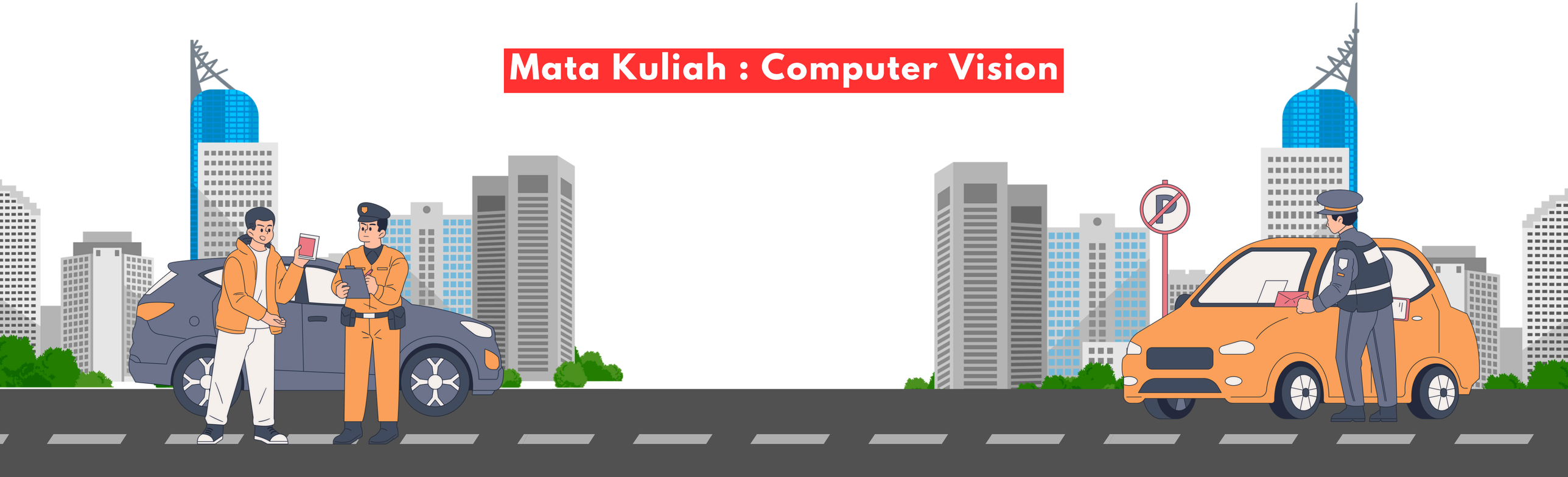
Sistem Deteksi Jenis Mobil dan Estimasi Kecepatan Berbasis Computer Vision untuk Analisis Potensi Kecelakaan Kendaraan

Mata Kuliah : Computer Vision



Anggota Kelompok

RAMDHAN JAVAS DINTARISANTO
23090620048

ARGA ABITAMA
23090620068

M. NABIL KHAIRI IKHSAN
23090620080

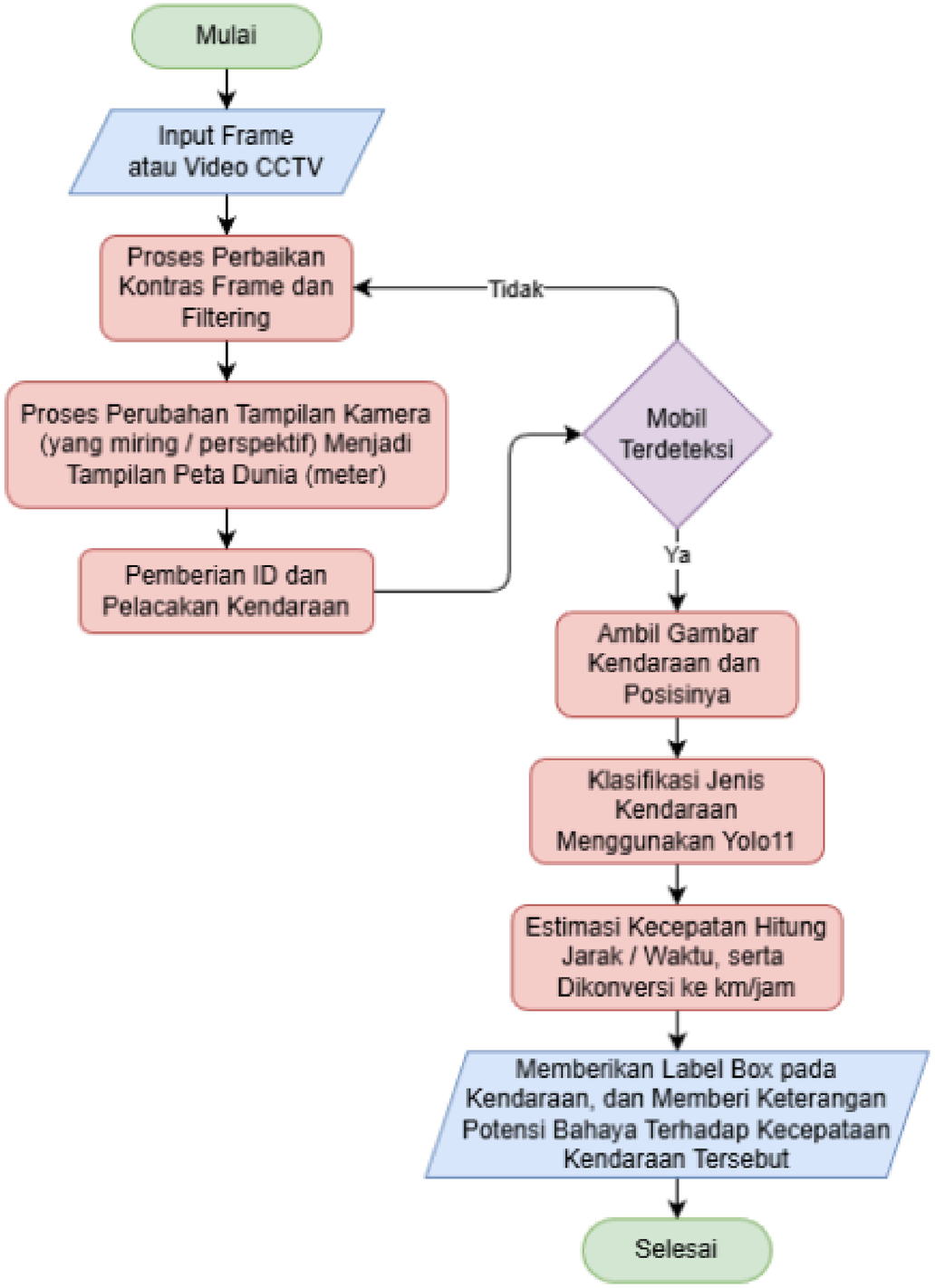
Latar Belakang

Meningkatnya jumlah kendaraan di jalan raya berpotensi menimbulkan angka kecelakaan yang lebih tinggi. Banyak kecelakaan terjadi akibat kecepatan kendaraan yang berlebihan dan kurangnya sistem pemantauan yang mampu mengidentifikasi kondisi berbahaya secara cepat. Dengan kemajuan teknologi kecerdasan buatan dan computer vision, proses deteksi kendaraan, pengukuran kecepatan, serta analisis potensi kecelakaan kini dapat dilakukan secara otomatis melalui kamera. Oleh karena itu, diperlukan sebuah sistem yang mampu mendeteksi tipe mobil, mengestimasi kecepatan kendaraan, serta memberikan peringatan dini potensi kecelakaan untuk meningkatkan keselamatan dan efektivitas monitoring lalu lintas.

Tujuan

- Mengembangkan sistem berbasis computer vision untuk mendeteksi tipe mobil secara otomatis dari citra atau video lalu lintas.
- Mengimplementasikan metode estimasi untuk mengukur kecepatan kendaraan (km/jam) secara real-time.
- Menganalisis dan memprediksi potensi kecelakaan kendaraan berdasarkan pergerakan dan jarak antar kendaraan.
- Menyediakan sistem monitoring lalu lintas yang dapat memberikan peringatan dini terhadap kondisi berbahaya untuk membantu meningkatkan keselamatan di jalan raya.

FlowChart Cara Kerja



Arsitektur Diagram

Mencari Video

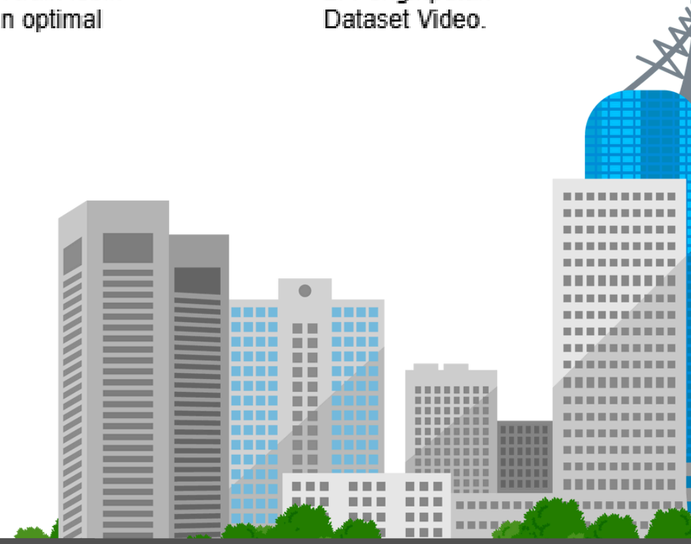
Membuat Video Menjadi Beberapa Gambar

Membagi 1 Detik Video Menjadi 15 Gambar (30fps/2fps)

1. Membuat 4 Class (Mobil, Motor, Truck, Bis).
2. Menandai semua class di semua dataset.
3. Membagi Train 80% dan Valid 20%.

Mengganti CPU Intel dengan GPU Nvidia RTX 3050 untuk train lebih cepat dan optimal

1. Pelatihan 100 Epoch.
2. Penginputan Dataset Video.



Program Training Model

```python
# COPY TRAIN
print(f"[COPY] train → fold{fold}")
for img in glob(src_train + "/images/*"):
    shutil.copy(img, train_img)
for lbl in glob(src_train + "/labels/*"):
    shutil.copy(lbl, train_lbl)

# COPY VALID=
print(f"[COPY] valid → fold{fold}")
for img in glob(src_valid + "/images/*"):
    shutil.copy(img, valid_img)
for lbl in glob(src_valid + "/labels/*"):
    shutil.copy(lbl, valid_lbl)
```

```python
def train_fold(fold):
    print(f"\n[TRAIN] Fold {fold}\n")
    model = YOLO("yolo11m.pt")

    fold_dir = f"D:/KCBUAS/kfold/fold{fold}"
    yaml_path = os.path.join(fold_dir, "data.yaml")

    load_classes(yaml_path)

    model.train(
        data=yaml_path,
        epochs=100,
        batch=4,
        imgsz=640,
        name=f"fold_{fold}",
        pretrained=True,
        workers=2,
        half=True
    )
```

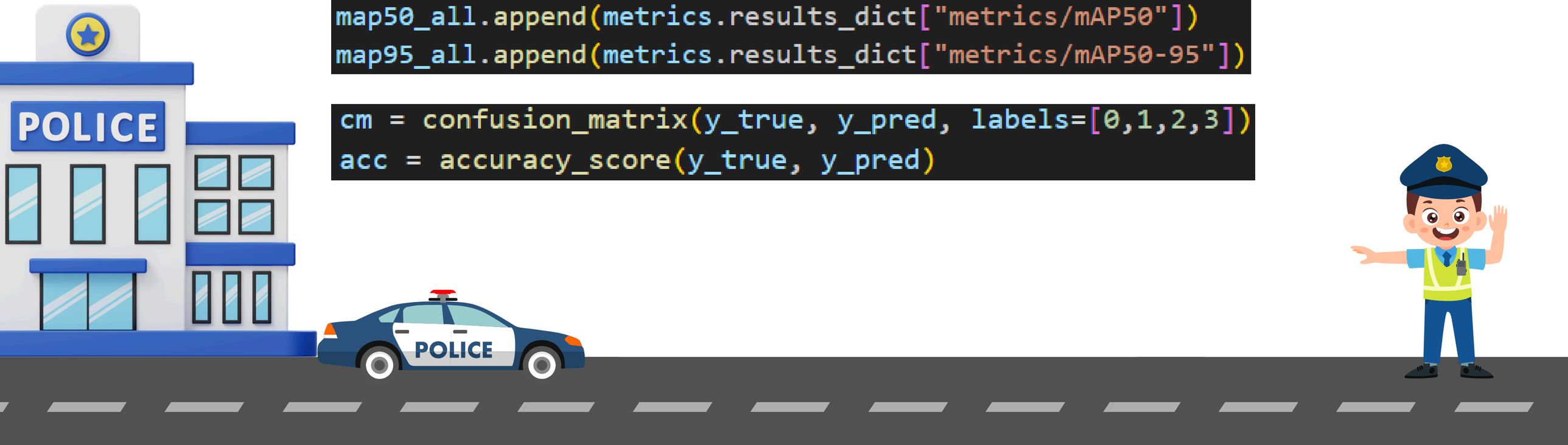
Program Training Model

```
results = model.predict(img, conf=0.25, verbose=False)
```

```
confs = results[0].boxes.conf.cpu().numpy()
clss = results[0].boxes.cls.cpu().numpy()
pred_class = int(clss[np.argmax(confs)])
```

```
metrics = model.val()
map50_all.append(metrics.results_dict["metrics/mAP50"])
map95_all.append(metrics.results_dict["metrics/mAP50-95"])
```

```
cm = confusion_matrix(y_true, y_pred, labels=[0,1,2,3])
acc = accuracy_score(y_true, y_pred)
```



Program Pemrosesan Video untuk Deteksi Objek, Tracking, dan Evaluasi Kecepatan Kendaraan

```python
image_pts = np.array([
    [533, 45],    # A
    [701, 45],    # B
    [1677, 855],  # D
    [56, 865]     # C

], dtype=np.float32)

LEBAR_JALAN_METER = 15
PANJANG_JALAN_METER = 60

world_pts = np.array([
    [0, 0],
    [LEBAR_JALAN_METER, 0],
    [LEBAR_JALAN_METER, PANJANG_JALAN_METER],
    [0, PANJANG_JALAN_METER]
], dtype=np.float32)

M = cv2.getPerspectiveTransform(image_pts, world_pts)
print("Matriks Transformasi (M) berhasil dibuat.")
```

```python
tracked_objects = {}  # track_id : (prev_x, prev_y, prev_time)
speed_history = {}    # track_id : list of previous speeds
ema_speed = {}        # track_id : last filtered speed

def classify_risk(speed_kmh):
    if speed_kmh >= 80: return "Tinggi", (0, 0, 255)
    elif speed_kmh >= 40: return "Sedang", (0, 255, 255)
    elif speed_kmh >= 20: return "Rendah", (0, 255, 0)
    else: return "Aman", (255, 255, 255)

print("Memulai proses prediksi dan perhitungan kecepatan...")

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        print("Video selesai diproses.")
        break

    current_time = cap.get(cv2.CAP_PROP_POS_MSEC) / 1000.0
```

Program Pemrosesan Video untuk Deteksi Objek, Tracking, dan Evaluasi Kecepatan Kendaraan

```python
#  HITUNG KECEPATAN  #
if track_id in tracked_objects:
    prev_x, prev_y, prev_time = tracked_objects[track_id]
    selisih_waktu = current_time - prev_time
    if selisih_waktu > 0:
        titik_lama = np.array([[[prev_x, prev_y]]], dtype=np.float32)
        titik_baru = np.array([[[center_x, center_y]]], dtype=np.float32)
        world_old = cv2.perspectiveTransform(titik_lama, M)
        world_new = cv2.perspectiveTransform(titik_baru, M)
        jarak_meter = np.linalg.norm(world_new - world_old)
        speed_ms = jarak_meter / selisih_waktu
        speed_kmh = speed_ms * 3.6
```

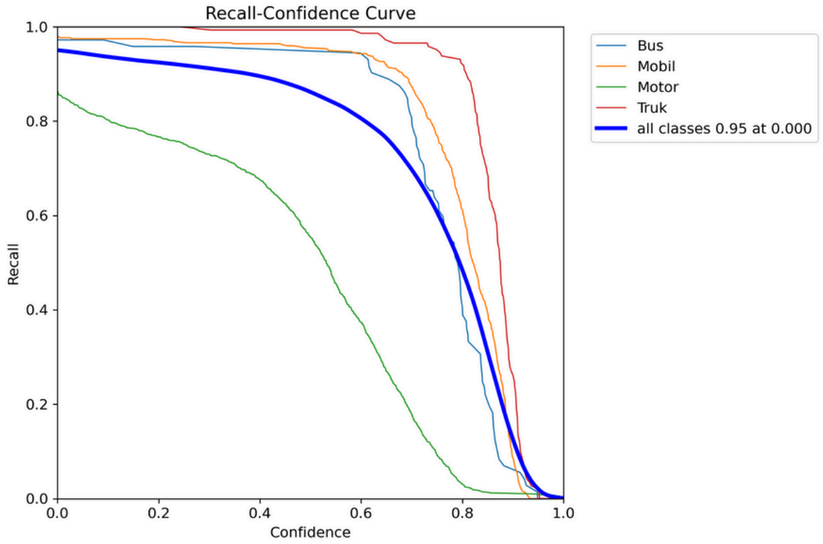
```python
#  STABILISASI KECEPATAN  #
if speed_kmh > 200:
    speed_kmh = 200

if track_id not in speed_history:
    speed_history[track_id] = []
speed_history[track_id].append(speed_kmh)
if len(speed_history[track_id]) > 5:
    speed_history[track_id].pop(0)
avg_speed = sum(speed_history[track_id]) / len(speed_history[track_id])

alpha = 0.25
if track_id not in ema_speed:
    ema_speed[track_id] = avg_speed
else:
    ema_speed[track_id] = (alpha * avg_speed) + (1 - alpha) * ema_speed[track_id]

final_speed = ema_speed[track_id]
```

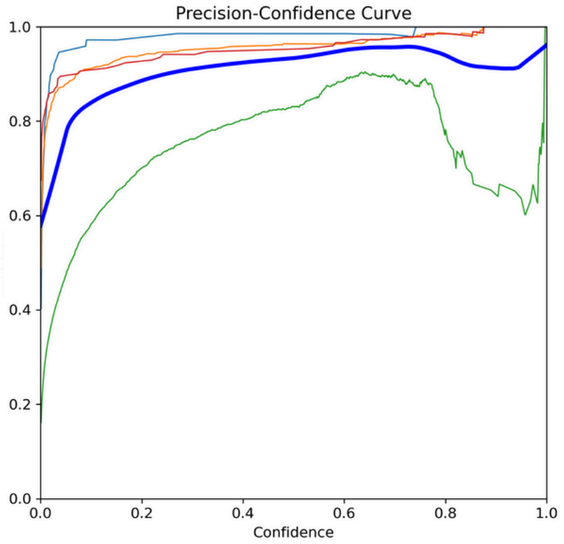
Hasil Selama Pelatihan (Train) Pembuatan Model Sistem

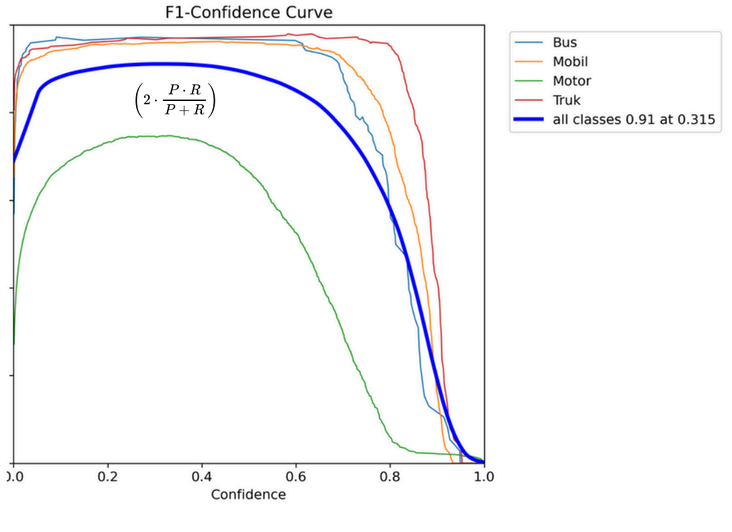
```
     Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    99/100      2.56G      0.995     0.4974     0.9385         43        640: 100% ━━━━━━━━━━━━ 165/165 3.3it/s 49.3s
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 28/28 8.1it/s 3.5s
                   all        222       2163      0.922      0.926      0.928      0.66

     Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
   100/100      2.57G     0.9896     0.4913     0.9365         53        640: 100% ━━━━━━━━━━━━ 165/165 3.3it/s 50.4s
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 28/28 7.9it/s 3.6s
                   all        222       2163      0.924      0.925       0.93      0.664

100 epochs completed in 2.219 hours.
Optimizer stripped from runs\detect\yolov11_train_kendaraan\weights\last.pt, 40.5MB
Optimizer stripped from runs\detect\yolov11_train_kendaraan\weights\best.pt, 40.5MB

Validating runs\detect\yolov11_train_kendaraan\weights\best.pt...
Ultralytics 8.3.228  Python-3.9.9 torch-2.8.0+cu126 CUDA:0 (NVIDIA GeForce RTX 3050 Laptop GPU, 4096MiB)
YOLO11m summary (fused): 125 layers, 20,033,116 parameters, 0 gradients, 67.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 28/28 8.2it/s 3.4s
                   all        222       2163      0.924      0.925       0.93      0.664
                   Bus         72         72      0.999      0.958      0.983      0.778
                 Mobil        178        474      0.951      0.968      0.969       0.72
                 Motor        207       1472      0.787      0.781      0.783       0.37
                  Truk        135        145       0.96      0.993      0.983       0.79
Speed: 0.3ms preprocess, 8.8ms inference, 0.0ms loss, 1.8ms postprocess per image
Results saved to runs\detect\yolov11_train_kendaraan
```

```
     Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
   100/100      1.93G       1.06      0.521     0.9426         12        416: 100% ━━━━━━━━━━━━ 371/371 7.3it/s 50.6s
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 61/61 15.4it/s 4.0s
                   all        242       2188      0.919      0.909      0.915       0.64

100 epochs completed in 1.565 hours.
Optimizer stripped from D:\KCBUAS\runs\detect\fold_0\weights\last.pt, 40.5MB
Optimizer stripped from D:\KCBUAS\runs\detect\fold_0\weights\best.pt, 40.5MB

Validating D:\KCBUAS\runs\detect\fold_0\weights\best.pt...
Ultralytics 8.3.228  Python-3.9.9 torch-2.8.0+cu126 CUDA:0 (NVIDIA GeForce RTX 3050 Laptop GPU, 4096MiB)
YOLO11m summary (fused): 125 layers, 20,033,116 parameters, 0 gradients, 67.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 61/61 17.6it/s 3.5s
                   all        242       2188      0.913       0.91      0.913      0.642
                   Bus         72         72      0.986      0.956      0.975      0.764
                 Mobil        178        474      0.953      0.966       0.97      0.703
                 Motor        227       1497      0.769      0.725      0.721      0.312
                  Truk        135        145      0.944      0.993      0.987      0.788
Speed: 0.3ms preprocess, 6.9ms inference, 0.0ms loss, 2.0ms postprocess per image
Results saved to D:\KCBUAS\runs\detect\fold_0
```

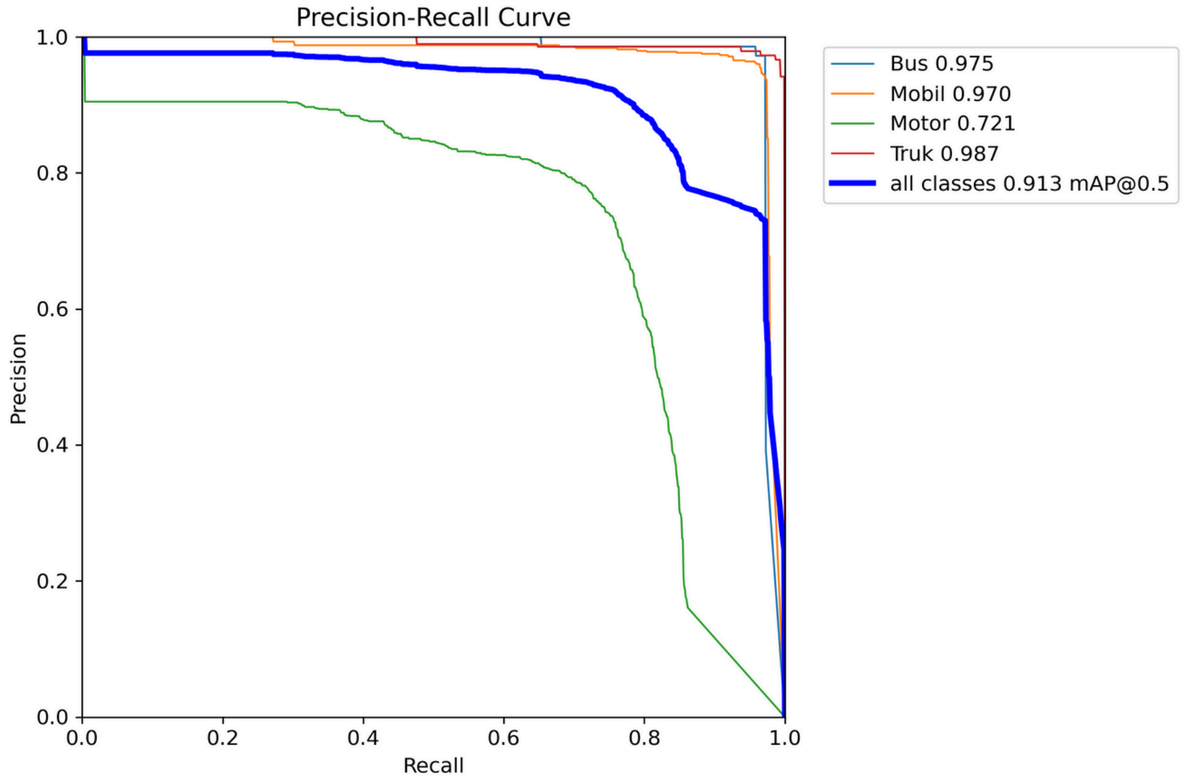
Hasil Selama Pelatihan (Train) Pembuatan Model Sistem



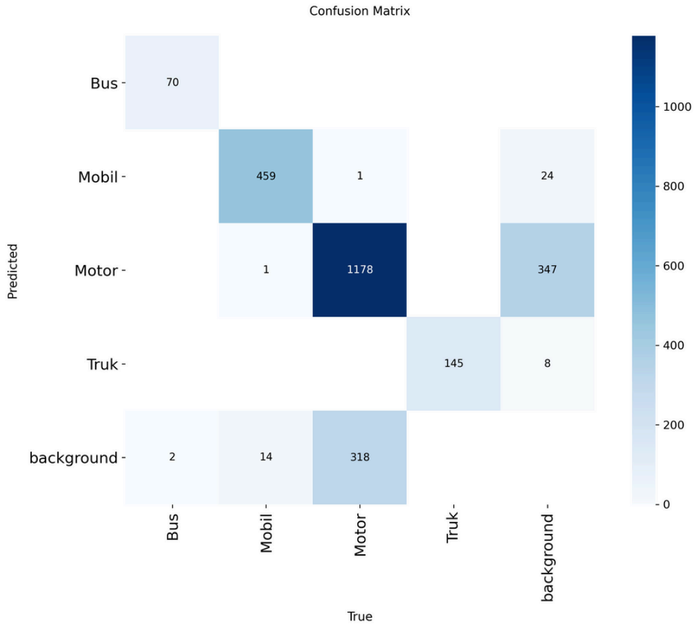


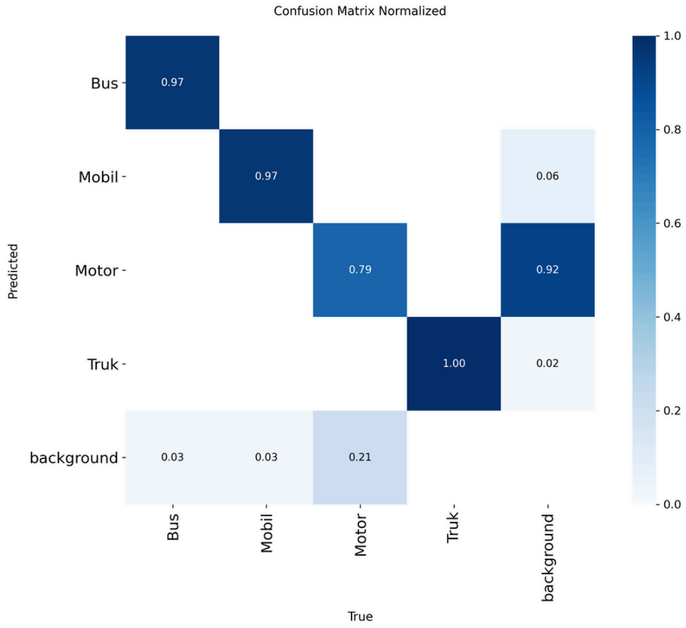


Hasil Selama Pelatihan (Train) Pembuatan Model Sistem



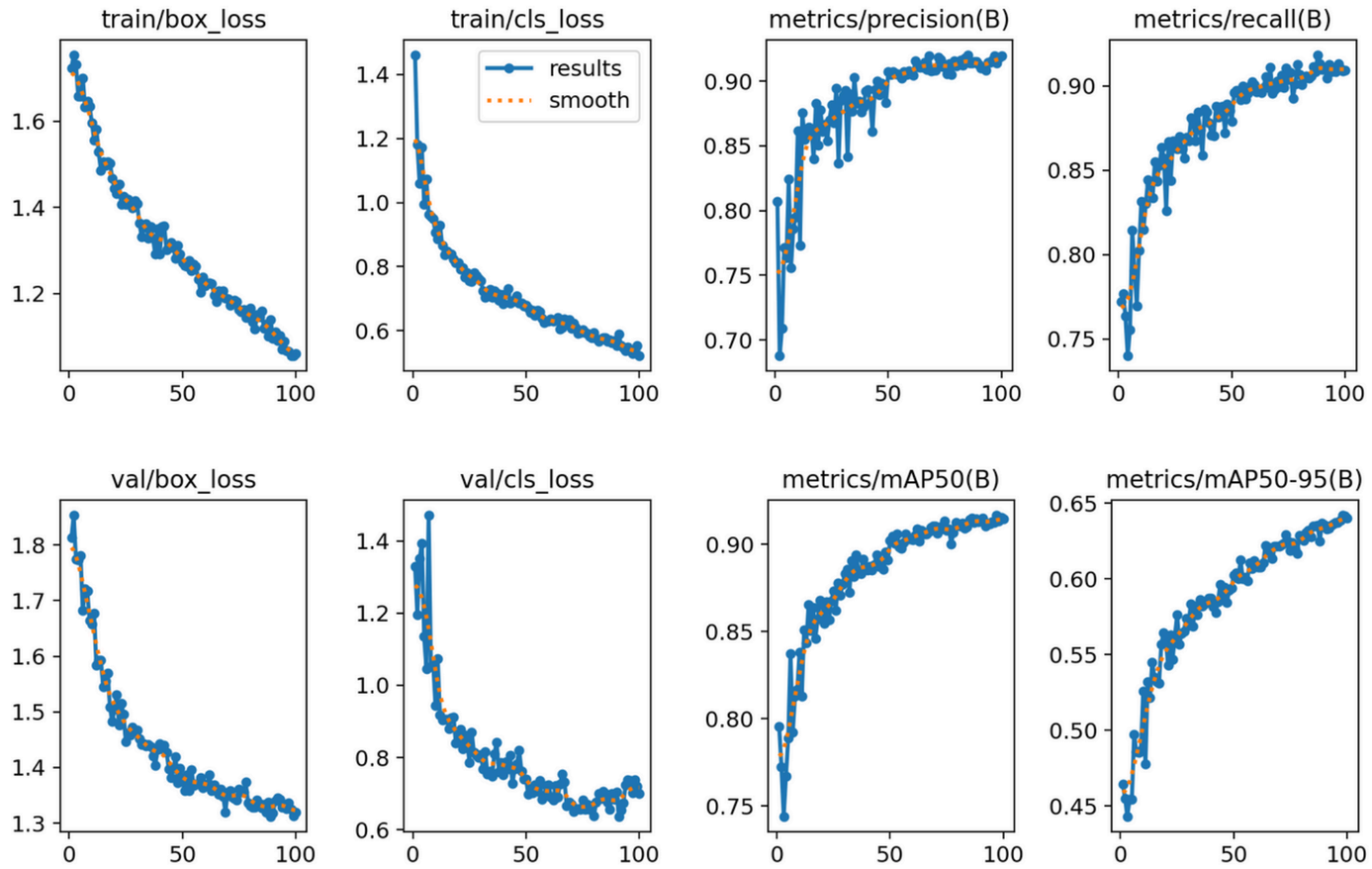
Hasil Selama Pelatihan (Train) Pembuatan Model Sistem



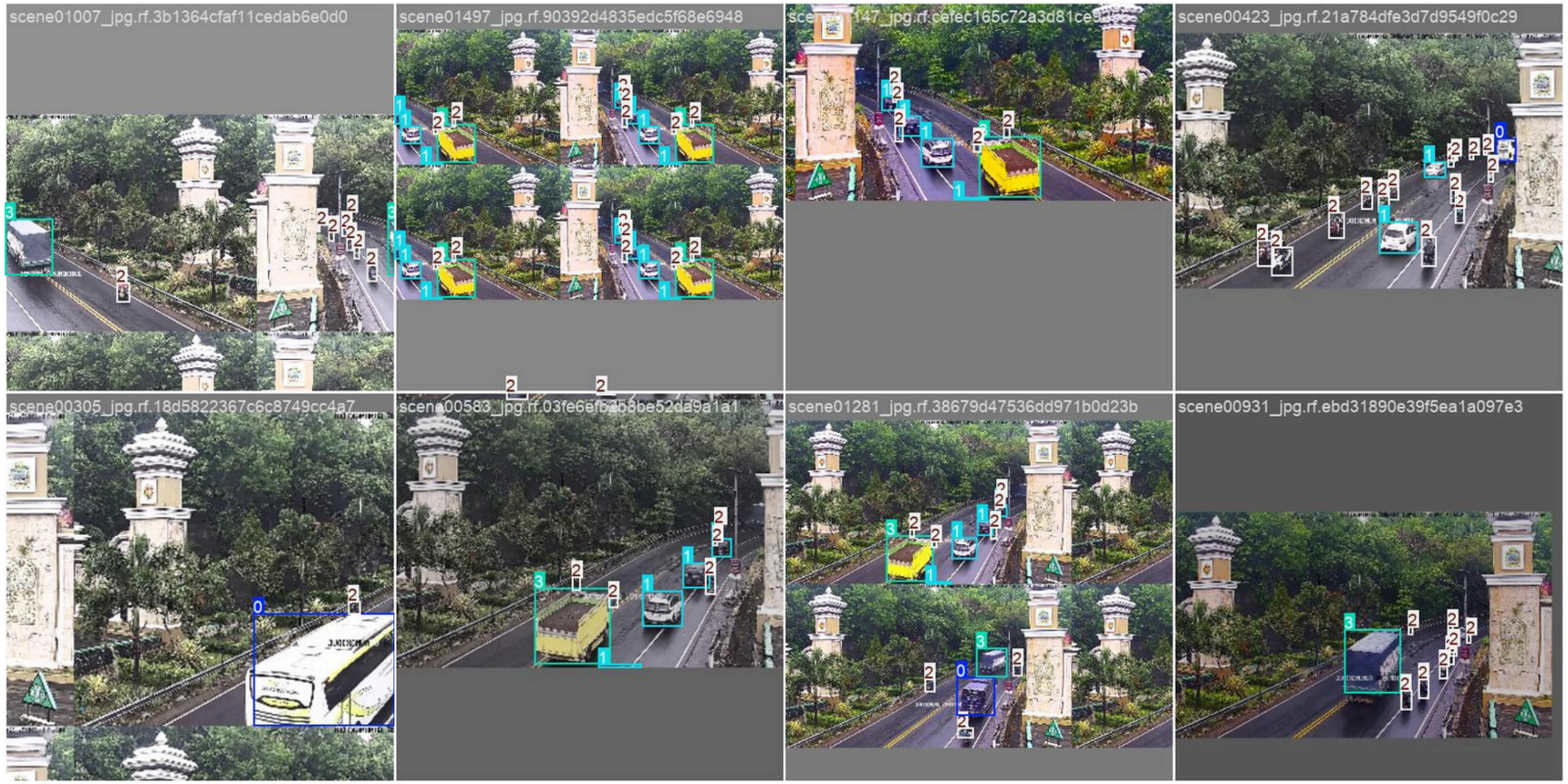


Class	Images	Instances	Box(P	R	mAP50	mAP50-95):
all	242	2188	0.913	0.91	0.913	0.642
Bus	72	72	0.986	0.956	0.975	0.764
Mobil	178	474	0.953	0.966	0.97	0.703
Motor	227	1497	0.769	0.725	0.721	0.312
Truk	135	145	0.944	0.993	0.987	0.788

Hasil Selama Pelatihan (Train) Pembuatan Model Sistem



Hasil Selama Pelatihan (Train) Pembuatan Model Sistem

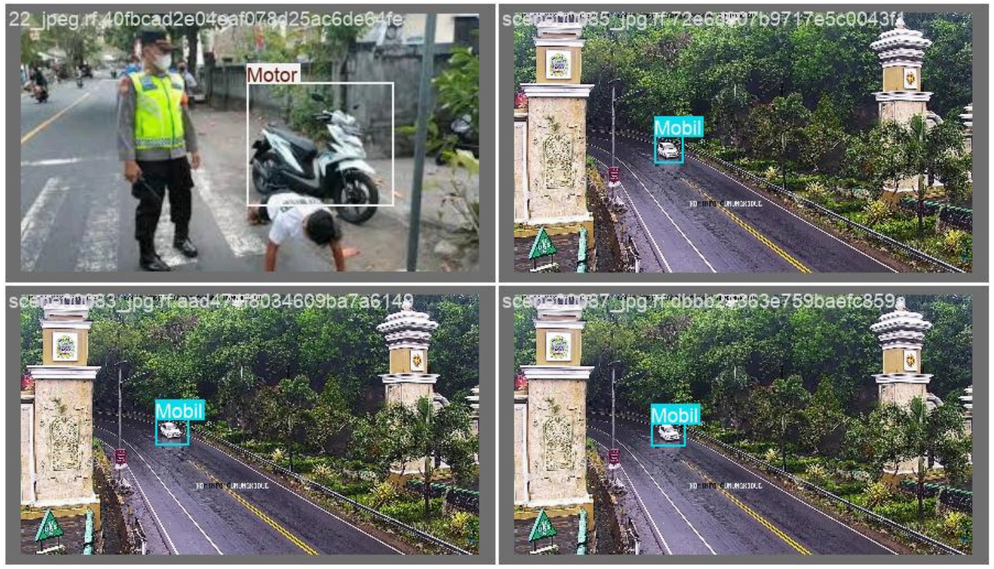


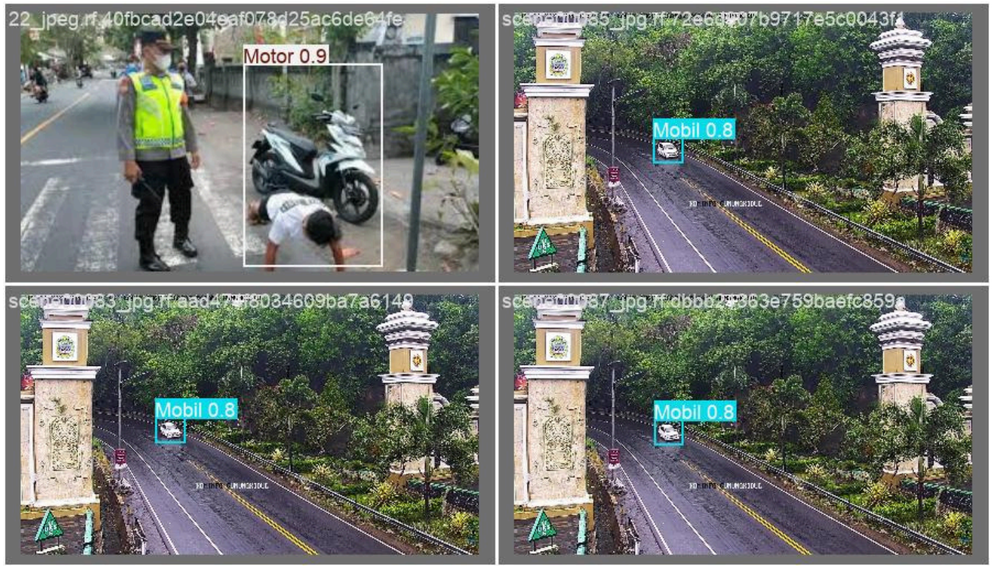
data.yaml

dataset > data.yaml

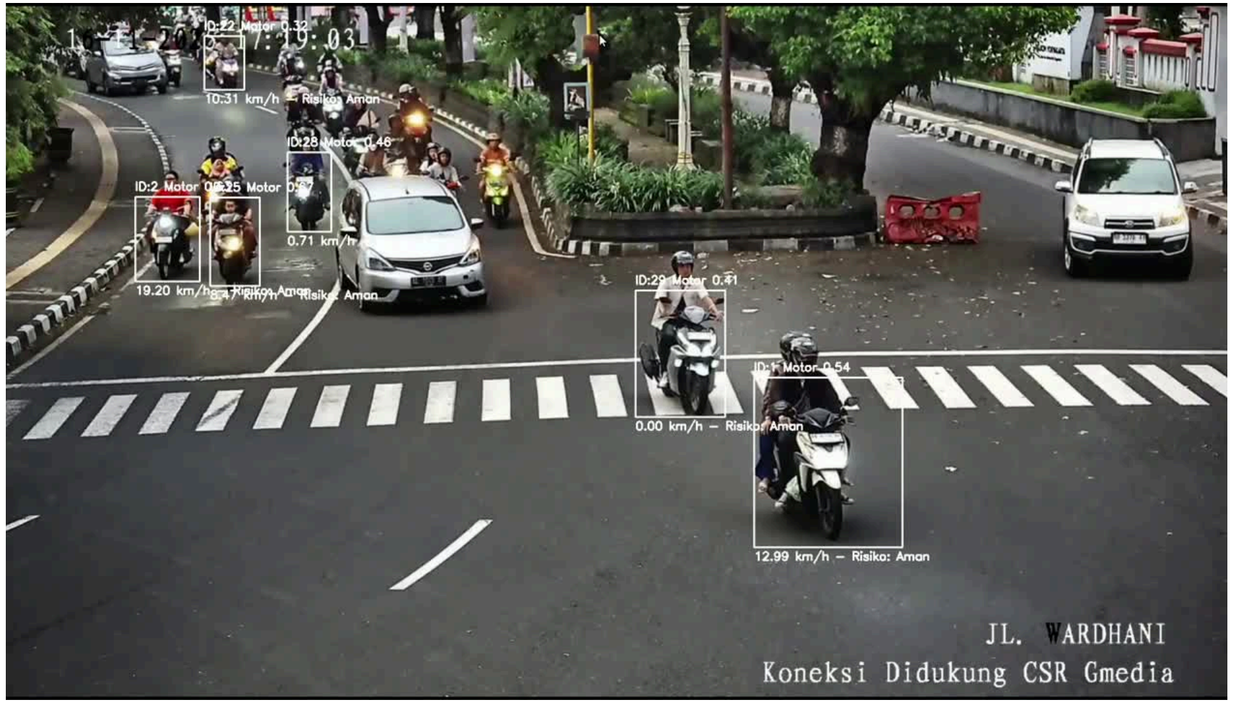
```yaml
train: ../train/images
val: ../valid/images
test: ../test/images

nc: 4
names: ['Bus', 'Mobil', 'Motor', 'Truk']
```

Hasil Selama Pelatihan (Train) Pembuatan Model Sistem

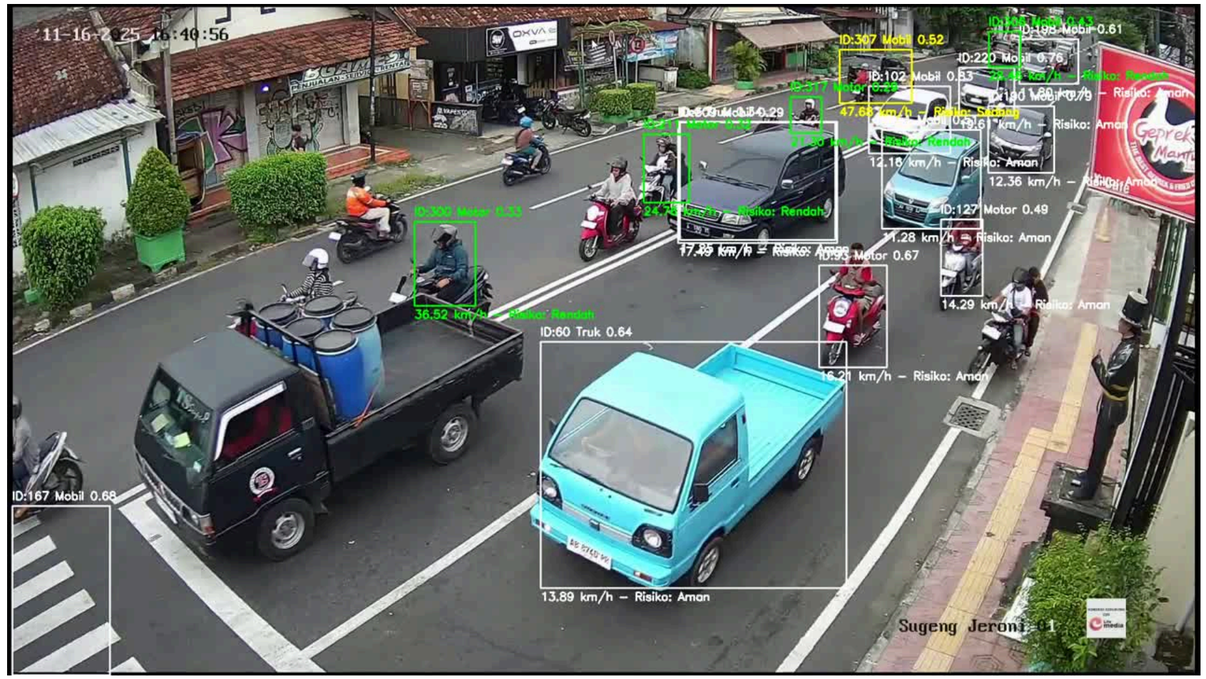


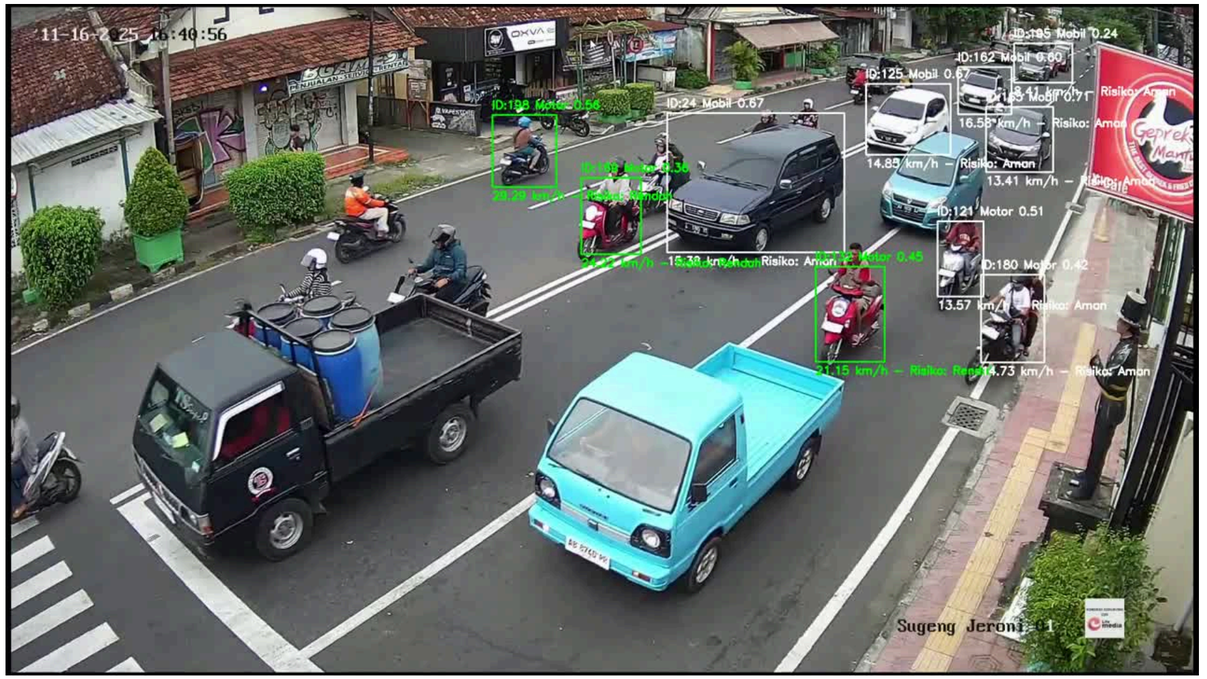


Hasil Perbandingan Train Default dan Train Pengembangan Data Uji



Hasil Perbandingan Train Default dan Train Pengembangan Data Uji





Hasil Data Uji Prediksi pada Input Video

Kesimpulan

Dari projek ini disimpulan dapat untuk mendeteksi kendaraan secara real-time, melacak pergerakannya, dan menghitung kecepatan berdasarkan perubahan posisi dalam frame video menggunakan transformasi perspektif.

Model yang digunakan telah dilatih selama 100 epoch menggunakan dataset kendaraan yang telah dianotasi, dan memberikan hasil yang baik pada proses pengujian. Sistem ini tidak hanya mampu mendeteksi jenis kendaraan, tetapi juga mengukur kecepatan dengan akurasi tinggi, kemudian menganalisis tingkat risiko kecelakaan berdasarkan kecepatan kendaraan yang terdeteksi.

Penggunaan YOLOv11 sebagai model deteksi, ByteTrack untuk pelacakan objek, serta metode estimasi kecepatan dari pergerakan pixel ke meter memungkinkan proses dilakukan secara real-time dan responsif. Untuk mengurangi fluktuasi kecepatan yang tidak akurat, digunakan teknik Exponential Moving Average (EMA) sehingga sistem dapat menampilkan hasil yang lebih stabil dan realistis.

Berdasarkan pengujian yang dilakukan, sistem mampu melakukan deteksi dan estimasi kecepatan secara efektif, sehingga berpotensi digunakan dalam analisis pelanggaran kecepatan, pemantauan lalu lintas, atau sistem peringatan dini kecelakaan.

Referensi

Kamil, D. A., Wahyono, & Harjoko, A. (2024). Vehicle Speed Estimation Using Consecutive Frame Approaches and Deep Image Homography for Image Rectification on Monocular Videos. IEEE Access. DOI:10.1109/ACCESS.2024.3508135.

Ahmed, M. W. (2024). Deep-Learning-Based Method for Vehicle Speed Estimation. Algorithms (MDPI).

Henny, H. (2023). Aplikasi Penghitung Kecepatan Mobil dengan Akurasi Tinggi. Jurnal Telekontran, Universitas Komputer Indonesia (UNIKOM).

Deteksi Kecepatan Kendaraan Berbasis Data Ponsel Dalam Perspektif Spasial, E. L. B. Pinem et al., Geodika: Jurnal Kajian Ilmu dan Pendidikan Geografi, Vol. 7, No.1, Juni 2023

Pengembangan Sistem Deteksi Kecepatan Kendaraan Overspeed dengan YOLOv8 di Area Jalan Tol, A. A. Fauzi, A. Wahid & A. B. Kaswar, JIMU: Jurnal Ilmiah Multidisipliner, Vol. 3, No.2, 2025.

Terima Kasih