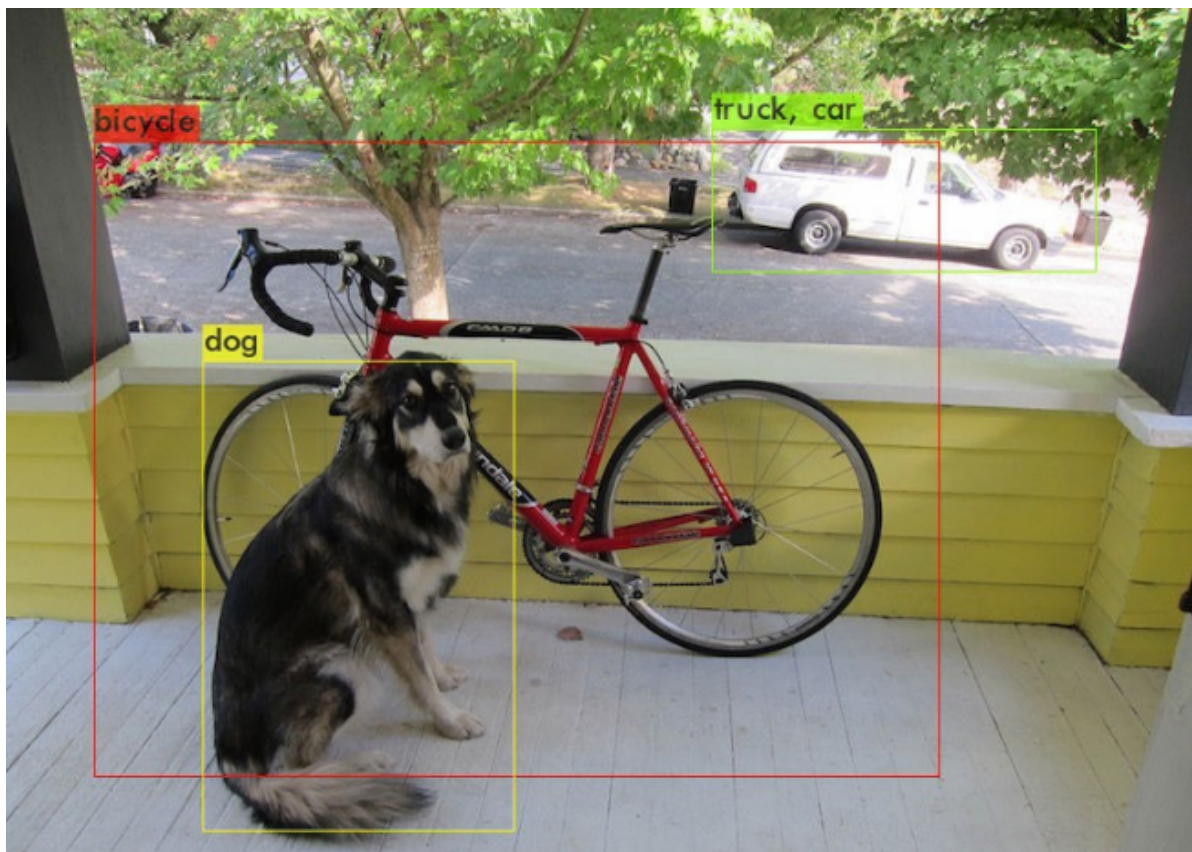
Comparison of YOLOv4 and YOLOv4-Tiny detection results, source network

YOLOv4 detection results



```
Done ! Loaded 162 layers from weights - file
data/dog.jpg : Predicted in 27.039000 milli - seconds.
bicycle : 92 %
dog : 98 %
truck : 92 %
pottedplant : 33 %
```

YOLOv4-Tiny detection results



```
Done ! Loaded 38 layers from weights - file
data/dog.jpg : Predicted in 2.609000 milli - seconds.
bicycle : 29 %
dog : 72 %
truck : 82 %
car : 46 %
```

We can see that the detection accuracy of Yolov4-tiny has decreased, but Yolov4-tiny has obvious advantages in terms of time consumption: Yolov4-tiny detection takes only 2.6 milliseconds, while Yolov4 detection takes 27 milliseconds, which is faster More than 10 times!

• 6.2. Use

Supports Raspberry Pi version.

(Note: The python used by Raspberry Pi 4B is version 3.7, please use python3.7 when running commands!)

jetson motherboard/Raspberry Pi 4B

```
roslaunch yahboomcar_yolov4_tiny yolodetect.launch display:=true
```

Raspberry Pi 5

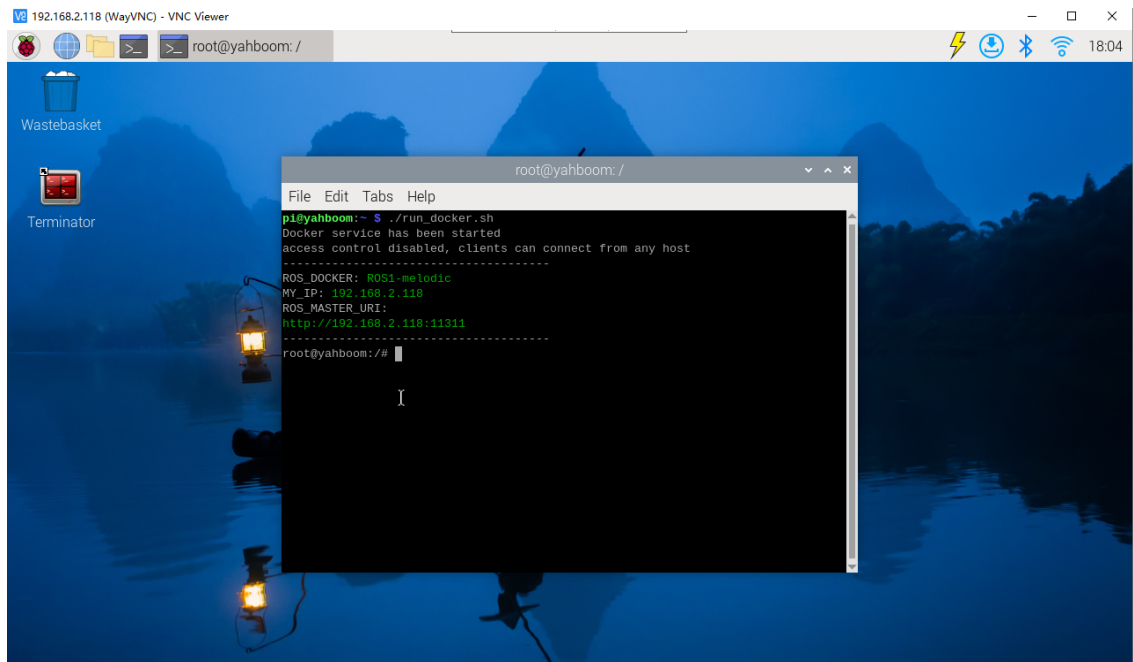
Before running, please confirm that the large program has been permanently closed

Enter docker

Note: If there is a terminal that automatically starts docker, or there is a docker terminal that has been opened, you can directly enter the docker terminal to run the command, and there is no need to manually start docker

Start docker manually

```
./run_docker.sh
```



```
roslaunch yahboomcar_yolov4_tiny yolodetect.launch display:=true
```

- [display] parameter: whether to enable the visual interface.

Supports real-time monitoring of web pages, such as:

Support real-time monitoring of web pages, such as:

```
192.168.2.89:8080
```

View node information

jetson motherboard/Raspberry Pi 4B

```
rqt_graph
```

Raspberry Pi 5

Enter the same docker from multiple terminals

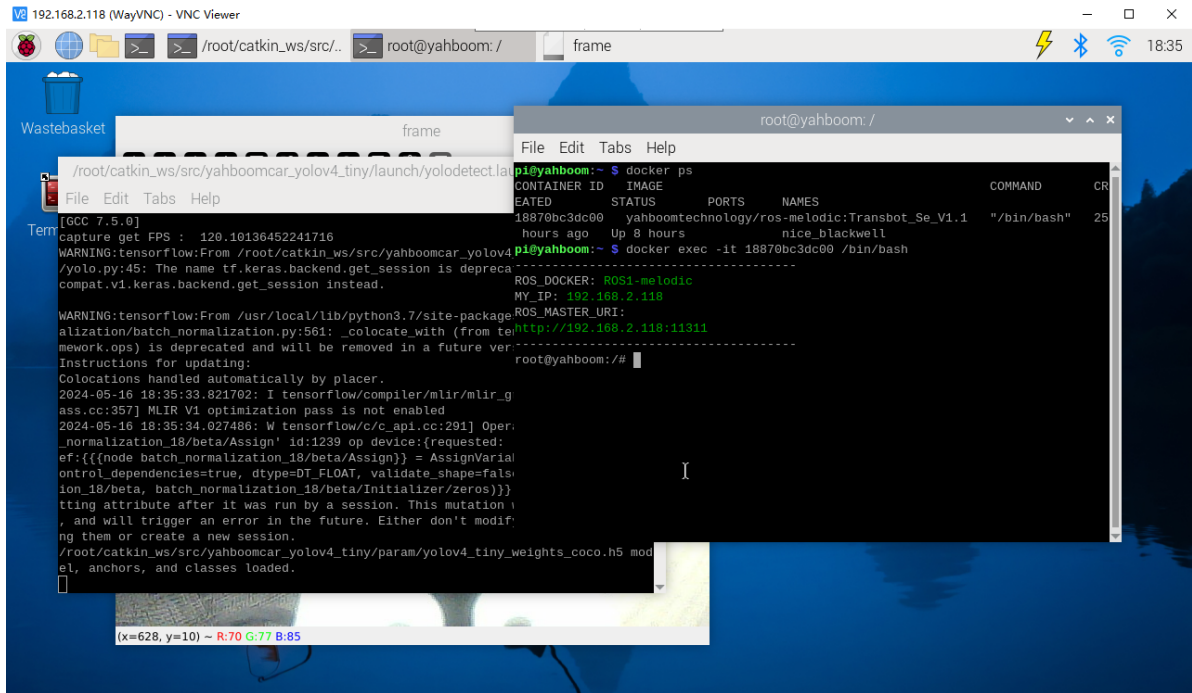
Keep the program of the previous docker terminal running and open a new terminal

Enter the following command

```
docker ps
```

Enter the same docker and use the following 18870bc3dc00 to modify the ID displayed on the actual terminal.

```
docker exec -it 18870bc3dc00 /bin/bash
```

rqt_graph



Print detection information

rostopic echo /DetectMsg

print as follows

```

data :
-
  frame_id : "person"
  stamp :
    secs : 1646128857
    nsecs : 530594825
  scores : 0.609634816647
  ptx : 109.685585022
  pty : - 2.94450759888
  distw : 374.364135742
  disth : 236.672561646
  centerx : 187.182067871
  centery : 118.336280823
  
```

- frame_id: Identifying name.
- scores: Identify scores.
- ptx, pty: the coordinates of the upper left corner of the recognition box.
- distw, disth: The width and height of the recognition box.
- centerx, centery: Identify the center.

6.3 folder structure

```
yolo4 - tiny - tf2
├─ font                                # Store font package
│   └─ Block_Simplified.TTF
├─ garbage_data                        # dataset
│   ├── GetData.py                    # Get the dataset code
│   ├── image                          # target source file
│   ├── JPEGImages                    # dataset images(as many as possible)
│   ├── texture                       # background image(as many as possible)
│   └─ train The .txt                 label file corresponding to the
dataset image
├─ img                                # store the test image
│   └─ 1.j pg
├─ logs                               # Store the test log and the final
training model last1.h5.
├─ model_data                          # Store the pre-trained model(weight
file)
│   ├── coco.txt
│   ├── garbage.h5
│   ├── garbage # .txt                Custom label file(corresponding to
the target source file)
│   ├── yolo_anchors.txt
│   ├── yolo4_tiny_weights_coco.h5    # weights file
│   └─ yolo4_tiny_weights_voc.h5      # weights file
├─ predict_img .py                    # detection Image                code
├─ predict_video .py                  # detection video                code
├─ README.md
├─ train # .py                        training model code
├─ utils                              # library file
└─ yolo_nets                          # Network structure library file
```

The concept of anchor box was introduced in the YOLO-v2 version, which greatly increased the performance of target detection. The essence of anchor is the reverse of the idea of SPP(spatial pyramid pooling), and what SPP itself does is to combine different sizes The input resize becomes the output of the same size, so the inverse of SPP is to push the output of the same size backward to get the input of different size.

6.4 Environmental requirements

The factory image is already configured, no need to install

```
tensorflow-gpu==2.2.0
lxml
matplotlib
pandas
Pillow
scikit-learn
seaborn
tqdm
imgaug
```

Installation example

```
pip install imgaug
```

6.5 Custom training data set

6.5.1 Making a dataset

Method 1: Take some photos first, use the annotation tool to mark the target on each photo, create a [train.txt] file under the [garbage_data] folder, and write the target information in a specific format.

Method 2: Put background images(as many as possible) in the [garbage_data/texture] folder, modify the [GetData.py] code as required, and execute [GetData.py] to generate a dataset(as many as possible).

The name of the image and the label file should correspond. The label format in the [train.txt] file is as follows:

```
./garbage_data/JPEGImages/0.j, pg, 113, 163 293 #, 298, image, 9 label y y + w,  
x + h,
```

Take method 2 as an example.

Open the [GetData.py] file

```
sudo vim GetData.py
```

Modify the total number of generated datasets and fill in as required. [More], too few datasets will lead to suboptimal training results.

```
img_total = 10000
```

Run the [GetData.py] file to get the dataset

```
python3 GetData.py
```

6.5.2 Add weight file

There are good weight files(pre-training model) [yolov4_tiny_weights_coco.h5] and [yolov4_tiny_weights_voc.h5] under the [model_data] file. Choose one of the two, and recommend coco's weight file.

If you need the latest weight file, you can download it by Baidu search.

6.5.3 Make label file

Be careful not to use Chinese labels and no spaces in the folder!

For example: garbage.txt

```
Zip_top_can
Old_school_bag
Newspaper
Book
Toilet_paper
...
```

6.5.4 Modify the train.py file

According to your needs, refer to the notes to modify.

```
#tag position
annotation_path = 'garbage_data/train.txt'
# Get the location of classes and anchors
classes_path = 'model_data/garbage.txt'
anchors_path = 'model_data/yolo_anchors.txt'
# The location of the pretrained model
weights_path = 'model_data/yolov4_tiny_weights_coco.h5'
# get classes and anchors
class_names = get_classes(classes_path)
anchors = get_anchors(anchors_path)
# How many classes are there in total
num_classes = len(class_names)
num_anchors = len(anchors)
# The location where the trained model is saved
log_dir = 'logs/'
# Enter the image size, if the video memory is relatively large, you can use
608x608
input_shape = (416, 416)
# initial eTOCh value
Init_eTOCh = 0
# Freeze training eTOCh values
Freeze_eTOCh = 50
# The size of Batch_size, which indicates how much data is fed each time. If OOM
or insufficient video memory, please reduce it.
batch_size = 16
# max learning rate
learning_rate_base = 1e-3
# total eTOCh value
ETOCh = 100
```

According to the above process, after the operation is completed, you can directly run the [train.py] file for training.

```
python3 train.py
```

6.5.5 Model checking

- Modify the yolov4-tiny-tf2/utils/yolo.py file

```
class YOLO(object):
    _defaults = {
        # For detection, the trained model path.
```



```

    "model_path" : 'model_data/garbage.h5',
    # yolo's model parameter anchors path
    "anchors_path" : 'model_data/yolo_anchors.txt',
    # Custom label file path
    "classes_path" : 'model_data/garbage.txt',
    "score" : 0.5,
    "iou" : 0.3,
    "eager" : False,
    # Use 416x416 by default(image size)
    "model_image_size" : (416, 416)
}
...
# font package path
self.font_path = 'font/Block_Simplified.TTF'

```

- Image detection
- **jetson motherboard/Raspberry Pi 4B**

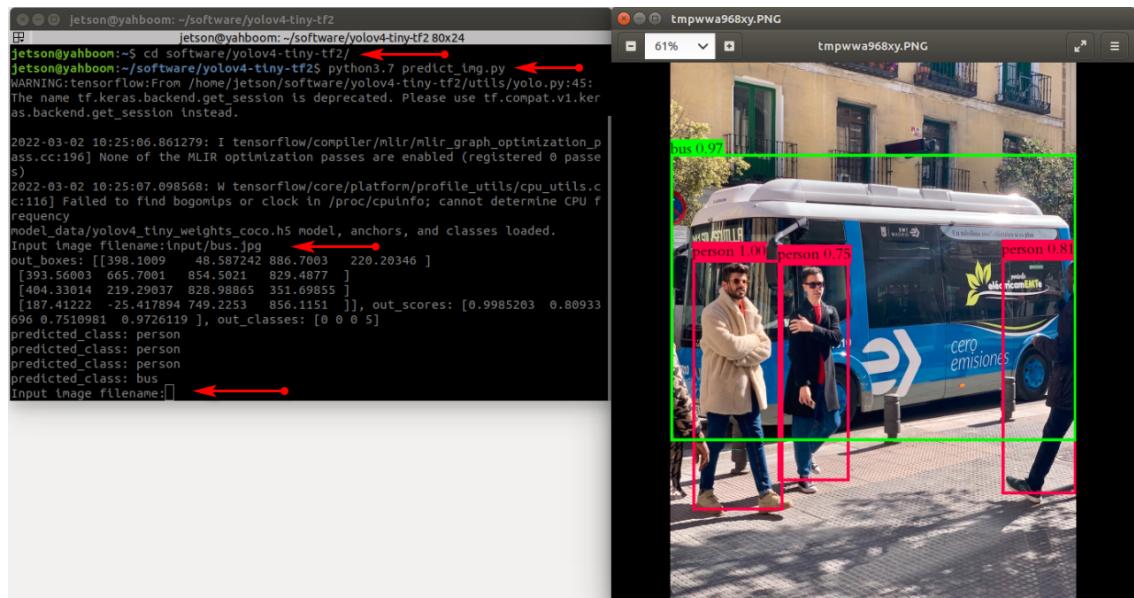
```
python3 predict_img.py
```

Raspberry Pi 5

Just open the terminal of Raspberry Pi 5 and run it directly without entering docker.

```
cd software/yolov4-tiny-tf2
python3 predict_img.py
```

During this period, you need to manually enter the images that need to be detected, as shown below:



Video detection

```
python3 predict_video.py
```

