LAMPIRAN

```
!pip install ultralytics

from ultralytics import YOLO

models = [YOLO("yolov8n.pt"),
          YOLO("yolov9t.pt"),
          YOLO("yolov10n.pt"),
          YOLO("yolo11n.pt")]

results = dict()

for model in models:
  metrics = model.val(data='coco128.yaml', imgsz=640,
epochs=100, save=True, plots=True, verbose=False)

  model_name = model.model_name.split('.')[0]
  result     = metrics.results_dict
  precision, recall, map50, map5095, fitness = result.values()

  results[model_name] = dict()
  results[model_name]["precision"]  = float(round(precision, 3))
  results[model_name]["recall"]     = float(round(recall, 3))
  results[model_name]["map50"]      = float(round(map50, 3))
  results[model_name]["map5095"]    = float(round(map5095, 3))
  results[model_name]["fitness"]    = float(round(fitness, 3))

for model in results:
  print(model)

  for metric in results[model]:
    print(f"{metric}: {results[model][metric]}")
  print("=" * 20)
```

```
!pip install ultralytics
!pip install roboflow

from ultralytics import YOLO
```

```
from roboflow import Roboflow

models = [YOLO("yolov8n.pt"),
          YOLO("yolov9t.pt"),
          YOLO("yolov10n.pt"),
          YOLO("yolo11n.pt")]

rf = Roboflow(api_key="GSwlztLrfT4RvpTyPmjx")
project = rf.workspace("myproject-cc5hp").project("people-
tracking-and-counting")
version = project.version(3)
dataset = version.download("yolov11")

dataset_path = 'People Tracking and Counting'

results = dict()

for model in models:
  metrics = model.val(data='coco128.yaml', imgsz=640,
epochs=100, save=True, plots=True, verbose=False)

  model_name = model.model_name.split('.')[0]
  result     = metrics.results_dict
  precision, recall, map50, map5095, fitness = result.values()

  results[model_name] = dict()
  results[model_name]["precision"]  = float(round(precision, 3))
  results[model_name]["recall"]     = float(round(recall, 3))
  results[model_name]["map50"]      = float(round(map50, 3))
  results[model_name]["map5095"]    = float(round(map5095, 3))
  results[model_name]["fitness"]    = float(round(fitness, 3))

for model in results:
  print(model)

  for metric in results[model]:
    print(f"{metric}: {results[model][metric]}")

  print("=" * 20)
```

```
!pip install ultralytics
!pip install roboflow

import pandas as pd
import numpy as np
import os
import ultralytics as ut

from roboflow import Roboflow
from ultralytics import YOLO

rf = Roboflow(api_key="GSwlztLrfT4RvpTyPmjx")
project = rf.workspace("myproject-cc5hp").project("people-
tracking-and-counting")
version = project.version(3)
dataset = version.download("yolov11")

rf_dataset_yaml = "/content/People-Tracking-and-Counting-
3/data.yaml"

models = [YOLO("yolo11n.pt"),
          YOLO("yolo11s.pt"),
          YOLO("yolo11m.pt"),
          YOLO("yolo11l.pt"),
          YOLO("yolo11x.pt")]

output_dir = "/content/trained_results"

os.makedirs(output_dir, exist_ok=True)

for model in models:
  model.train(data=rf_dataset_yaml, epochs=100, imgsz=640,
batch=16, device='cuda', workers=12, project=output_dir,
name=model.model_name, plots=True)
```

```
import pandas as pd
import numpy as np
import os
```

```
import time

from ultralytics import YOLO

DATASET_DIR       = "/content/drive/MyDrive/Colab
Notebooks/[Skripsi] People Tracking with YOLO and Deep
SORT/dataset"
MOT_VER           = "MOT20"
MOT_DIR           = os.path.join(DATASET_DIR, MOT_VER)
TRAIN_DIR         = os.path.join(MOT_DIR, 'train')
TRAIN_SEQUENCES   = sorted(os.listdir(TRAIN_DIR))
TRAIN_SEQ_DIRS    = [os.path.join(TRAIN_DIR, seq) for seq in
TRAIN_SEQUENCES]
TRAIN_IMG_DIRS    = [os.path.join(train_seq_dir, "img1") for
train_seq_dir in TRAIN_SEQ_DIRS]
IMG_EACH_SEQ      = [os.listdir(train_img_dir) for train_img_dir
in TRAIN_IMG_DIRS]

IMGS_SEQUENCES   = dict()
for i in range(len(TRAIN_IMG_DIRS)):
  IMGS_SEQUENCES[TRAIN_SEQUENCES[i]] = list()
  for img in IMG_EACH_SEQ[i]:
    img_path = os.path.join(TRAIN_IMG_DIRS[i], img)
    IMGS_SEQUENCES[TRAIN_SEQUENCES[i]].append(img_path)

def extract_results(source, model, tracker, mot_ver, seq_names):
  results_dict = dict()
  time_dict = dict()

  tracker_name = tracker.split(".")[0]

  start = time.time()

  for i, seq_dir in enumerate(source):
      mot_results = []

      for j, img in enumerate(os.listdir(seq_dir)):
          frame = j + 1
```

```
          img_path = os.path.join(seq_dir, img)
          results = model.track(img_path, tracker=tracker,
conf=0.20, save=False, show_labels=False, show_conf=False,
show_boxes=False, verbose=False)

          id = results[0].boxes.id
          conf = results[0].boxes.conf

          xyxy = results[0].boxes.xyxy
          x1, y1 = xyxy[:, 0], xyxy[:, 1]

          xywh = results[0].boxes.xywh
          w, h = xywh[:, 2], xywh[:, 3]

          if id is not None:
            for k in range(len(id)):
              mot_results.append([frame, id[k], x1[k], y1[k],
w[k], h[k], conf[k], -1, -1, -1])

      results_dict[seq_names[i]] = mot_results

  end = time.time()

  processing_speed = end - start
  time_dict[tracker_name] = processing_speed

  print(f"Processing Time {mot_ver}-{tracker_name}:
{processing_speed} seconds")

  return results_dict

def convert_to_df(results):
  mots = dict()

  for i in results:
    df = pd.DataFrame(results[i]).astype(np.float64)
    mots[i] = df

  return mots
```

```
def save_results(mots, tracker, split, model, mot_ver):
  print(f"Saving Results")

  model_name = model.model_name.split(".")[0]
  tracker_name = tracker.split(".")[0]

  mot_dir = mot_ver
  model_directory = os.path.join(mot_dir, model_name)
  tracker_directory = os.path.join(model_directory,
tracker_name)
  output_directory = os.path.join("/content", tracker_directory,
split)

  os.makedirs(output_directory, exist_ok=True)

  for mot in mots:
    mots[mot].to_csv(f"{output_directory}/{mot}.csv",
index=False, header=False)
    print(f"Result saved to: {output_directory}/{mot}.csv")
    print(f"=" * 50)

def extract_convdf_save(source, model, tracker, seq_names,
mot_ver, split):
  results_dict = extract_results(source, model, tracker,
mot_ver, seq_names)
  mots = convert_to_df(results_dict)
  save_results(mots, tracker, split, model, mot_ver)

model_path  = "/content/drive/MyDrive/Colab Notebooks/[Skripsi]
People Tracking with YOLO and Deep
SORT/models/yolo11n/weights/best.pt"
model       = YOLO(model_path)
model.fuse()

trackers  = ["botsort.yaml", "bytetrack.yaml"]
sources   = TRAIN_IMG_DIRS
seq_names = TRAIN_SEQUENCES
```

```
for tracker in trackers:
  extract_convdf_save(sources, model, tracker, seq_names,
"MOT20", "train")
```

```
import ultralytics as ut
import math
import time
import cv2
import os

from ultralytics import solutions

footages_dir =
"path_menuju_direktori_penyimpanan_seluruh_footage"
footage_1 = dict(path = "path_menuju_footage_1", region = [(211,
208), (0, 398), (0, 720), (1280, 720), (1280, 398), (1069,
208)])
footage_2 = dict(path = "path_menuju_footage_2", region = [(211,
208), (0, 398), (0, 720), (1280, 720), (1280, 398), (1069,
208)])
footage_3 = dict(path = "path_menuju_footage_3", region = [(0,
664), (1280, 170), (1280, 442), (835, 720), (0, 720)])
footage_4 = dict(path = "path_menuju_footage_4", region = [(73,
256), (558, 152), (1280, 400), (1234, 720), (230, 718)])
footage_5 = dict(path = "path_menuju_footage_5", region = [(634,
473), (909, 265), (1085, 276), (971, 573), (991, 720), (802,
720)])

def run_tracking(model_path, tracker_path, footage, region,
conf_score, iou_score, saving=True):
  cap = cv2.VideoCapture(footage)
  assert cap.isOpened(), "Error reading video file"

  w, h, fps = (int(cap.get(x)) for x in
(cv2.CAP_PROP_FRAME_WIDTH, cv2.CAP_PROP_FRAME_HEIGHT,
cv2.CAP_PROP_FPS))

  # Membuat Directory Khusus Hasil Tracking
  output_dir = "/content/drive/MyDrive/Track Results"
```

```
  os.makedirs(output_dir, exist_ok=True)

  print(f"Now processing {footage.split('/')[-
1].split('.')[0]}")
  print(f"Tracker used is '{tracker_path.split('/')[-
1].split('.')[0]}'")

  output_filename = f"trackzone_{footage.split('/')[-
1].split('.')[0]}_{tracker_path.split('/')[-
1].split('.')[0]}.avi"
  output_path     = os.path.join(output_dir, output_filename)

  # Video writer
  video_writer  = cv2.VideoWriter(output_path,
cv2.VideoWriter_fourcc(*"mp4v"), fps, (w, h))
  region_points = region  # For rectangle region tracking

  # Menginisiasi Sistem Pelacak Track Zone
  trackzone = solutions.TrackZone(region=region_points,
model=model_path, tracker=tracker_path, conf=conf_score,
iou=iou_score, device=0, show=True)

  # Memproses Video
  total_ids = set()
  while cap.isOpened():
      success, im0 = cap.read()
      if not success:
          print("Frame tidak terdeteksi atau sudah selesai
memproses video!")
          break

      start_time = time.time()
      results    = trackzone(im0) # Melakukan Pelacakan
      end_time   = time.time()

      # Menghitung FPS
      fps = 1 / (end_time - start_time)

      plot_im = results.plot_im
```

```
      for id in trackzone.track_ids:
        total_ids.add(id)

      # Mendrawing skor FPS dan Total Pengunjung
      fps_xpos, fps_ypos = math.ceil(w - (w * 15 / 100)),
math.ceil((h - (h * 95 / 100)))
      visitor_xpos, visitor_ypos = math.ceil(w - (w * 99 /
100)), math.ceil((h - (h * 95 / 100)))

      cv2.putText(plot_im, f"FPS: {fps:.2f}", (fps_xpos,
fps_ypos), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
      cv2.putText(plot_im, f"Total pengunjung:
{len(total_ids)}", (visitor_xpos, visitor_ypos),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

      # cv2_imshow(plot_im)
      video_writer.write(plot_im)   # membuat video dari frame

  cap.release()
  video_writer.release()
  cv2.destroyAllWindows()

model_path   = "/content/drive/MyDrive/Colab Notebooks/[Skripsi]
People Tracking with YOLO and Deep
SORT/models/yolo11n/weights/v5_no_gray_ht1.pt"
footages = [footage_1, footage_2, footage_3, footage_4,
footage_5]

for footage in footages:
  run_tracking(model_path=model_path,
                   tracker_path="path_menuju_file_tracker",
                   footage=footage['path'],
                   region=footage['region'],
                   conf_score=0.27,
                   iou_score=0.50,
                   saving=True)
```

Untuk kode-kode selengkapnya dapat dilihat pada *link* GitHub berikut

GitHub: *https://github.com/dthief55/skripsi*