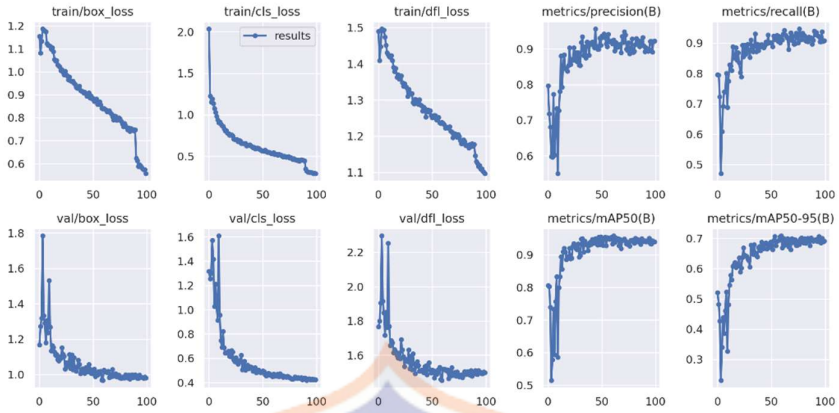
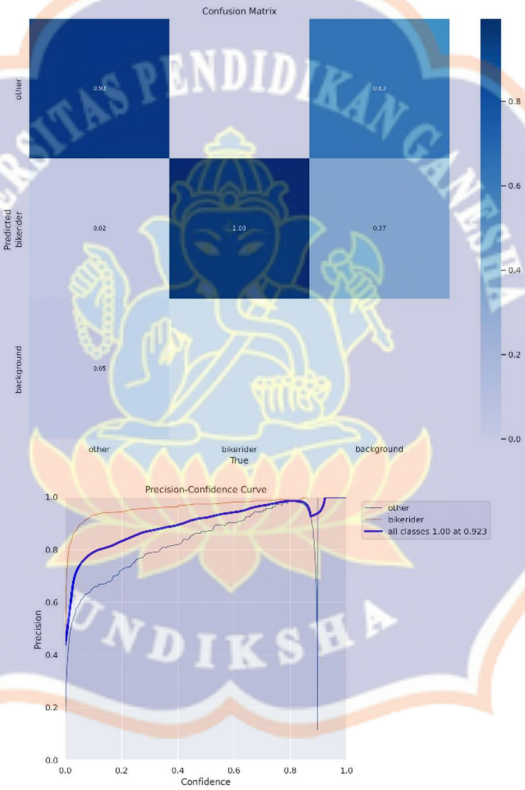
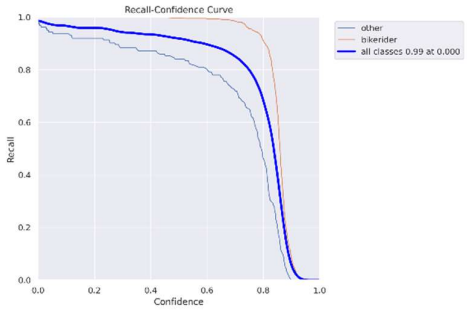
LAMPIRAN

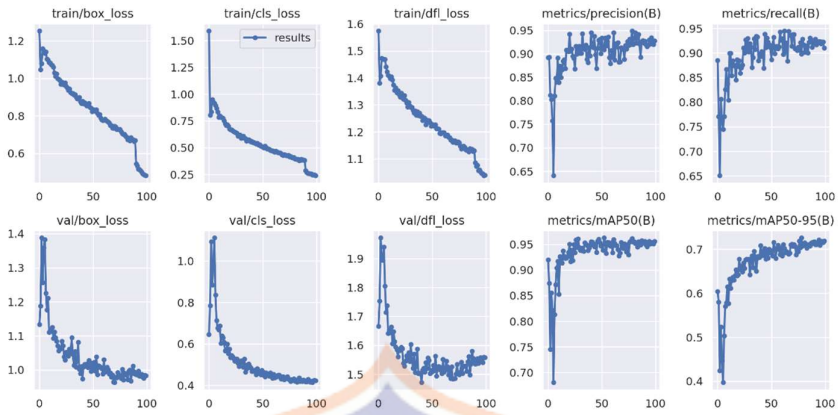
Lampiran 1. Hasil Pelatihan Model *Bikerider* Versi n

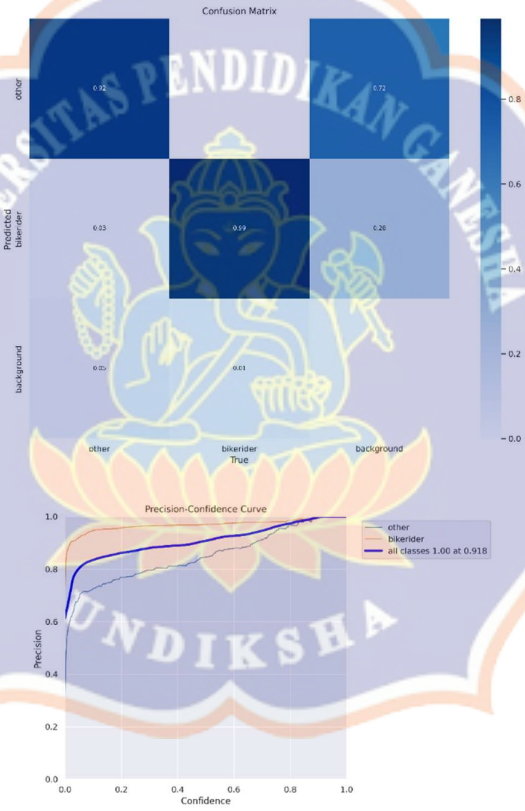


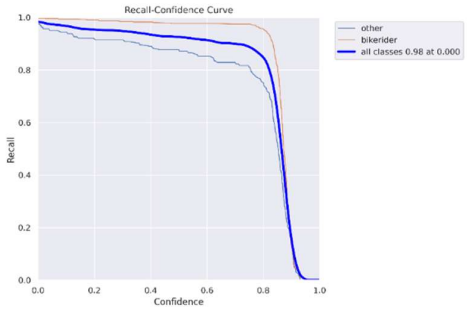




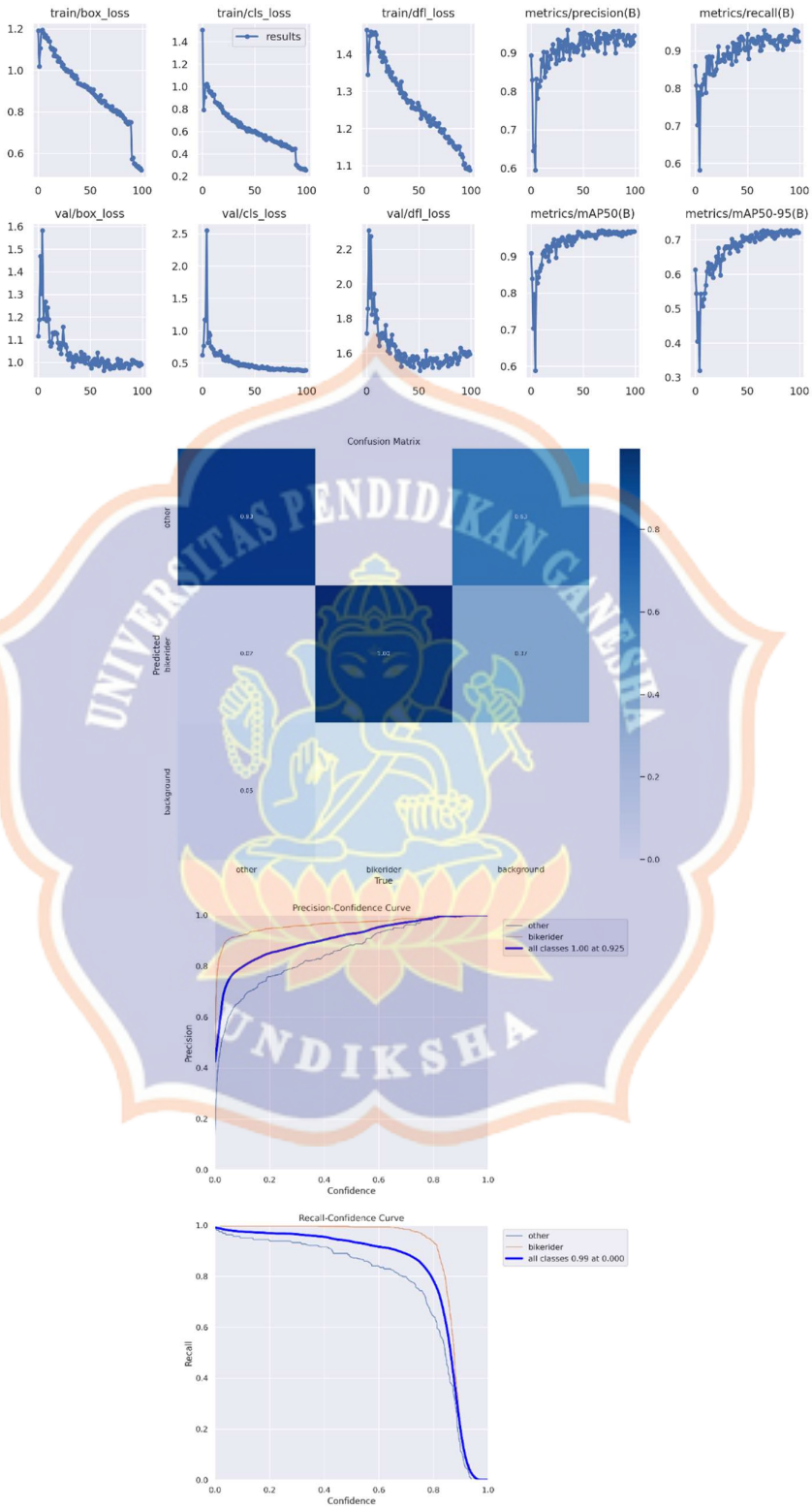
Lampiran 2. Hasil Pelatihan Model *Bikerider* Versi s



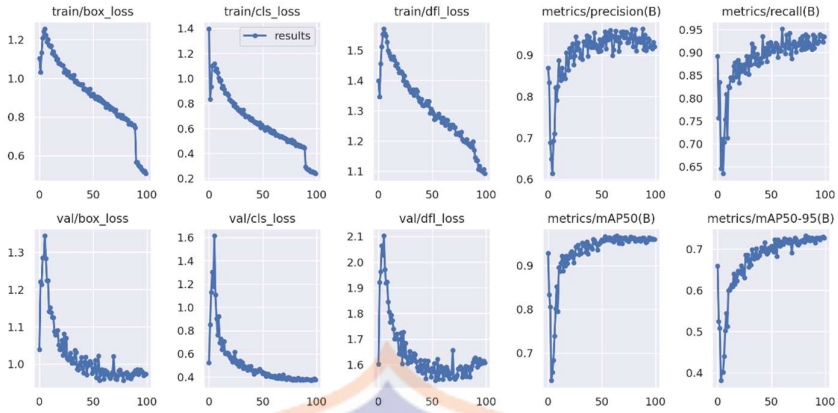


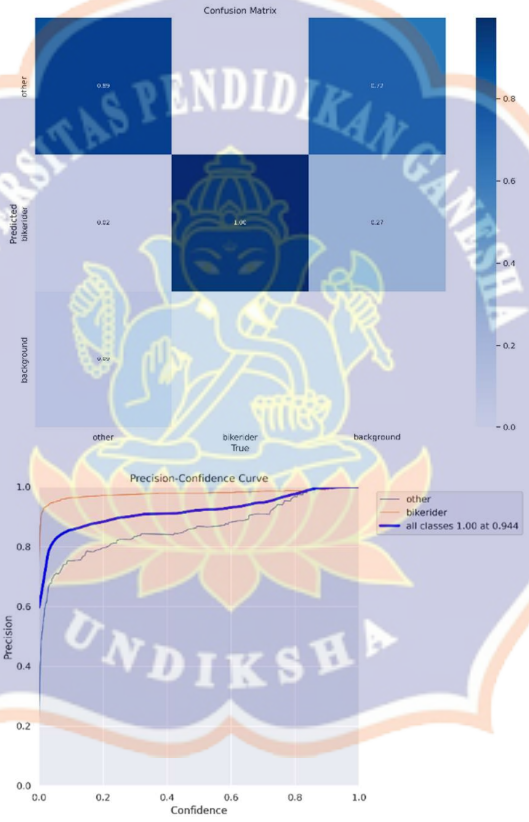


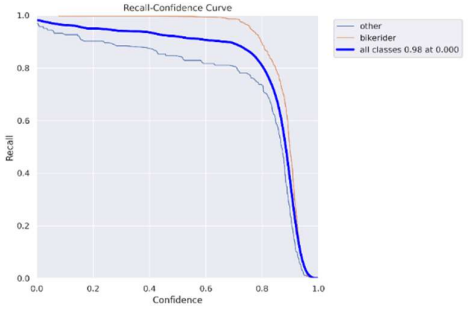
Lampiran 3. Hasil Pelatihan Model *Bikerider* Versi m



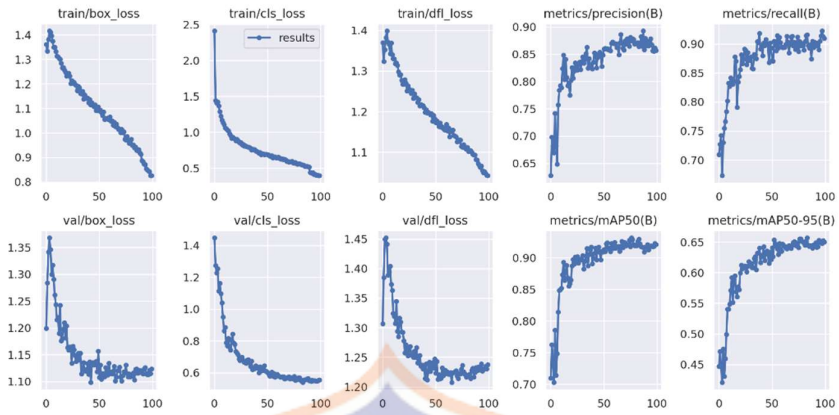
Lampiran 4. Hasil Pelatihan Model *Bikerider* Versi l

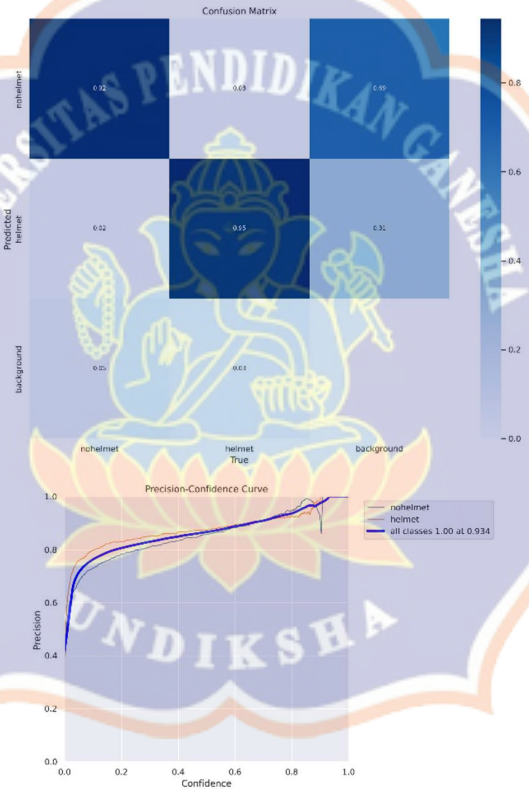


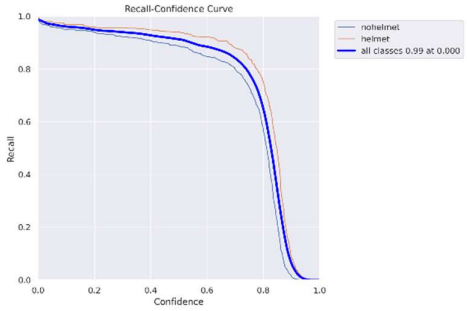




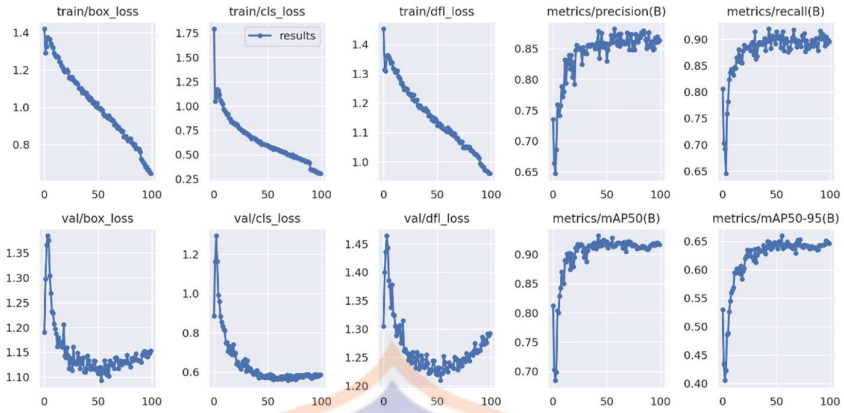
Lampiran 5. Hasil Pelatihan Model *Helmet* Versi n

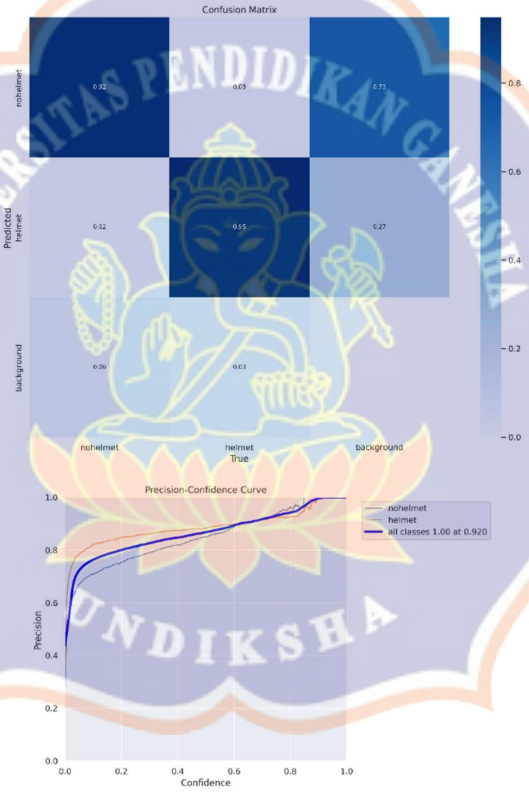


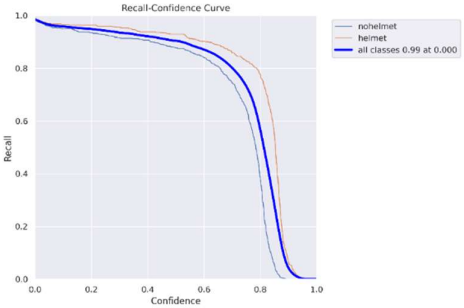




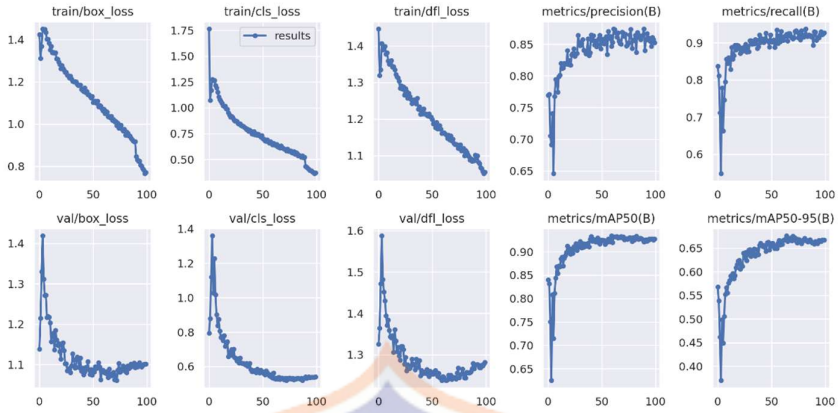
Lampiran 6. Hasil Pelatihan Model *Helmet* Versi s

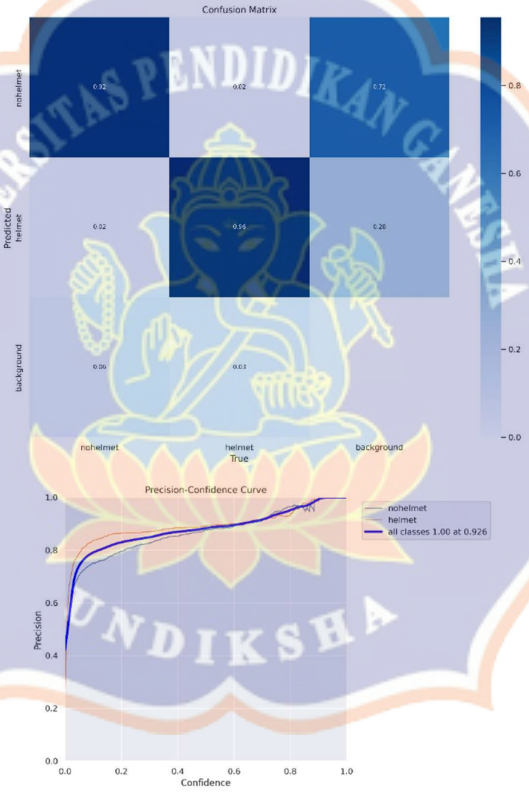


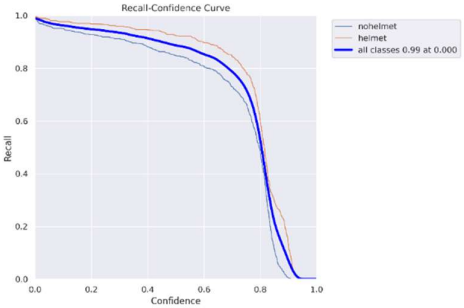




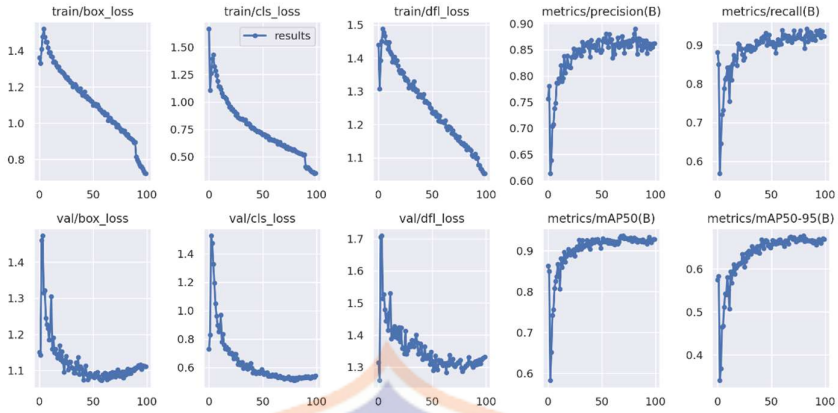
Lampiran 7. Hasil Pelatihan Model *Helmet* Versi m

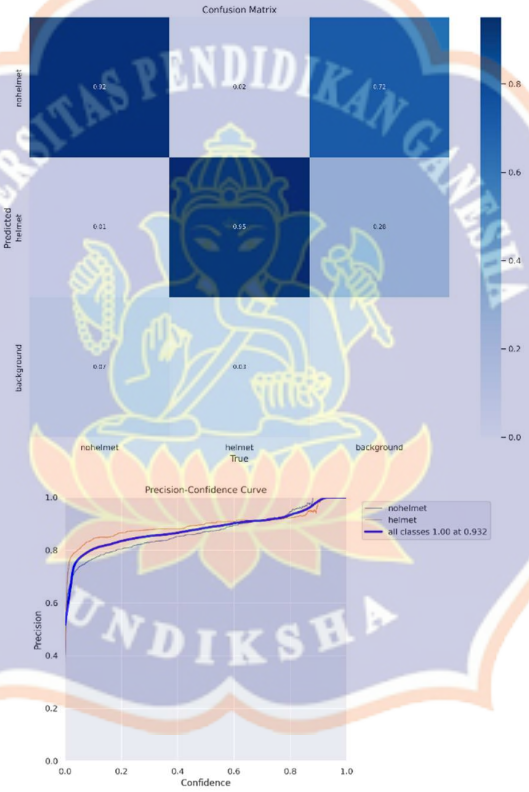


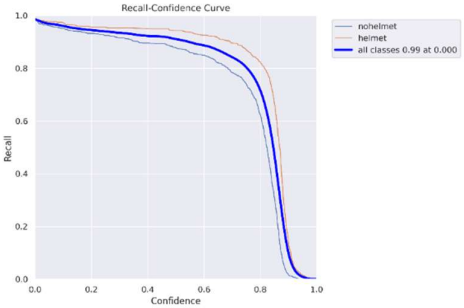




Lampiran 8. Hasil Pelatihan Model *Helmet* Versi l







Lampiran 9. Fungsi Utama pada Sistem

```
import os
import threading
from collections import defaultdict
from datetime import datetime
import subprocess

import cv2
import pandas as pd
import socketio
from flask import render_template, Response, request,
redirect, url_for
from shapely import Point, LineString
from ultralytics import YOLO
from ultralytics.utils.plotting import Annotator,
colors

from __init__ import socketio
from utils.detect_camera import detect_cameras
from . import stream

available_cameras = detect_cameras(4)
camera_running = False
camera_thread = None

def generate_frames(camera_index):
    folder_name = datetime.now().strftime('%Y_%m_%d')
    path = f"static/img/stream/{folder_name}"
    if not os.path.exists(path):
        os.makedirs(path)
    model = YOLO("app/models/bikerider_yolox.pt")
    model2 = YOLO("app/models/helmet_yolox.pt")
```

```
    cap = cv2.VideoCapture(camera_index)
    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))


    # Define region points
     line_points = [(w / 2, h), (w / 2, 0)]


      if line_type == "d1":
          line_points = [(0, h), (w, 0)]
      elif line_type == "d2":
          line_points = [(0, 0), (w, h)]
      elif line_type == "v1":
          line_points = [(w / 2, 0), (w / 2, h)]
      elif line_type == "v2":
          line_points = [(w / 3, 0), (w / 3, h)]
      elif line_type == "v3":
          line_points = [(2 * w / 3, 0), (2 * w / 3,
h)]
      elif line_type == "h1":
          line_points = [(0, h / 2), (w, h / 2)]
      elif line_type == "h2":
          line_points = [(0, h / 3), (w, h / 3)]
      elif line_type == "h3":
          line_points = [(0, 2 * h / 3), (w, 2 * h /
3)]
       track_history = defaultdict(list)
       draw_tracks = True
       track_color = None
       count_ids = []
       counting_region = LineString(line_points)
```

```
total_counts = 0

helmet_counts = 0

no_helmet_counts = 0


while camera_running:

    success, frame = cap.read()

    if not success:

        break


    tracks = model.track(frame, persist=True,
    show=False)


    annotator = Annotator(frame, 1, model.names)


    # Draw Garis

    annotator.draw_region(reg_pts=line_points,
    color=(255, 0, 255), thickness=1)


    if tracks[0].boxes.id is not None:

       boxes = tracks[0].boxes.xyxy.cpu()

       clss = tracks[0].boxes.cls.cpu().tolist()

       track_ids =
       tracks[0].boxes.id.int().cpu().tolist()


    # Extract tracks

    for box, track_id, cls in zip(boxes, track_ids,
    clss):


    # Visualisasi Box Bikerider model

    annotator.box_label(box,
    label=f"{model.names[cls]}#{track_id}",
    color=colors(int(track_id), True))
```

```
# Draw Tracks
track_line = track_history[track_id]
track_line.append((float((box[0] + box[2]) /
2), float((box[1] + box[3]) / 2)))
if len(track_line) > 30:
   track_line.pop(0)


# Visualisasi Tracking Bikerider
if draw_tracks:
    annotator.draw_centroid_and_tracks(
    track_line, color=track_color if
    track_color else colors(int(track_id),
    True), track_thickness=1,)


prev_position = track_history[track_id][-2]
if len(track_history[track_id]) > 1 else None
if prev_position is not None and track_id not
in count_ids:
   distance = Point(track_line[-
   1]).distance(counting_region)
   if distance < 15 and track_id not in
   count_ids:
   count_ids.append(track_id)
   if model.names[cls] == "bikerider":
   crop_obj = frame[int(box[1]): int(box[3]),
   int(box[0]): int(box[2])]
    results = model2.predict(crop_obj)
   helm_boxes = results[0].boxes.cpu().data
   helm_coors =
   pd.DataFrame(helm_boxes).astype("float")
   is_there_no_helmet = False
   # menghitung jumlah helmet dan no-helmet
   for _, row in helm_coors.iterrows():
       helm_cls = model2.names[row[5]]
```

```
                if 'helmet' in helm_cls:
                annotator.box_label(
                [row[0] + box[0], row[1] + box[1],
                row[2] + box[0], row[3] + box[1]],
                label=f"Helmet", color=(255, 0, 0)
                helmet_counts += 1

                if 'nohelmet' in helm_cls:
                annotator.box_label(
                [row[0] + box[0], row[1] + box[1],
                row[2] + box[0], row[3] + box[1]],
                label=f"No Helmet", color=(0, 255, 0))
                is_there_no_helmet = True
                no_helmet_counts += 1

        if is_there_no_helmet:
            file_name =
            datetime.now().strftime('%H_%M_%S')
            image_path = os.path.join(path,
            f"{file_name}.png")
            cv2.imwrite(image_path, crop_obj)

            total_counts += 1

    socketio.emit('helmet_count_stream', {'count':
    helmet_counts})
    socketio.emit('no_helmet_count_stream', {'count':
    no_helmet_counts})

    _, jpeg = cv2.imencode('.jpg', frame)
    frame = jpeg.tobytes()
    yield (b'--frame\r\n'
           b'Content-Type: image/jpeg\r\n\r\n' + frame
           + b'\r\n')

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