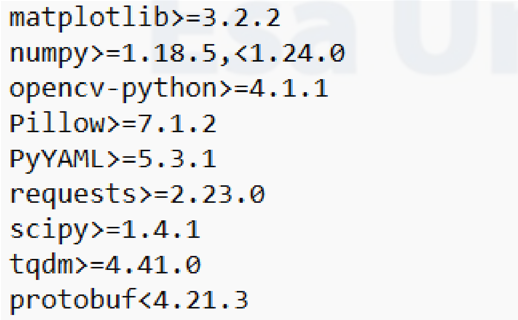
Lampiran

1.1 Lampiran Source Code

1. Requirements



```
matplotlib>=3.2.2
numpy>=1.18.5,<1.24.0
opencv-python>=4.1.1
Pillow>=7.1.2
PyYAML>=5.3.1
requests>=2.23.0
scipy>=1.4.1
tqdm>=4.41.0
protobuf<4.21.3
```

2. Training Data

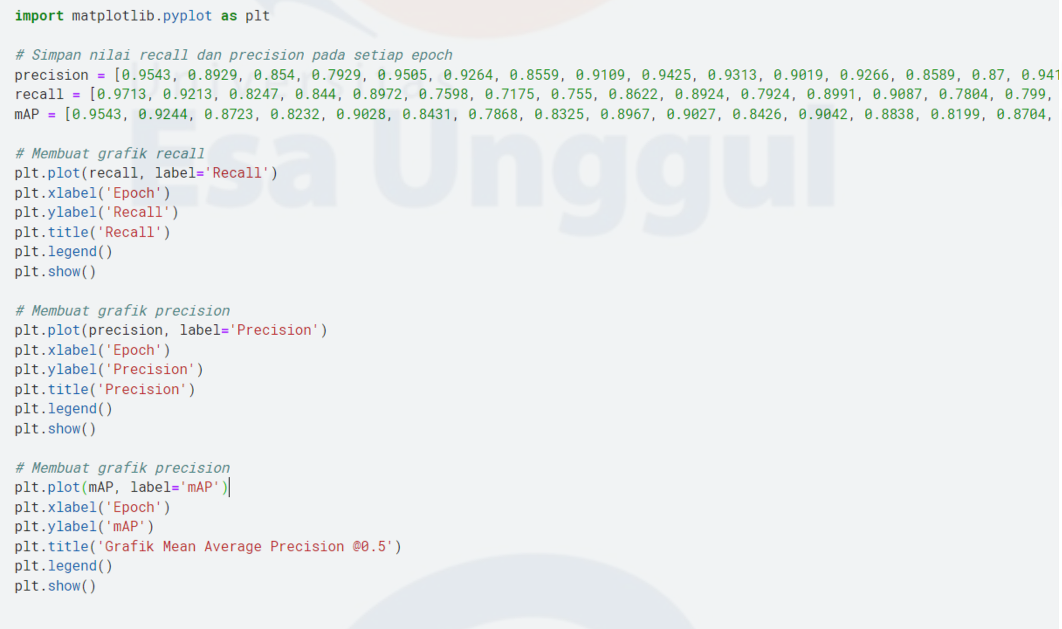
```
!WANDB_MODE="dryrun" python train.py --workers 1 --device 0 --batch-size 8 --epochs 100 --img 640 640 --hyp data/hyp.scratch.custom.yaml --name yolov7-custom --weights last.pt --resume
```

3. Detect Object

```
import cv2

import telepot

import torch

import time

import torch.backends.cudnn as cudnn

from preprocessing import draw, normalize, resizeAndPad

from models.experimental import attempt_load

from utils.general import non_max_suppression

# Bot Iniziation

class_to_monitor = "fall"

display = False

token = '5814517375:AAHctAaHWAyi5ihxQ98eokFelV3MBVygCEs'

receiver_id = 1391371644

bot = telepot.Bot(token)

bot.sendMessage(receiver_id, 'Your Camera is active now')
```

```
weights = f"weights/yolov7_fall.pt"
print(f"Loading model {weights}")
device = torch.device('cuda')
model = attempt_load(weights, map_location=device)
stride = int(model.stride.max())
cudnn.benchmark = True
print(f"Model {weights} loaded")

# Initiate Video Streaming with OpenCV
cap = cv2.VideoCapture(1)  # Change the parameter to your video file or device index
W,H = (640,640)  # Specify the desired width and height

while True:
    time.sleep(0.2)  # wait for 0.2 second
    ret, frame = cap.read() # read frame

    resized_frame = resizeAndPad(frame, (W,H))

    # preprocess resized_frame
    img = normalize(resized_frame, device)

    # prediction
    pred = model(img, augment=False)[0]
    pred = non_max_suppression(prediction=pred, conf_thres=0.5, iou_thres=0.5, agnostic=True)
    pred = pred[0]
    pred = pred.cpu().data.numpy().tolist()
```

```
# if there's detection, send notif to bot
    if len(pred) > 0:
        # draw bb, class, score
        resized_frame = draw(resized_frame, pred)
        time_stamp = int(time.time())
        fcm_photo = f'C:/Users/Fajar Maulana/Documents/yolov7-custom/torch
api/detected/{time_stamp}.png'
        cv2.imwrite(fcm_photo, resized_frame)  # notification photo
        last_data = pred[-1]  # Mengakses data terakhir dalam array utama
        inner_data = last_data[5]
        print(inner_data)
        if inner_data == 1.0 :
            bot.sendPhoto(receiver_id, photo=open(fcm_photo, 'rb'), caption='HELLO!!!
SOMEONE HAS FALLING, HELP IMMEDIATELY!')
            print(f'{time_stamp}.png has been sent.')


        time.sleep(1)  # wait for 1 second. Only when it detects.


    cv2.imshow('Resized Frame', resized_frame)



    print(type(resized_frame))
    print(resized_frame.shape)


    if cv2.waitKey(1) & 0xFF == ord('q'):
        break


cap.release()
cv2.destroyAllWindows()
```



```python
import matplotlib.pyplot as plt

# Simpan nilai recall dan precision pada setiap epoch
precision = [0.9543, 0.8929, 0.854, 0.7929, 0.9505, 0.9264, 0.8559, 0.9109, 0.9425, 0.9313, 0.9019, 0.9266, 0.8589, 0.87, 0.941
recall = [0.9713, 0.9213, 0.8247, 0.844, 0.8972, 0.7598, 0.7175, 0.755, 0.8622, 0.8924, 0.7924, 0.8991, 0.9087, 0.7804, 0.799,
mAP = [0.9543, 0.9244, 0.8723, 0.8232, 0.9028, 0.8431, 0.7868, 0.8325, 0.8967, 0.9027, 0.8426, 0.9042, 0.8838, 0.8199, 0.8704,

# Membuat grafik recall
plt.plot(recall, label='Recall')
plt.xlabel('Epoch')
plt.ylabel('Recall')
plt.title('Recall')
plt.legend()
plt.show()

# Membuat grafik precision
plt.plot(precision, label='Precision')
plt.xlabel('Epoch')
plt.ylabel('Precision')
plt.title('Precision')
plt.legend()
plt.show()

# Membuat grafik precision
plt.plot(mAP, label='mAP')
plt.xlabel('Epoch')
plt.ylabel('mAP')
plt.title('Grafik Mean Average Precision @0.5')
plt.legend()
plt.show()
```