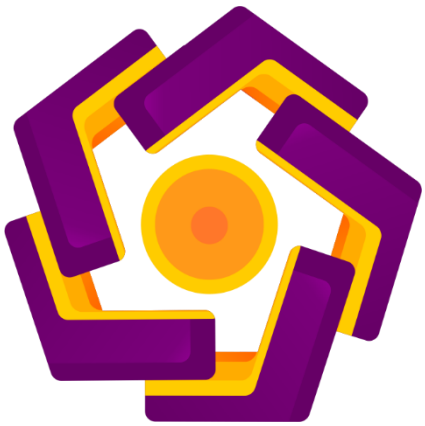
OBJECT DETECTION JENIS KENDARAAN MENGGUNAKAN ALGORITMA YOLO V8

Dosen pengampu : Anna Baita, S.Kom., M.Kom.



disusun oleh

FAIZ DAFFA KURNIA

22.11.4627

PROGRAM STUDI INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS AMIKOM YOGYAKARTA

YOGYAKARTA

2025

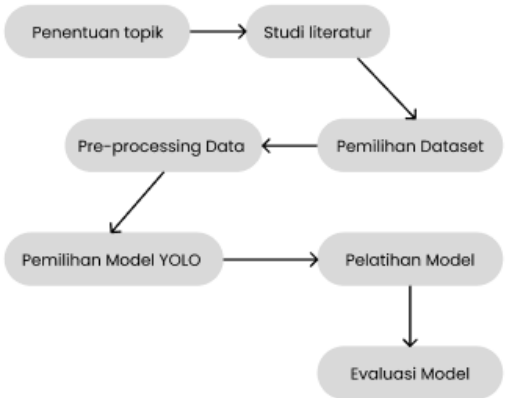
1. Latar Belakang

Perkembangan teknologi kecerdasan buatan, khususnya dalam bidang computer vision, memberikan pengaruh yang signifikan pada berbagai sektor, termasuk transportasi. Object detection, sebagai salah satu cabang computer vision, memungkinkan identifikasi dan klasifikasi objek dalam citra atau video secara real-time, yang sangat penting untuk aplikasi seperti pengelolaan lalu lintas, sistem keamanan jalan, dan analisis mobilitas perkotaan. Menurut Redmon et al. (2016), "YOLO (You Only Look Once) adalah pendekatan object detection yang menggabungkan kecepatan dan akurasi dengan memproses gambar dalam satu kali lintasan jaringan saraf" (Redmon et al., 2016). Pendekatan ini telah berevolusi hingga versi terbaru yaitu YOLO v8, yang memiliki peningkatan signifikan dalam hal akurasi dan efisiensi dibandingkan generasi sebelumnya.

Dalam konteks transportasi, deteksi jenis kendaraan seperti mobil, motor, truk, atau bus memiliki peran krusial. Sebagai contoh, sistem pengelolaan lalu lintas cerdas memerlukan deteksi kendaraan secara akurat untuk mengurangi kemacetan dan meningkatkan keselamatan jalan. Penelitian oleh Bochkovskiy et al. (2020) menunjukkan bahwa "algoritma berbasis deep learning seperti YOLO dapat mendeteksi objek dengan presisi tinggi bahkan dalam kondisi lingkungan yang menantang, seperti perubahan pencahayaan atau kepadatan lalu lintas" (Bochkovskiy et al., 2020). YOLO v8, yang dikembangkan oleh Ultralytics, mengintegrasikan arsitektur jaringan saraf konvolusi (CNN) yang canggih, memungkinkan deteksi cepat dan akurat pada dataset gambar kendaraan.

Proyek ini bertujuan untuk mengembangkan model object detection menggunakan YOLO v8 untuk mengidentifikasi jenis kendaraan. Dengan memanfaatkan dataset yang mencakup berbagai kondisi lingkungan, proyek ini diharapkan menghasilkan model yang robust untuk aplikasi di dunia nyata, seperti sistem pengawasan lalu lintas atau analisis pola transportasi.

2. Diagram Alur Penelitian



a. **Penentuan Topik & Studi Literatur**

Penelitian ini dimulai dengan menentukan topik utama yang menggunakan Algoritma YOLO v8 Object Detection untuk jenis jenis kendaraan yang ada di jalan raya. Disini juga melibatkan pengkajian literatur penelitian sebelumnya terkait teknologi object detection, khususnya YOLO v8, dan tentang deteksi kendaraan berbasis deep learning untuk memahami metode dan tantangan yang ada.

b. **Pemilihan Dataset**

Dataset yang digunakan bersumber dari Kaggle. Berisi 1254 jumlah file .jpg dan 1254 file .txt, dataset ini memiliki 5 kelas kendaraan yaitu mobil, motor, ambulan, bis, dan truk, disertai dengan labelnya.
Link dataset : https://www.kaggle.com/datasets/alkanerturan/vehicledetection

c. **Pre-processing Data**

Preprocessing data dilakukan terlebih dahulu sebelum pengolahan data dan pembuatan model supaya data yang digunakan memberikan hasil yang maksimal dan akurat.

d. **Pemilihan Model YOLO**

Model YOLO memiliki beberapa jenis varian model yang berbedea ukuran dan kapasitasnya mulai dari yang paling ringan yaitu YOLOv8n hingga yang tertinggi YOLOv8x. Saya memilih menggunakan YOLOv8n (nano) dikarenakan prosesnya yang paling ringan dan cepat, walaupun tingkat keakurasian yang diperoleh akan lebih rendah.

e. **Pelatihan Model**

Data akhir yang telah diproses pada tahapan pre-processing akan lanjut dilatih menggunakan Model YOLOv8n dan menggunakan data val untuk validasi, guna mengoptimalkan kemampuan model dalam mengenali jenis kendaraan.

f. **Evaluasi Model**

Performa model akan dievaluasi menggunakan metrik mAP (mean Average Precision), IoU (Intersection over Union), dan tingkat akurasi deteksi untuk memastikan model dapat mendeteksi jenis kendaraan dengan baik.

3. Analisis Kode Program

1. Mount dataset

```
[ ] from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive
```

Menggunakan from google.colab import drive untuk mengconnect dengan Google Drive ke direktori /content/drive. Pesan Mounted at /content/drive menunjukkan bahwa proses mounting telah berhasil.

2. Pre-processing Data

Menghitung Jumlah File Dataset

```
[ ] def count_images_labels(path):
        for subset in ['train', 'valid', 'test']:
            img_path = os.path.join(path, subset, 'images')
            label_path = os.path.join(path, subset, 'labels')
            img_count = len(os.listdir(img_path))
            label_count = len(os.listdir(label_path))
            print(f"{subset.upper()} : Images = {img_count}, Labels = {label_count}")

    count_images_labels(dataset_path)

    TRAIN : Images = 878, Labels = 878
    VALID : Images = 250, Labels = 250
    TEST : Images = 126, Labels = 126
```

Menghitung keseluruhan jumlah file gambar dan teks dalam direktori train, test, dan val. Ini bertujuan untuk mencari missing value.

Mengecek Format Dataset

```
[ ] import os
    from collections import Counter

    # Path ke folder gambar
    img_dir = os.path.join(dataset_path, 'train', 'images')

    # Ambil seluruh ekstensi file
    extensions = []
    for file in os.listdir(img_dir):
        if os.path.isfile(os.path.join(img_dir, file)):
            ext = os.path.splitext(file)[1].lower()  # ambil ekstensi, huruf kecil
            extensions.append(ext)

    # Hitung masing-masing format
    ext_counter = Counter(extensions)

    # Tampilkan hasil
    print("Distribusi format gambar:")
    for ext, count in ext_counter.items():
        print(f"{ext} : {count} file")

    Distribusi format gambar:
    .jpg : 878 file
```

Lalu melakukan pengecekan format gambar menggunakan os agar memastikan semua gambar formatnya sama. Penting untuk kompatibilitas model YOLO v8.

Menghitung Distribusi Objek di Seluruh Kelas

```
[ ] from collections import Counter

    def count_classes(label_dir):
        class_counts = Counter()
        for file in os.listdir(label_dir):
            with open(os.path.join(label_dir, file), 'r') as f:
                for line in f.readlines():
                    # Change here to handle float strings:
                    class_id = int(float(line.split()[0]))
                    class_counts[class_id] += 1
        return class_counts

    class_names = ['Ambulance', 'Bus', 'Car', 'Motorcycle', 'Truck']

    # Gabungkan semua subset
    all_labels = [os.path.join(dataset_path, "train/labels"),
                  os.path.join(dataset_path, "valid/labels"),
                  os.path.join(dataset_path, "test/labels")]
    total_class_count = Counter()
    for folder in all_labels:
        total_class_count += count_classes(folder)

    print("Distribusi Kelas:")
    for cls_id, count in total_class_count.items():
        class_name = class_names[cls_id] if cls_id < len(class_names) else f"Unknown({cls_id})"
        print(f"{class_name.capitalize()} (Class {cls_id}) = {count} objek")
```

```
Distribusi Kelas:
Car (Class 2) = 1302 objek
Bus (Class 1) = 282 objek
Motorcycle (Class 3) = 280 objek
Ambulance (Class 0) = 252 objek
Truck (Class 4) = 272 objek
```

Lalu pengecekan distribusi keseimbangan kelas, hasilnya menunjukkan ketidakseimbangan pada kelas car sehingga perlu di augmentasi.

Augmentasi Data untuk Menyeimbangkan Kelas

```
[ ] import albumentations as A
    import cv2
    import os
    import shutil

    # Definisikan augmentasi
    augment = A.Compose([
        A.HorizontalFlip(p=0.5),
        A.RandomBrightnessContrast(p=0.3),
        A.RandomGamma(p=0.3),
        A.Rotate(limit=10, p=0.5),
    ], bbox_params=A.BboxParams(format='yolo', label_fields=['class_labels']))

    # Kelas minor yang ingin di-augmentasi
    minor_classes = [0, 1 , 3, 4]  # Ambulance, Bus, Motorcycle, Truck

    # Paths
    train_img_dir = os.path.join(dataset_path, 'train', 'images')
    train_lbl_dir = os.path.join(dataset_path, 'train', 'labels')

    # Folder baru untuk hasil augmentasi
    aug_img_dir = os.path.join(dataset_path, 'train', 'images_aug')
    aug_lbl_dir = os.path.join(dataset_path, 'train', 'labels_aug')
    os.makedirs(aug_img_dir, exist_ok=True)
    os.makedirs(aug_lbl_dir, exist_ok=True)
```

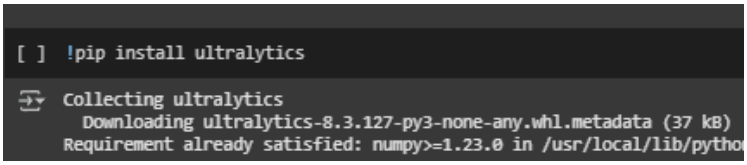
Augmentasi dilakukan untuk menduplikasikan kelas lain selain car, dan disimpan di folder baru, yang nantinya akan di merge supaya jumlah kelas seimbang.

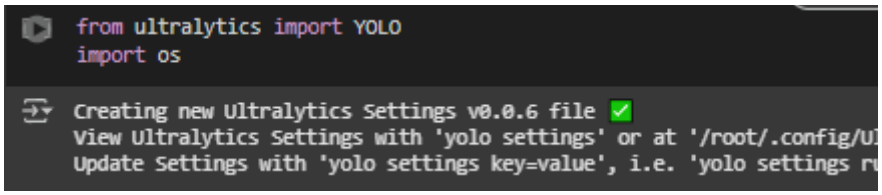
Menampilkan Gambar Random dengan Bounding Box dan Label

```
[ ] import random

    def plot_image_with_boxes(img_path, label_path, class_names):
        # img_path should be the full path to an image file
        img = cv2.imread(img_path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        h, w = img.shape[:2]

        with open(label_path, 'r') as f:
            labels = f.readlines()

        for label in labels:
            cls, x, y, bw, bh = map(float, label.strip().split())
            x1 = int((x - bw / 2) * w)
            y1 = int((y - bh / 2) * h)
            x2 = int((x + bw / 2) * w)
            y2 = int((y + bh / 2) * h)
            cv2.rectangle(img, (x1, y1), (x2, y2), (255, 0, 0), 2)
            cv2.putText(img, class_names[int(cls)], (x1, y1-5), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 0, 0), 2)

        plt.figure(figsize=(10, 6))
        plt.imshow(img)
        plt.axis('off')
        plt.title("Contoh Gambar + Bounding Box")
        plt.show()

    # Get all images and labels
    img_dir = os.path.join(dataset_path, "train", "images")
    label_dir = os.path.join(dataset_path, "train", "labels")

    # Choose a random image and its label file
    img_files = os.listdir(img_dir)
    random_image = random.choice(img_files)
    random_image_path = os.path.join(img_dir, random_image)
    random_label_path = os.path.join(label_dir, random_image[:-4] + '.txt')

    plot_image_with_boxes(random_image_path, random_label_path, class_names)
```

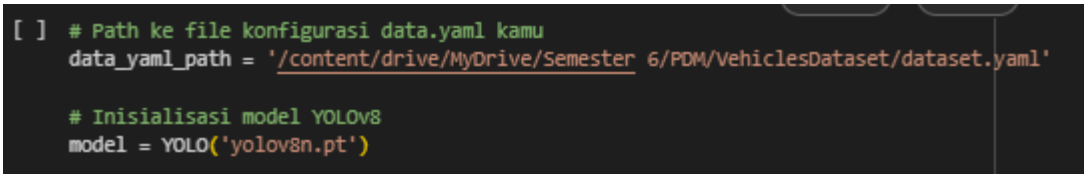
Lalu proses visualisasi bounding box untuk memastikan bahwa anotasi label pada dataset telah sesuai dengan objek yang ditandai. Bounding box ini dibuat berdasarkan informasi label berformat YOLO, yaitu nilai-nilai relatif dari posisi tengah objek dan ukuran lebar serta tinggi objek. Fungsi plot_image_with_boxes membaca gambar dan file label yang sesuai, lalu menghitung koordinat aktual bounding box dalam piksel. Selanjutnya, kotak ditampilkan di atas gambar beserta nama kelas objek.

3. Model Training & Evaluasi





Menginstall paket ultralytics yang mencakup library YOLO v8. Diikuti dengan mengimpor kelas YOLO dari library ultralytics.



Kode selanjutnya mengatur path konfigurasi data (data.yaml) di Google Drive dan inisialisasi model YOLO v8 dengan model = YOLO('yolov8n.pt'), yang menunjukkan penggunaan model pre-trained YOLO v8 nano untuk deteksi kendaraan.

Kode ini berfungsi untuk memulai proses pelatihan model. Ini menunjukkan pelatihan dilakukan selama 50 epoch dengan dengan ukuran gambar 416 dan batch size 16, menyimpan hasil di direktori spesifik, dan menggunakan GPU (device=0) untuk optimasi performa deteksi jenis kendaraan.

```
# Evaluate model pada data validasi
metrics = model.val()
print(metrics)
```

```
Ultralytics 8.3.124 🚀 Python-3.11.12 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 72 layers, 3,006,623 parameters, 0 gradients, 8.1 GFLOPs
val: Fast image access ✅ (ping: 12.6±27.1 ms, read: 10.4±11.1 MB/s, size: 31.4 KB)
val: Scanning /content/drive/MyDrive/Semester 6/PDM/VehiclesDataset/valid/labels.cache... 250 images,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all        250        454      0.679      0.553      0.592       0.45
             Ambulance         50         64      0.726      0.844      0.817      0.713
                   Bus         30         46      0.638      0.614      0.663      0.567
                   Car         90        238      0.561      0.472        0.5       0.34
            Motorcycle         42         46      0.677      0.456      0.536      0.291
                 Truck         38         60       0.79      0.377      0.446      0.338
Speed: 1.2ms preprocess, 3.0ms inference, 0.0ms loss, 2.1ms postprocess per image
Results saved to /content/drive/MyDrive/Semester 6/PDM/VehiclesDataset/yolo_training_output/yolov8_veh
ultralytics.utils.metrics.DetMetrics object with attributes:
```

Setelah proses training model, diperoleh hasil evaluasi model YOLOv8 pada data validasi yang terdiri dari 250 gambar dengan total 454 objek, model menunjukkan performa yang cukup baik secara keseluruhan dengan nilai **mAP@0.5 sebesar 59.2%** dan **mAP@0.5:0.95 sebesar 45.0%**. Kelas dengan performa terbaik adalah *Ambulance*, dengan precision 72.6%, recall 80.4%, dan mAP tertinggi yaitu 81.7% pada IoU 0.5. Sementara itu, kelas *Car*, Motorcycle dan *Truck* memiliki mAP yang lebih rendah, masing-masing 50.0%, 53.6%, dan 44.6%, dengan recall yang juga rendah, menandakan model masih kurang optimal dalam mendeteksi ketiga jenis kendaraan tersebut. Secara umum, model sudah cukup baik dalam mengenali objek tertentu namun masih perlu peningkatan, terutama dalam meningkatkan recall dan akurasi pada kelas dengan performa rendah.

```
[ ] best_model = YOLO('/content/drive/MyDrive/Semester 6/PDM/VehiclesDataset/yolo_training_output/yolov8_vehicles_model/weights/best.pt')
    metrics = best_model.val(split = 'test')
```

```
Ultralytics 8.3.124 🚀 Python-3.11.12 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 72 layers, 3,006,623 parameters, 0 gradients, 8.1 GFLOPs
val: Fast image access ✅ (ping: 0.6±0.1 ms, read: 2.3±5.0 MB/s, size: 26.2 KB)
val: Scanning /content/drive/MyDrive/Semester 6/PDM/VehiclesDataset/test/labels... 126 images, 0 backgrounds, 0 corrupt: 100%|

                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|          | 8/8 [00:02<00:00,  3.49it/s
                   all        126        258      0.735      0.587      0.654      0.526
             Ambulance         18         18      0.721      0.889       0.92      0.884
                   Bus         22         38      0.867      0.689       0.78      0.575
                   Car         60        150      0.696       0.67      0.686      0.458
            Motorcycle         12         32      0.568      0.188      0.278      0.207
                 Truck         14         20      0.822        0.5      0.607      0.507
Speed: 1.4ms preprocess, 3.3ms inference, 0.0ms loss, 2.7ms postprocess per image
Results saved to runs/detect/val
```

Berdasarkan hasil evaluasi model terbaik (best model) YOLOv8 yang terdiri dari 126 gambar dan 258 objek, model menunjukkan performa yang cukup baik dengan nilai **mAP@0.5 sebesar 65.4%** dan **mAP@0.5:0.95 sebesar 52.6%**, yang merupakan peningkatan dibanding hasil pada data validasi. Kelas *Ambulance* kembali menjadi kelas dengan performa tertinggi, memiliki precision 72.1%, recall 88.9%, dan nilai mAP@0.5 tertinggi yaitu 92.0%. Sementara itu, kelas *Bus* dan *Car* juga menunjukkan hasil yang baik dengan mAP masing-masing sebesar 78.0% dan 68.6%. Namun, kelas *Motorcycle* menunjukkan performa yang rendah dengan recall hanya 18.8% dan mAP@0.5 sebesar 27.0%, yang menandakan model masih kesulitan mendeteksi objek motor secara konsisten.

Link google Colab : https://colab.research.google.com/drive/1YkfJiOliOkZ0-KTE0jzyexgpttKqM6Ee?usp=sharing

Dari hasil evaluasi tersebut, ada beberapa cara yang dapat digunakan untuk memperbaiki beberapa hal terutama meningkatkan recall dan akurasi pada kelas motorcycle, car, dan truck. Hal ini bisa disebabkan oleh kemiripan bentuk antar kendaraan, distribusi data yang tidak seimbang, ataupun faktor lainnya. Dengan menggunakan model YOLOv8 yang lebih akurat seperti YOLOv8s, atau YOLOv8m kemungkinan tingkat keakurasiannya juga semakin besar, akan tetapi memakan waktu yang jauh lebih lama. Bisa juga dengan cara menaikkan jumlah epoch dengan catatan tidak overfitting, jumlah epoch bisa dicoba dinaikkan ke 100 dan dipantau mAP nya apakah meningkat ataupun malah stagnan atau bahkan menurun.

4. Rancangan UI/UX

Link Figma : https://www.figma.com/design/AhRT4Cyb1lzGPUXQkrBEYo/UI-PDM?node-id=0-1&t=bdPtognSX4MwwjpU-1