

**IMPLEMENTASI YOLO UNTUK PENDETEKSI KEMACETAN LALU  
LINTAS SECARA *REAL TIME***

**SKRIPSI**

**FAJRIATMOKO NUGROHO**

**201401143**



**PROGRAM STUDI S-1 ILMU KOMPUTER**

**FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI**

**UNIVERSITAS SUMATERA UTARA**

**MEDAN**

**2024**

**IMPLEMENTASI YOLO UNTUK PENDETEKSI KEMACETAN LALU  
LINTAS SECARA *REAL TIME***

**SKRIPSI**

**Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah  
Sarjana Ilmu Komputer**

**FAJRIATMOKO NUGROHO**

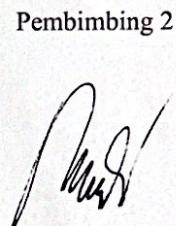
**201401143**



**PROGRAM STUDI S-1 ILMU KOMPUTER  
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI  
UNIVERSITAS SUMATERA UTARA  
MEDAN  
2024**

### PERSETUJUAN

Judul : IMPLEMENTASI YOLO UNTUK  
 PENDETEKSI KEMACETAN  
 LALU LINTAS SECARA  
*REAL TIME*  
 Kategori : SKRIPSI  
 Nama : FAJRIATMOKO NUGROHO  
 Nomor Induk Mahasiswa : 201401143  
 Program Studi : SARJANA (S-1) ILMU KOMPUTER  
 Fakultas : ILMU KOMPUTER DAN TEKNOLOGI  
 INFORMASI UNIVERSITAS SUMATERA  
 UTARA  
 Komisi Pembimbing :  
 Pembimbing 2



Prof. Dr. Poltak Sihombing M.Kom.  
NIP. 196203171991031001

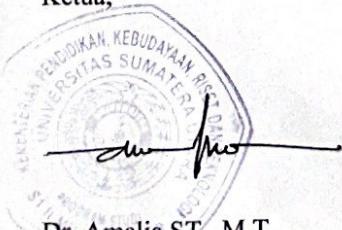
Medan, 08 Juli 2024

Pembimbing 1



Herriyance S.T., M.Kom  
NIP. 198010242010121002

Diketahui/Disetujui Oleh  
Program Studi S-1 Ilmu Komputer  
Ketua,



Dr. Amalia ST., M.T.  
NIP. 197812212014042001

**PERNYATAAN****IMPLEMENTASI YOLO UNTUK PENDETEKSI KEMACETAN LALU  
LINTAS SECARA *REAL TIME*****SKRIPSI**

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, 08 Juli 2024



Fajriatmoko Nugroho

201401143

## PENGHARGAAN

*Bismillahirrahmanirrahim,* Dengan menyebut nama Allah Yang Maha Pengasih lagi Maha Penyayang, segala puji hanya milik Allah, Tuhan semesta alam, atas segala nikmat dan petunjuk-Nya yang telah memungkinkan saya untuk menyelesaikan penulisan skripsi ini sebagai persyaratan untuk memperoleh gelar Sarjana Komputer di jurusan Ilmu Komputer, Universitas Sumatera Utara.

Penulis ingin mengungkapkan penghargaan dan terima kasih yang besar kepada:

1. Bapak Prof. Dr. Muryanto Amin, S.Sos, M.Si. selaku Rektor Universitas Sumatera Utara.
2. Ibu Dr. Maya Silvi Lydia, B.Sc., M.Sc., selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara.
3. Ibu Dr. Amalia, ST. MT., selaku Kepala Program Studi S-1 Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara.
4. Ibu Sri Melvani Hardi S.Kom., M.Kom., sebagai Sekretaris Program Studi S1 Ilmu Komputer Universitas Sumatera Utara
5. Bapak Herriyance S.T., M.Kom., M.Kom., selaku Dosen Pembimbing Akademik sekaligus dosen pembimbing I, yang telah memberikan arahan, nasihat, dan dukungan kepada penulis sebagai mahasiswa dalam menempuh berbagai tahapan akademik dan telah memberikan bimbingan, kritik, saran, serta dukungan semangat kepada penulis dalam menyelesaikan skripsi ini.
6. Bapak Prof. Drs. Poltak Sihombing M.Kom., Ph.D, selaku dosen pembimbing II yang telah memberikan bimbingan, arahan, dukungan serta masukan kepada penulis.
7. Seluruh Dosen Program Studi S-1 Ilmu Komputer yang telah meluangkan waktu dan energi untuk mengajar dan membimbing, sehingga penulis dapat mencapai tahap penyusunan skripsi ini.
8. Pak Seniman S.Kom., M.Kom. selaku dosen penguji yang telah memberikan kritik dan saran kepada penulis.

9. Ibu Hayatunnufus S.Kom, M.Cs., selaku dosen penguji yang telah memberikan kritik dan saran kepada penulis.
10. Kepada kedua orang tua, Bapak Haspin Wallad dan Ibu Hesti Kuswandari, yang telah berjuang keras untuk kelancaran kuliah penulis, dan juga skripsi yang telah penulis selesaikan ini, serta cinta dan kasih sayang mereka yang senantiasa tak kunjung habis.
11. Kepada Regina Putri Patrecella, terima kasih telah memberikan penulis motivasi hingga nasehat dalam menyelesaikan proses pembuatan skripsi dan senantiasa menemani perjuangan penulis.
12. Sahabat penulis yaitu Acel, Amru, Koko, Jubed, Sonia, Cemol dan Andrew yang telah bersama-sama perjalanan kuliah dan menjalani kehidupan perantauan bersama.
13. Dan semua pihak yang terlibat, baik secara langsung maupun tidak langsung, yang tidak dapat disebutkan satu per satu.

Medan, 08 Juli 2024



Fajriatmoko Nugroho

201401143

## ABSTRAK

### **IMPLEMENTASI YOLO UNTUK Pendeteksi KEMACETAN LALU LINTAS SECARA *REAL TIME***

Kemacetan lalu lintas merupakan tantangan serius yang dihadapi oleh sistem transportasi perkotaan di seluruh dunia. Dalam upaya mengatasi masalah ini, implementasi teknologi deteksi objek berbasis YOLO (You Only Look Once) telah menjadi perhatian utama. Pada penelitian ini dibangun sistem yang dapat mendeteksi kemacetan lalu lintas secara real-time dengan menggunakan algoritma YOLOv8, guna mengurangi kemacetan lalu lintas dengan mencegah terjadinya penumpukan pada persimpangan jalan. penelitian ini melibatkan tahapan studi pustaka untuk memahami konsep deteksi objek menggunakan algoritma YOLOv8, pengumpulan dataset, pelabelan, training, evaluasi dan implementasi sistem berbasis IOT dengan memanfaatkan kamera ESP32-CAM untuk menangkap kondisi lalu lintas secara real-time. Dari hasil pengujian yang dilakukan untuk mengevaluasi performa terhadap model dengan menggunakan kombinasi epoch 100 dengan learning rate 0.01, didapatkan nilai mAP tertinggi yaitu sebesar 0.849 untuk semua kelas dan nilai F1 confidence score sebesar 0.713. Penelitian ini menunjukkan bahwa sistem dengan model YOLOv8 mampu mendeteksi objek kendaraan dan menganalisa kondisi pada persimpangan jalan secara real-time.

Kata kunci : Kemacetan lalu lintas, YOLOv8, deteksi objek, pengenalan kendaraan, *real-time*, sistem manajemen lalu lintas.

***ABSTRACT******IMPLEMENTATION OF YOLO FOR REAL TIME TRAFFIC  
CONGESTION DETECTION***

*Traffic congestion is a serious challenge faced by urban transportation systems worldwide. In an effort to address this issue, the implementation of YOLO (You Only Look Once) object detection technology has gained significant attention. This research builds a system capable of detecting traffic congestion in real-time using the YOLOv8 algorithm to reduce traffic congestion by preventing bottlenecks at road intersections. The study involves several stages, including a literature review to understand the concept of object detection using the YOLOv8 algorithm, dataset collection, labeling, training, evaluation and system implementation based on IoT, utilizing the ESP32-CAM camera to capture real-time traffic conditions. From the testing conducted to evaluate the model's performance using a combination of 100 epochs with a learning rate of 0.01, the highest mAP value obtained was 0.849 for all classes, and the F1 confidence score was 0.713. This research demonstrates that a system using the YOLOv8 model is capable of detecting vehicle objects and analyzing conditions at road intersections in real-time..*

*Keywords:* Traffic congestion, YOLOv8, object detection, vehicle recognition, real-time, traffic management system.

## DAFTAR ISI

<b>PERSETUJUAN.....</b>	iii
<b>PENGHARGAAN.....</b>	v
<b>ABSTRAK .....</b>	vii
<b>ABSTRACT .....</b>	viii
<b>DAFTAR ISI.....</b>	ix
<b>DAFTAR TABEL .....</b>	xii
<b>DAFTAR GAMBAR .....</b>	xiii
<b>BAB 1 PENDAHULUAN .....</b>	1
<b>1. 1 Latar Belakang .....</b>	1
<b>1. 2 Rumusan Masalah.....</b>	2
<b>1. 3 Batasan Masalah.....</b>	2
<b>1. 4 Tujuan Penelitian.....</b>	3
<b>1. 5 Manfaat penelitian .....</b>	3
<b>1. 6 Metodologi Penelitian.....</b>	3
<b>1. 7 Penelitian Relevan .....</b>	4
<b>1. 8 Sistematika Penulisan .....</b>	7
<b>BAB 2 LANDASAN TEORI .....</b>	8
<b>2. 1 INTERNET OF THINGS (IOT) .....</b>	8
<b>2. 2 Machine Learning .....</b>	8
<b>2. 3 <i>You Only Look Once</i> (YOLO) .....</b>	9
<b>2. 4 ESP 32 – CAM .....</b>	11
<b>BAB 3 ANALISIS DAN PERANCANGAN.....</b>	12
<b>3. 1 Analisis Sistem .....</b>	12
<b>3. 1. 1 Analisis Masalah .....</b>	12

3. 1. 2	Analisis proses .....	12
3. 1. 3	Arsitektur Umum .....	13
<b>3.2</b>	<b>Pemodelan Sistem</b> .....	<b>14</b>
3.2.1	<i>Use Case Diagram</i> .....	14
3.2.2	<i>Activity Diagram</i> .....	15
<b>3.3</b>	<b>Flowchart</b> .....	<b>17</b>
3.3.1	<i>Flowchart</i> Sistem .....	17
3.3.2	<i>Flowchart</i> Model YOLO .....	19
<b>3.4</b>	<b>Perancangan Sistem</b> .....	<b>20</b>
3.4.1	Perancangan Model .....	20
3.4.2	Perancangan Alat .....	21
3.4.3	Perancangan Interface .....	22
<b>BAB 4 IMPLEMENTASI DAN PENGUJIAN SISTEM</b> .....		<b>25</b>
<b>4.1</b>	<b>Implementasi sistem</b> .....	<b>25</b>
4.1.1	Pengujian Sistem Spesifikasi Perangkat Keras .....	25
4.1.2	Spesifikasi Perangkat Lunak .....	25
<b>4.2</b>	<b>Pengumpulan <i>Dataset</i></b> .....	<b>25</b>
<b>4.3</b>	<b>Pelabelan <i>Dataset</i></b> .....	<b>28</b>
<b>4.4</b>	<b><i>Preprocessing, Augmentasi dan split dataset</i></b> .....	<b>30</b>
<b>4.5</b>	<b>Training Dataset</b> .....	<b>31</b>
4.5.1	Mengunduh Library Ultralytics.....	31
4.5.2	<i>Import</i> YOLO .....	31
4.5.3	Inisialisasi Model YOLO .....	31
4.5.4	Proses Mengakses <i>Dataset</i> .....	32
4.5.5	Proses Mendefinisikan Jumlah <i>Class</i> pada <i>Dataset</i> .....	32
4.5.6	Proses <i>Training Dataset</i> .....	33

4.5.7	<i>Tuning Hyperparameter</i> .....	33
<b>4.6</b>	<b>Evaluasi Model</b> .....	37
<b>4.7</b>	<b>Proses Deteksi Model YOLO</b> .....	43
4.7.1	<i>Resize</i> citra .....	43
4.7.2	Arsitektur Yolov8 .....	46
<b>4.8</b>	<b>Pengujian Sistem</b> .....	51
4.8.1	Persiapan Pengujian sistem .....	51
4.8.2	Hasil Pengujian Sistem .....	53
<b>BAB 5 KESIMPULAN DAN SARAN</b> .....		62
<b>5.1</b>	<b>Kesimpulan</b> .....	62
<b>5.2</b>	<b>Saran</b> .....	62
<b>DAFTAR PUSTAKA</b> .....		64

**DAFTAR TABEL**

<b>Tabel 4. 1</b> Training Epoch 100, Learning Rate 0.01 .....	34
<b>Tabel 4. 2</b> Summary Training Epoch 100, Learning Rate 0.01 .....	34
<b>Tabel 4. 3</b> Training Epoch 150, Learning Rate 0.01 .....	35
<b>Tabel 4. 4</b> Summary Training Epoch 150, Learning Rate 0.01 .....	35
<b>Tabel 4. 5</b> Training Epoch 200, Learning Rate 0.01 .....	36
<b>Tabel 4. 6</b> Summary Training Epoch 200, Learning Rate 0.01 .....	36
<b>Tabel 4. 7</b> Layer pada Model YOLOv8 .....	46

## DAFTAR GAMBAR

<b>Gambar 2. 1</b> Bounding Box .....	9
<b>Gambar 2. 2</b> Intersection Over Unit ( <a href="http://www.gabormelli.com/RKB">www.gabormelli.com/RKB</a> ) .....	10
<b>Gambar 2. 3</b> ESP 32-CAM .....	11
<b>Gambar 3. 1</b> Arsitektur Umum .....	13
<b>Gambar 3. 2</b> Use Case Diagram .....	15
<b>Gambar 3. 3</b> Activity Diagram .....	16
<b>Gambar 3. 4</b> Flowchart .....	17
<b>Gambar 3. 5</b> Flowchart Model YOLO .....	19
<b>Gambar 3. 6</b> Perancangan Model .....	20
<b>Gambar 3. 7</b> Perancangan Alat .....	21
<b>Gambar 3. 8</b> Perancangan Interface .....	22
<b>Gambar 3. 9</b> Output Total Kendaraan .....	23
<b>Gambar 3. 10</b> Output Status Lalu Lintas .....	23
<b>Gambar 3. 11</b> Output Durasi Lampu Lalu Lintas .....	24
<b>Gambar 4. 1</b> Contoh dataset dari ATCS Medan .....	26
<b>Gambar 4. 2</b> Contoh dataset dari gawai peneliti .....	27
<b>Gambar 4. 3</b> Pelabelan Kelas mobilis, motors, truks .....	28
<b>Gambar 4. 4</b> Proses persiapan dataset .....	29
<b>Gambar 4. 5</b> Proses Preprocessing, Augmentasi dan Split Dataset .....	30
<b>Gambar 4. 6</b> Kode untuk Mengunduh Library Ultralytics .....	31
<b>Gambar 4. 7</b> Kode untuk Import YOLO .....	31
<b>Gambar 4. 8</b> Kode untuk menginisialisasi Model YOLO .....	31
<b>Gambar 4. 9</b> Kode untuk Mengakses Dataset .....	32
<b>Gambar 4. 10</b> File data.yaml .....	32
<b>Gambar 4. 11</b> Kode untuk Mendefinisikan Jumlah Class pada Dataset .....	32
<b>Gambar 4. 12</b> Kode untuk Training Dataset .....	33
<b>Gambar 4. 13</b> Kode untuk Training Dataset .....	37
<b>Gambar 4. 14</b> Grafik Box Loss .....	38
<b>Gambar 4. 15</b> Grafik Classification Loss .....	38
<b>Gambar 4. 16</b> Grafik Distribution Focal Loss .....	39
<b>Gambar 4. 17</b> F1-Confidence score model .....	40

<b>Gambar 4. 18</b> Confusion matrix .....	41
<b>Gambar 4. 19</b> Gambar Input .....	44
<b>Gambar 4. 20</b> Gambar setelah Resize 640 x 640 .....	44
<b>Gambar 4. 21</b> Grid Cell 7 x 7 .....	45
<b>Gambar 4. 22</b> Arsitektur YOLO .....	46
<b>Gambar 4. 23</b> Pencarian Bounding Box .....	49
<b>Gambar 4. 24</b> Prediksi Bounding Box .....	49
<b>Gambar 4. 25</b> Prediksi IOU .....	50
<b>Gambar 4. 26</b> Modul ESP32-CAM .....	51
<b>Gambar 4. 27</b> Pengambilan kondisi lalu lintas .....	52
<b>Gambar 4. 28</b> Output dari sistem .....	53
<b>Gambar 4. 29</b> Hasil Uji 1 .....	54
<b>Gambar 4. 30</b> Hasil Uji 2 .....	54
<b>Gambar 4. 31</b> Hasil Uji 3 .....	55
<b>Gambar 4. 32</b> Hasil Uji 4 .....	55
<b>Gambar 4. 33</b> Hasil uji 5 .....	56
<b>Gambar 4. 34</b> Hasil Uji 6 .....	56
<b>Gambar 4. 35</b> Hasil Uji 7 .....	57
<b>Gambar 4. 36</b> Hasil Uji 8 .....	57
<b>Gambar 4. 37</b> Hasil Uji 9 .....	58
<b>Gambar 4. 38</b> Hasil Uji 10 .....	58
<b>Gambar 4. 39</b> Hasil Sampel 1 .....	59
<b>Gambar 4. 40</b> Hasil Sampel 2 .....	59
<b>Gambar 4. 41</b> Hasil Sampel 3 .....	60

## BAB 1

### PENDAHULUAN

#### 1.1 Latar Belakang

Sistem transportasi di kawasan perkotaan di seluruh dunia saat ini dihadapkan dengan tantangan signifikan berupa kemacetan lalu lintas. Fenomena kemacetan ini tidak hanya menyebabkan pemborosan waktu dan energi, tetapi juga berdampak luas mencakup aspek ekonomi, sosial, dan lingkungan yang signifikan. Urbanisasi yang pesat serta peningkatan populasi di kota-kota besar telah memperburuk kondisi ini, sehingga kemacetan lalu lintas menjadi semakin parah. Akibatnya, para pengguna jalan mengalami tingkat frustrasi yang tinggi dan risiko kecelakaan lalu lintas meningkat secara signifikan (HermaTelkomwan et al., 2021).

Salah satu cara untuk mencegah permasalahan kompleks ini, dengan mengembangkan sistem yang dapat mengatur lalu lintas berdasarkan kepadatan kendaraan di setiap persimpangan. Sistem ini memanfaatkan teknologi deteksi objek untuk menghitung jumlah kendaraan yang melewati titik-titik tertentu. Salah satu metode yang digunakan dalam sistem ini adalah *You Only Look Once* (YOLO), sebuah teknik canggih dalam jaringan saraf konvolusional (CNN) yang mampu melakukan pengenalan objek dalam waktu yang singkat dan akurat. Metode YOLO akan menganalisa objek dari gambar selama proses pelatihan dengan menggunakan model terpadu, yang memungkinkan satu jaringan konvolusional untuk memprediksi kotak pembatas dan probabilitas dalam waktu yang bersamaan (Ren et al., 2017).

Penelitian terdahulu menunjukkan bahwa sistem pemantauan lalu lintas berbasis YOLO telah diimplementasikan, namun aplikasi ini umumnya terbatas pada deteksi kendaraan dalam gambar tanpa menghitung jumlah kendaraan yang ada. YOLO dikenal sebagai metode yang memiliki kecepatan dan akurasi tinggi dalam deteksi objek secara *real-time*, sehingga diharapkan dapat meningkatkan efisiensi dan efektivitas dalam pengaturan sistem lalu lintas. Dengan demikian, diharapkan tercipta ketertiban dalam berkendara serta pengurangan kemacetan lalu lintas secara signifikan (HermaTelkomwan et al., 2021).

Walaupun sudah banyak penelitian yang dilakukan mengenai deteksi kemacetan lalu lintas, metode YOLO terus menunjukkan perkembangan signifikan

dalam konteks ini. Keunggulan utama YOLO terletak pada kecepatan inferensinya, dimana YOLO hanya perlu melalui jaringan saraf konvolusional satu kali untuk mendeteksi objek. Kecepatan dalam merespons perubahan kondisi lalu lintas sangatlah penting dalam implementasi sistem ini, menjadikan YOLO sebagai metode yang sangat optimal untuk deteksi dan penghitungan objek baik secara *real-time* maupun *non-real-time*. (Rofii et al., 2021) Dengan potensi besar ini, YOLO diharapkan mampu memberikan peningkatan signifikan dalam manajemen lalu lintas di berbagai kota besar di seluruh dunia.

Secara keseluruhan, tantangan kemacetan lalu lintas yang terjadi pada kota besar merupakan suatu masalah kompleks yang memerlukan solusi inovatif. Penggunaan teknologi deteksi objek berbasis YOLO menawarkan pendekatan yang sangat menjanjikan untuk meningkatkan efisiensi pengaturan lalu lintas. Dengan kemampuannya yang cepat dan akurat, YOLO dapat membantu mengurangi kemacetan dan meningkatkan keselamatan jalan, yang pada akhirnya akan memberikan manfaat luas bagi masyarakat perkotaan.

## 1.2 Rumusan Masalah

Kemacetan lalu lintas merupakan permasalahan yang dihadapi pada kawasan perkotaan termasuk kota Medan. Lampu lalu lintas yang tidak mampu menyesuaikan durasi dengan keadaan pada jalan dapat menyebabkan penumpukan kendaraan. Oleh karena itu dibutuhkan sistem pendekripsi kemacetan yang dapat mendeteksi kemacetan berdasarkan jumlah dan kecepatan kendaraan guna mengatur lampu lalu lintas.

## 1.3 Batasan Masalah

Beberapa batasan masalah dalam penelitian ini sebagai berikut.

1. Sistem disimulasikan pada Jembatan Penyeberangan Orang (JPO) Jl. Sisingamangaraja, Kota Medan.
2. Pemrosesan dilakukan pada server lokal dengan pengujian pengambilan video *real-time* menggunakan modul ESP32-CAM dengan resolusi 2 MP.
3. Pendekripsi objek kendaraan menggunakan model YOLOv8.
4. Kelas yang diidentifikasi oleh model adalah mobil, sepeda motor dan truk.
5. Output berupa hasil video deteksi kemacetan lalu lintas.

#### **1.4 Tujuan Penelitian**

Penelitian ini bertujuan menerapkan algoritma YOLO dalam mendeteksi kemacetan lalu lintas secara *real-time*. Menggunakan model YOLO untuk menghitung jumlah dan kecepatan kendaraan pada jalan umum dalam upaya mencapai analisis kondisi nyata. Adanya penelitian ini diharapkan dapat membantu dalam langkah pencegahan terjadinya kemacetan lalu lintas yang disebabkan oleh penumpukan kendaraan pada persimpangan jalan.

#### **1.5 Manfaat penelitian**

Beberapa manfaat dalam penelitian ini sebagai berikut.

1. Mengoptimalkan durasi lampu lalu lintas berdasarkan volume kendaraan yang sebenarnya di setiap jalur. Hal ini membantu mengurangi waktu tunggu yang tidak perlu pada persimpangan jalan yang padat.
2. Mengurangi antrian kendaraan menumpuk pada suatu persimpangan jalan. Hal ini dapat mencegah kemacetan yang terjadi akibat penumpukan kendaraan.

#### **1.6 Metodologi Penelitian**

Berikut ini merupakan tahapan-tahapan dari metode penelitian yang dilaksanakan, yaitu:

1. Studi Pustaka  
Studi pustaka dilakukan untuk mengumpulkan informasi dan landasan teori melalui literatur yang relevan, termasuk jurnal, buku, dan sumber data dari internet yang mendukung pengembangan sistem ini.
2. Pengumpulan Data Set  
Mengumpulkan data berupa video dalam format MP4 dan MOV. Data tersebut terdiri dari rekaman video lalu lintas kendaraan di lampu lalu lintas di Kota Medan, yang digunakan untuk analisis lebih lanjut dan pengembangan sistem pemantauan lalu lintas yang lebih efisien.
3. Pelabelan Data  
Pelabelan data pada dataset dilakukan untuk mengidentifikasi dan menandai objek kendaraan yang digunakan dalam proses deteksi kepadatan lalu lintas.

#### 4. *Training*

Setelah pelabelan data dilakukan, dataset tersebut digunakan untuk melatih model guna mendapatkan hasil yang diinginkan.

#### 5. Evaluasi

Pada tahap evaluasi model, kinerja YOLOv8 dievaluasi menggunakan metrik-metrik seperti akurasi, *recall*, dan presisi. Tujuan dari langkah ini adalah untuk memverifikasi bahwa model mampu menghasilkan identifikasi yang akurat dan efisien.

#### 6. Implementasi Sistem

Model yang telah dievaluasi kemudian diimplementasikan ke dalam sistem yang terintegrasi dengan infrastruktur *Internet of Things* (IoT).

#### 7. Penyusunan Laporan

Penyusunan laporan akhir penelitian ini merangkum seluruh aspek penting yang meliputi identifikasi permasalahan yang dihadapi, penerapan metode penelitian yang digunakan, presentasi hasil yang diperoleh, dan analisis mendalam atas temuan yang relevan, semuanya disajikan dalam format skripsi akademik..

### 1.7 Penelitian Relevan

Penelitian ini dilakukan dengan melihat beberapa penelitian yang relevan terdahulu, antara lain :

1. (HermaTelkomwan et al., 2021) Penelitian ini bertujuan untuk mengatasi masalah kemacetan di persimpangan jalan pada jam sibuk di kota-kota padat penduduk dengan mengusulkan penerapan lampu lalu lintas yang diatur berdasarkan kepadatan kendaraan. Untuk mencapai tujuan tersebut, penelitian ini menggabungkan algoritma YOLO dan SORT. Algoritma YOLO, yang merupakan algoritma deteksi objek yang dikembangkan dari Convolutional Neural Network (CNN), dipilih karena tingkat akurasi dan kecepatan frame rate yang tinggi. Evaluasi dilakukan menggunakan dataset dengan parameter kinerja seperti Average Precision (AP) dan akurasi. Hasil penelitian menunjukkan bahwa model YOLOv3 mampu mendekripsi dengan baik, mencapai nilai AP tertinggi sebesar 0,89. Selain itu, pengaturan model yang tepat memberikan akurasi pengujian tertinggi sebesar 98,80%.

Meskipun telah diterapkan model dengan skema yang berbeda untuk menghindari overfitting, hasil pengujian menunjukkan tetap tingginya tingkat akurasi. Dengan demikian, penelitian ini memberikan solusi yang efektif dalam mengatasi masalah kemacetan dengan memanfaatkan teknologi terbaru dalam bidang pengenalan objek dan pengolahan citra.

2. (Rofii et al., 2021) Mencari slot parkir yang tersedia sering menjadi tantangan bagi pengemudi karena tingkat kepadatan area parkir di dalamnya. Sistem parkir memanfaatkan lahan dan menghemat waktu. Oleh karena itu, sistem ini dikembangkan untuk mendeteksi jumlah slot parkir dengan akurasi tinggi, memudahkan pengelolaan lahan parkir, dan mempercepat pencarian parkir oleh pengguna. Penggunaan pemrograman GPU yang digabungkan dengan M-YOLO dalam sistem ini memungkinkan pengolahan citra dan data secara paralel untuk mendeteksi mobil dan jumlahnya dengan akurasi yang tinggi. Hasil uji coba menunjukkan bahwa penggunaan GPU, dibandingkan dengan CPU, dapat mengurangi waktu komputasi, dengan tingkat akurasi mencapai 100%. Pengujian lebih lanjut dengan variasi kendaraan dan bentuk slot parkir yang lebih banyak disarankan untuk penelitian selanjutnya, serta pengembangan algoritma M-YOLO dengan CUDA untuk mencapai hasil baik dalam parkir yang lebih kompleks..
3. (Kusuma et al., 2021) Untuk mengatasi proses penghitungan jumlah orang di transportasi umum yang sering dilakukan secara manual menggunakan alat hitung tangan, penelitian ini memanfaatkan teknologi pengolahan citra, khususnya dengan metode YOLO. Fokus utama penelitian ini adalah pengembangan sistem yang dapat secara otomatis menghitung jumlah orang dalam gambar, mengeliminasi kebutuhan akan penghitungan manual. Pengujian dilakukan dengan menggunakan metode YOLOv4 dan transfer learning untuk satu kelas, dibagi menjadi tiga tahap pelatihan data dan dua tahap pengujian, dengan menganalisis akurasi, presisi, recall, F1 score, IoU, dan mAP. Hasil penelitian menunjukkan bahwa konfigurasi terbaik mencakup learning rate sebesar 0.001, random value 0, dan sub divisions 32. Dengan akurasi sistem mencapai 69%, dengan mAP sebesar 72.68%, presisi 77%, dan

- rata-rata IoU 62.88%. Namun, evaluasi juga mengungkapkan kesulitan dalam mendekripsi wajah dengan masker dan topi.
4. (Usen & Hayat, 2023) Umumnya, data kendaraan dikumpulkan dari berbagai wilayah dan disatukan menjadi satu dataset untuk menganalisis pertumbuhan kendaraan di suatu negara. Proses pengumpulan data ini umumnya dilakukan secara manual dan memakan waktu yang cukup lama. Namun, penelitian ini mengusulkan pendekatan baru dengan menggunakan teknologi pengolahan citra berbasis Android untuk mendekripsi plat kendaraan. Mereka memilih metode YOLO v4 karena kemampuannya dalam melatih data secara langsung dan memberikan hasil prediksi yang cepat. Hasil pengujian menunjukkan akurasi Secara rata-rata, sistem ini memiliki akurasi sebesar 86.82%, dengan berbagai parameter lainnya seperti presisi, recall, F1 score, IoU, dan mAP. Hasilnya menunjukkan bahwa sistem baik dalam mendekripsi plat kendaraan melalui . Berdasarkan penelitian ini, dapat disimpulkan bahwa sistem ini memiliki kinerja yang baik dan dapat digunakan secara efektif dalam mendekripsi plat kendaraan.
  5. (Indra et al., 2023) Penelitian ini mengembangkan metode untuk mendekripsi, mengklasifikasikan, dan menghitung jumlah kendaraan menggunakan model YOLOv4 berbasis visi komputer dan kecerdasan buatan. Setelah kendaraan terdeteksi dan diklasifikasikan, sistem memberikan nomor identifikasi unik serta menampilkan Data input diperoleh dari dataset video yang memperhitungkan posisi kamera, cahaya, dan kepadatan lalu lintas. Metode ini berhasil menghitung jumlah kendaraan. Evaluasi dilakukan dengan mempertimbangkan akurasi, presisi, dan recall total berdasarkan confusion matrix. Hasil pengujian menunjukkan performa terbaik pada kondisi siang hari , mencapai akurasi, presisi, dan recall total sebesar 83%, 93%, dan 94%. Namun, kondisi malam hari memberikan hasil terendah, yaitu akurasi, presisi, dan recall total sebesar 68%, 77%, dan 78%. Meskipun berhasil mengimplementasikan detektor objek YOLOv4, penelitian ini memiliki kekurangan seperti kurangnya pembatasan area perhitungan dan belum adanya pengujian real-time serta belum mempertimbangkan sumbu kendaraan sesuai standar dinas perhubungan.

## 1. 8 Sistematika Penulisan

Penelitian ini mengimplementasikan metode penulisan dengan langkah-langkah sebagai berikut.

### BAB 1 PENDAHULUAN

Dalam bab ini, kita membahas latar belakang penelitian, perumusan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, penelitian yang relevan, serta sistematika penulisan dalam skripsi ini.

### BAB 2 LANDASAN TEORI

Pada bab ini berisi tentang pembahasan pengertian *Internet Of Things* (IOT), *Machine Learning*, *You Only Look Once* (YOLO).

### BAB 3 ANALISIS PERANCANGAN

Bagian ini merincikan analisis yang telah dilakukan terhadap permasalahan yang ada serta merancang sistem sebagai solusinya. Ini mencakup pemahaman yang mendalam terhadap masalah yang dihadapi dan pembuatan diagram alir yang menggambarkan secara visual operasi sistem yang direncanakan.

### BAB 4 IMPLEMENTASI DAN PENGUJIAN

Pada bagian ini, dijabarkan langkah-langkah penerapan sistem yang telah disusun sebelumnya dan proses pengujian sistem untuk mendapatkan analisis kinerja yang terperinci. Hal ini mencakup eksekusi implementasi sistem yang telah direncanakan sebelumnya, bersama dengan serangkaian uji coba yang dilakukan untuk mengevaluasi kinerja sistem tersebut.

### BAB 5 KESIMPULAN DAN SARAN

Bagian ini memberikan kesimpulan berdasarkan analisis dalam bab-bab sebelumnya. Kesimpulan tersebut mencakup poin-poin kunci yang telah dibahas serta temuan utama yang diperoleh selama penelitian. Selanjutnya, disampaikan saran yang dapat digunakan oleh peneliti untuk penelitian mendatang, yang dapat mendukung pengembangan lebih lanjut atau perbaikan terhadap topik yang telah diteliti.

## **BAB 2**

### **LANDASAN TEORI**

#### **2. 1 INTERNET OF THINGS (IOT)**

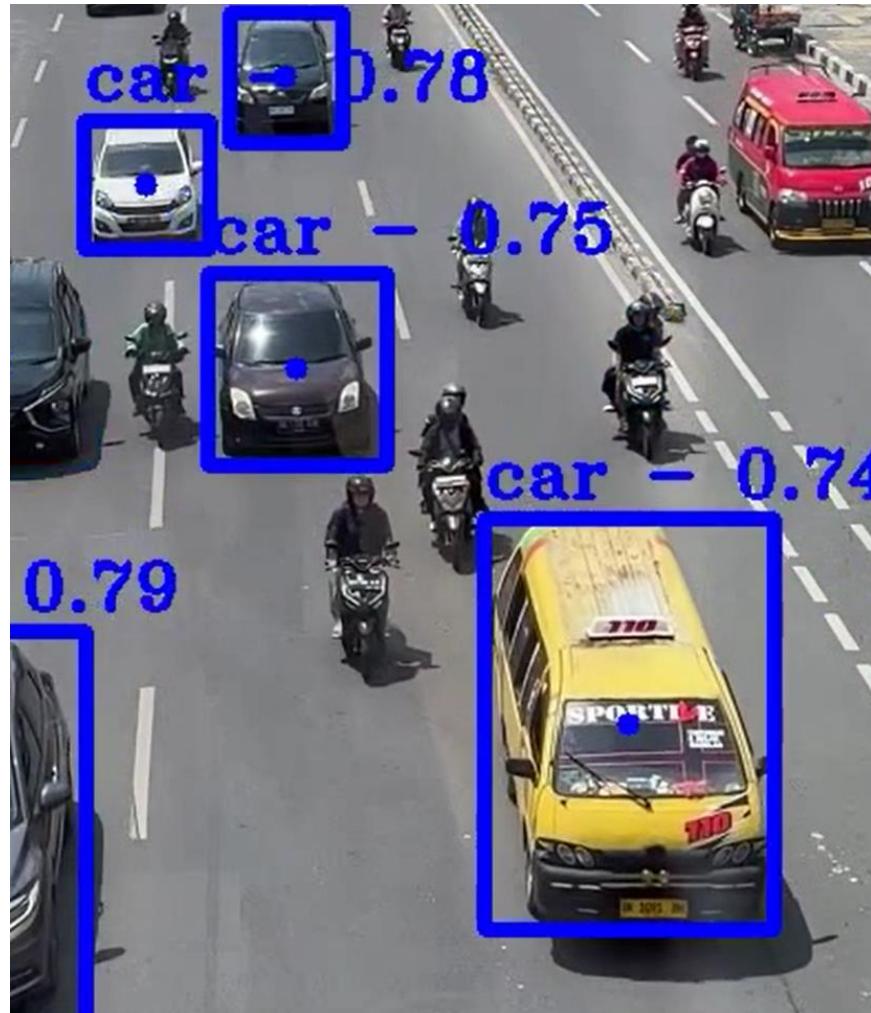
Menurut (Iswahyudi, 2021), IoT merupakan sebuah konsep yang bertujuan untuk memperluas manfaat yang diberikan oleh internet yang selalu tersambung. peradaban memasuki era tidak hanya ponsel pintar atau komputer yang mempunyai kemampuan untuk terhubung ke internet. IoT memungkinkan berbagai macam perangkat fisik, mulai dari peralatan rumah tangga hingga sistem industri, untuk berkomunikasi dan bertukar data melalui jaringan internet. Inovasi ini membuka peluang besar untuk meningkatkan efisiensi, mengoptimalkan penggunaan sumber daya, dan menciptakan pengalaman pengguna yang lebih baik melalui otomatisasi dan analisis data yang canggih.

IoT memungkinkan berbagai macam objek di dunia nyata berkomunikasi sebagai satu sistem terpadu, menggunakan jaringan internet sebagai penghubung. Pada dasarnya, perangkat IoT terdiri dari sensor yang mengumpulkan data, internet sebagai media penyedia komunikasi, dan server yang menganalisis informasi yang telah diterima dari sensor (Selay & arief, 2022).

#### **2. 2 Machine Learning**

Menurut (Vishwakarma, 2024) Pembelajaran mesin (ML) bertujuan untuk mengajarkan mesin cara untuk mengolah data dengan lebih efisien. Tujuan utama pembelajaran mesin adalah mempelajari data. Banyak penelitian telah dilakukan untuk membuat mesin belajar secara mandiri tanpa harus diprogram secara spesifik. Matematika dan pemrograman menggunakan berbagai pendekatan untuk menemukan solusi atas masalah yang melibatkan dataset besar. Pembelajaran mesin mengandalkan berbagai algoritma untuk menyelesaikan berbagai masalah.

### 2.3 You Only Look Once (YOLO)



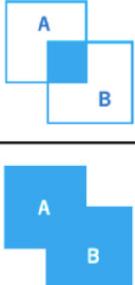
**Gambar 2. 1 Bounding Box**

YOLO adalah algoritma yang dirancang untuk mendeteksi objek. Metode deteksi ini memodifikasi classifier untuk tujuan deteksi objek. YOLO menggunakan Jaringan Saraf Tiruan untuk mendeteksi objek dalam gambar dengan membaginya menjadi beberapa wilayah, kemudian memprediksi kotak pembatas dan probabilitas untuk setiap wilayah (Redmon, 2016). Kotak pembatas ini kemudian dievaluasi berdasarkan probabilitas yang diprediksi.

Secara umum, klasifikasi adalah proses mengidentifikasi label dari data yang diuji. Namun, dalam YOLO, klasifikasi juga mencakup lokalisasi, yaitu menentukan lokasi objek dalam bentuk kotak pembatas ( $bx$ ,  $by$ ,  $bh$ ,  $bw$ ) (Rofii, 2021).

YOLOv8 menggunakan model terpisah untuk memproses objek, klasifikasi, dan tugas regresi secara mandiri. Desain ini memungkinkan setiap cabang untuk fokus pada tugasnya dan meningkatkan akurasi model secara keseluruhan. Pada lapisan *output* YOLOv8, mewakili probabilitas bahwa kotak pembatas berisi suatu objek. probabilitas kelas mewakili probabilitas objek yang termasuk dalam setiap kelas. (Terven et al., 2023).

Nilai kepercayaan (*confidence score*) yang dihasilkan YOLO pada setiap kotak pembatas yang merepresentasikan probabilitas bahwa kotak tersebut berisi objek. Nilai kepercayaan ini kemudian dikalikan dengan IoU (*Intersection over Union*) antara kotak prediksi dan kotak ground truth. IoU adalah metrik yang mengukur tingkat akurasi deteksi objek dalam suatu model. Proses perhitungan nilai kepercayaan untuk setiap kotak dijelaskan melalui gambar dan rumus di bawah ini, yang menetapkan hubungan antara probabilitas antara kotak prediksi dan kotak kebenaran dasar.

$$\text{Intersection over Unit (IoU)} = \frac{\text{Area of overlap}}{\text{Area of union}}$$


$$= \frac{|A \cap B|}{|A \cup B|}$$

**Gambar 2.2** *Intersection Over Unit* ([www.gabormelli.com/RKB](http://www.gabormelli.com/RKB))

## 2. 4 ESP 32 – CAM



**Gambar 2. 3 ESP 32-CAM**

mikrokontroler ESP32 serta kamera ini menawarkan berbagai fitur yang dapat dimanfaatkan oleh pengguna. Salah satu fitur unggulan yang tersedia adalah kemampuan untuk mengambil gambar. komponen ini dapat dikonfigurasi menggunakan editor Arduino IDE dan dapat memanfaatkan berbagai pustaka atau fitur yang sudah tersedia. ESP32-Cam, sebagai komponen serbaguna untuk berbagai proyek. Komponen ini memiliki mikrokontroler terintegrasi yang dapat beroperasi secara mandiri. Selain itu, komponen ini dilengkapi dengan konektivitas WiFi dan *Bluetooth*, serta kamera dan slot microSD untuk penyimpanan data.(Arrahma & Mukhaiyar, 2023)

## **BAB 3**

### **ANALISIS DAN PERANCANGAN**

#### **3. 1 Analisis Sistem**

Analisis sistem adalah proses mendetail yang bertujuan untuk memahami, merencanakan, dan mengevaluasi sistem yang dikembangkan atau dianalisis dalam skripsi.

##### **3. 1. 1 Analisis Masalah**

Analisis masalah adalah proses sistematis untuk mengidentifikasi, memahami, dan mendefinisikan masalah utama yang ingin dipecahkan oleh sistem agar dapat berfungsi sesuai kebutuhan secara efisien dan efektif. Dalam konteks pengaturan lalu lintas, ini melibatkan pemahaman mendalam tentang isu-isu seperti kemacetan, waktu tunggu, serta tantangan dalam menerapkan algoritma YOLO untuk deteksi visual berbasis Machine Learning. Tantangan utama termasuk memastikan akurasi dan kecepatan deteksi objek dalam kondisi lalu lintas yang dinamis, serta integrasi yang baik antara perangkat keras dan perangkat lunak. Kamera harus ditempatkan secara strategis, sementara sistem harus memiliki kapasitas komputasi yang memadai untuk memproses data. Aspek teknis dan non-teknis seperti biaya implementasi, pemeliharaan, dan pengujian sistem juga harus dipertimbangkan, bersama dengan dampak lingkungan dan sosial untuk memastikan solusi yang berkelanjutan dan diterima oleh masyarakat. Dengan analisis masalah yang komprehensif, pengembangan sistem pengaturan lampu lalu lintas berbasis algoritma YOLO dapat menjadi lebih efektif dalam mengatasi masalah lalu lintas dan beradaptasi dengan tantangan serta perubahan di masa depan.

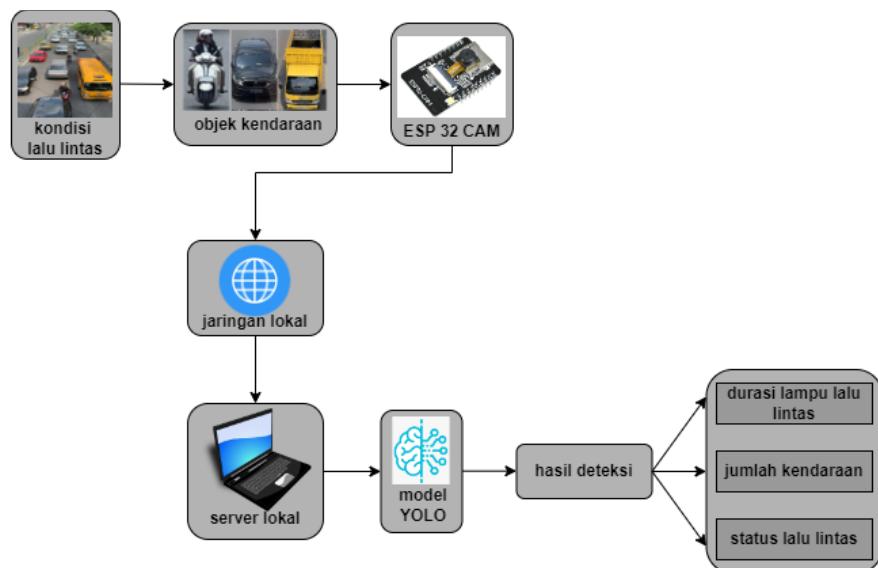
##### **3. 1. 2 Analisis proses**

Analisis proses mencakup identifikasi masalah kemacetan lalu lintas dan tujuan penelitian untuk mengembangkan sistem deteksi kemacetan real-time menggunakan algoritma YOLO. Studi literatur dilakukan untuk memahami Model YOLO dan relevansinya dalam manajemen lalu lintas. Desain sistem melibatkan perancangan arsitektur yang mengintegrasikan sensor kamera dan perangkat lunak untuk pengumpulan data secara efektif dan efisien. Data lalu lintas dikumpulkan secara terus-menerus melalui kamera, yang kemudian diolah oleh algoritma YOLO untuk mengidentifikasi dan menghitung kendaraan serta menentukan tingkat

kemacetan. Tantangan utama meliputi memastikan akurasi dan kecepatan deteksi objek dalam kondisi lalu lintas yang beragam. Pengujian dan evaluasi sistem dilakukan dengan mensimulasikan berbagai skenario lalu lintas untuk menilai kinerja dan akurasi, membandingkan hasil deteksi dengan data aktual, serta mengidentifikasi area yang memerlukan perbaikan. Analisis hasil memberikan wawasan tentang efektivitas sistem dan rekomendasi untuk pengembangan lebih lanjut. Seluruh proses penelitian didokumentasikan secara komprehensif dalam laporan skripsi, yang mencakup latar belakang, metodologi, hasil, dan kesimpulan, untuk memastikan bahwa penelitian ini dapat memberikan kontribusi signifikan dalam mengatasi masalah kemacetan di kota Medan melalui solusi inovatif.

### 3. 1. 3 Arsitektur Umum

Arsitektur umum mengacu pada cara suatu sistem dirancang dan bagaimana komponen-komponennya saling berinteraksi untuk mencapai tujuan tertentu. Berikut adalah desain arsitektur umum dari sistem ini:



**Gambar 3. 1 Arsitektur Umum**

Gambar di atas menggambarkan arsitektur umum dari sistem deteksi kemacetan untuk menentukan status lalu lintas dan pengaturan lampu lalu lintas menggunakan algoritma YOLO. Tahapan pertama adalah pengambilan video kondisi lalu lintas menggunakan kamera esp32 yang ditempatkan pada JPO Jalan Sisingamangaraja, Medan. Video yang diambil kemudian dipanggil oleh sistem melalui jaringan lokal yang berupa koneksi wi-fi yang sama yang berguna sebagai penghubung antara esp32 dengan server lokal untuk diproses. Kemudian, video kondisi lalu lintas

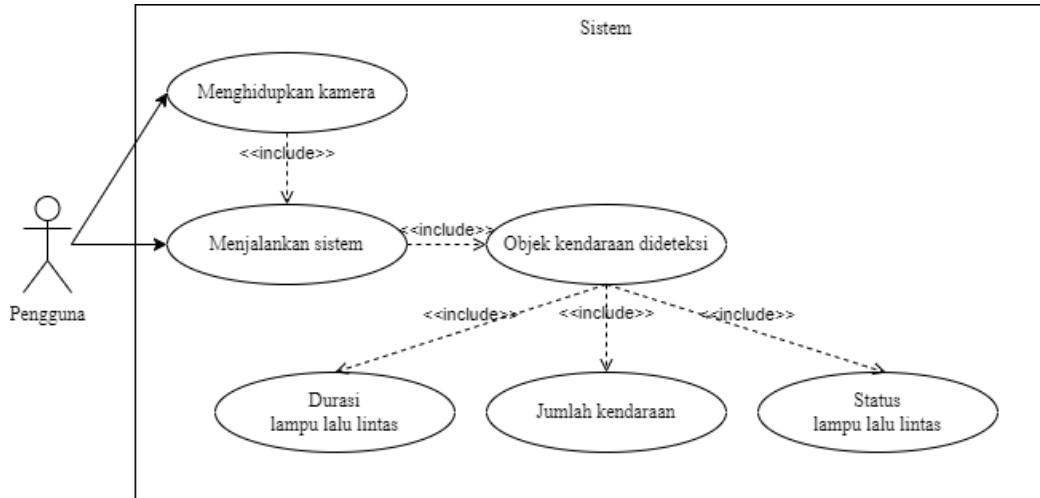
disimpan pada server lokal yang berupa laptop peneliti terlebih dahulu sebelum dipanggil oleh sistem untuk dideteksi objek-objek di dalam video menggunakan model YOLO. Setelah video dipanggil oleh Model YOLO, Model YOLO melakukan analisis mendalam untuk menghitung jumlah objek kendaraan dan mendeteksi kecepatan rata-rata dari kendaraan. Hasil dari analisis ini kemudian digunakan untuk memberikan durasi lampu lalu lintas yang ideal, guna mengoptimalkan aliran lalu lintas dan mengurangi kemacetan.

### **3.2 Pemodelan Sistem**

Pemodelan sistem melibatkan penjelasan rinci tentang bagaimana pengguna berinteraksi dengan aplikasi yang dikembangkan, sehingga memastikan sistem beroperasi secara optimal. Pemodelan ini menggunakan Unified Modeling Language (UML), sebuah bahasa pemodelan yang digunakan untuk menggambarkan hubungan antara komponen dalam sistem dan interaksinya dengan pengguna. Dalam penelitian ini, UML diterapkan melalui Use Case Diagram dan Activity Diagram. Use Case Diagram mengilustrasikan fungsionalitas sistem dari perspektif pengguna sementara Activity Diagram menggambarkan alur kerja dalam sistem. Dengan menggunakan kerangka UML ini, pemodelan sistem dilakukan secara terstruktur dan sistematis, memastikan bahwa semua aspek interaksi dan fungsionalitas sistem tercakup dengan baik. Pendekatan ini mendukung pengembangan aplikasi yang efisien dan efektif sesuai dengan kebutuhan pengguna.

#### **3.2.1 Use Case Diagram**

*Use case* diagram adalah alat pemodelan yang digunakan untuk menggambarkan interaksi antara pengguna dan sistem yang telah dirancang. Diagram ini mengilustrasikan bagaimana pengguna berinteraksi dengan sistem. use case diagram membantu dalam memahami suatu sistem serta memastikan bahwa kebutuhan pengguna dapat dipenuhi dengan baik dalam sistem tersebut.

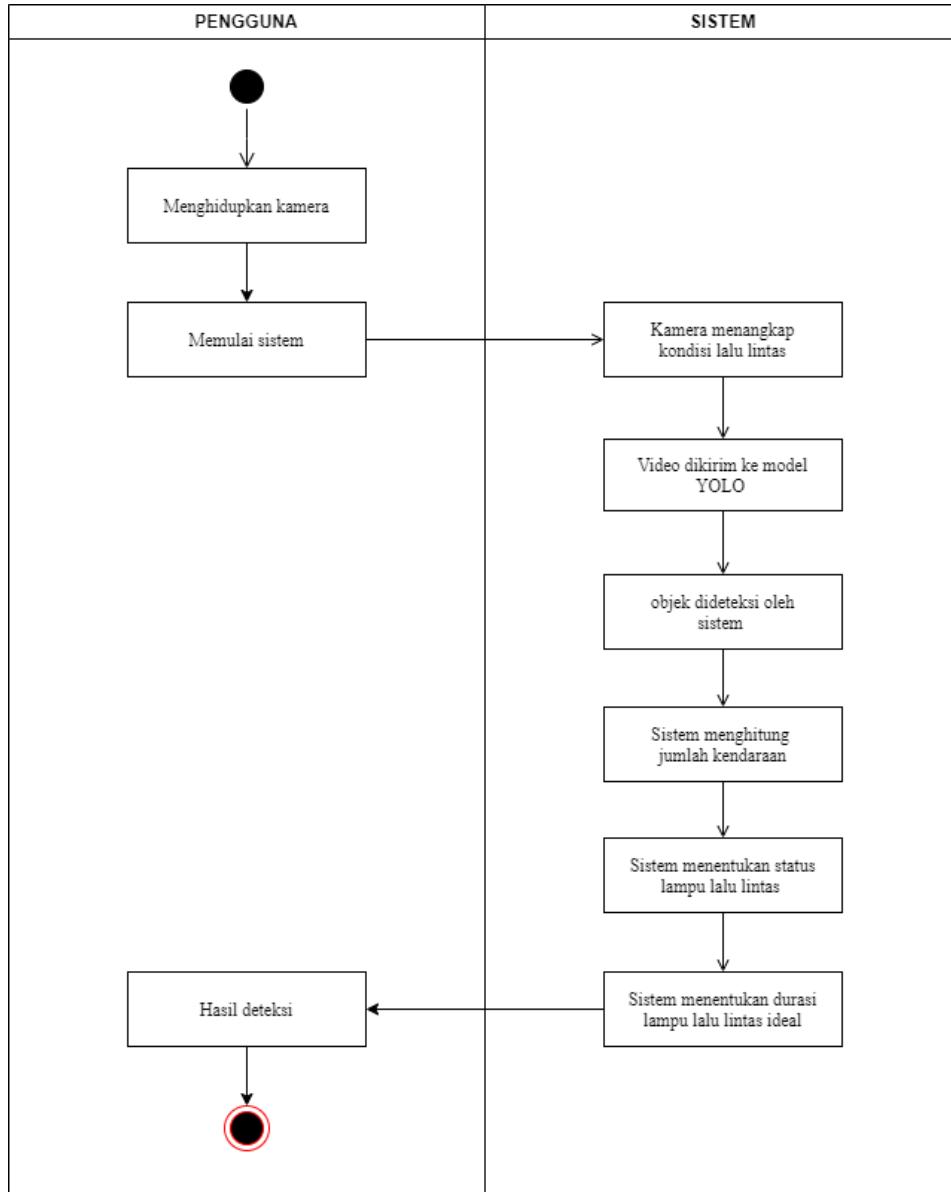


**Gambar 3. 2 Use Case Diagram**

Gambar 3.2 menggambarkan interaksi antara pengguna dan sistem. Pengguna memulai sistem dengan menghidupkan kamera. Setelah kamera dihidupkan pengguna menjalankan sistem terlebih dahulu. Sistem tidak dapat mendeteksi apabila kamera dalam keadaan tidak aktif. Setelah sistem dijalankan, sistem memanggil hasil tangkapan kamera. Objek berupa kendaraan dalam hasil tangkapan tersebut dideteksi oleh sistem. Setelah sistem mendeteksi objek kendaraan, sistem menampilkan hasil dari deteksinya. Berdasarkan hasil deteksinya ini, sistem menghitung jumlah kendaraan, menentukan status lalu lintas serta durasi lampu lalu lintas ideal sesuai kondisi lalu lintas yang telah dianalisa.

### 3.2.2 Activity Diagram

Activity Diagram adalah alat visual yang digunakan untuk menggambarkan alur kerja dan aktivitas sistem. Diagram ini mencatat langkah-langkah dan kegiatan yang berlangsung dari awal hingga akhir suatu proses atau aktivitas. Diagram ini menunjukkan urutan langkah-langkah, pengambilan keputusan, garis waktu, dan aliran kontrol dalam proses tersebut. Oleh karena itu, *Activity Diagram* mempermudah pemahaman secara visual tentang bagaimana suatu sistem atau proses berfungsi dan bagaimana elemen-elemen yang terlibat saling berinteraksi.



**Gambar 3. 3 Activity Diagram**

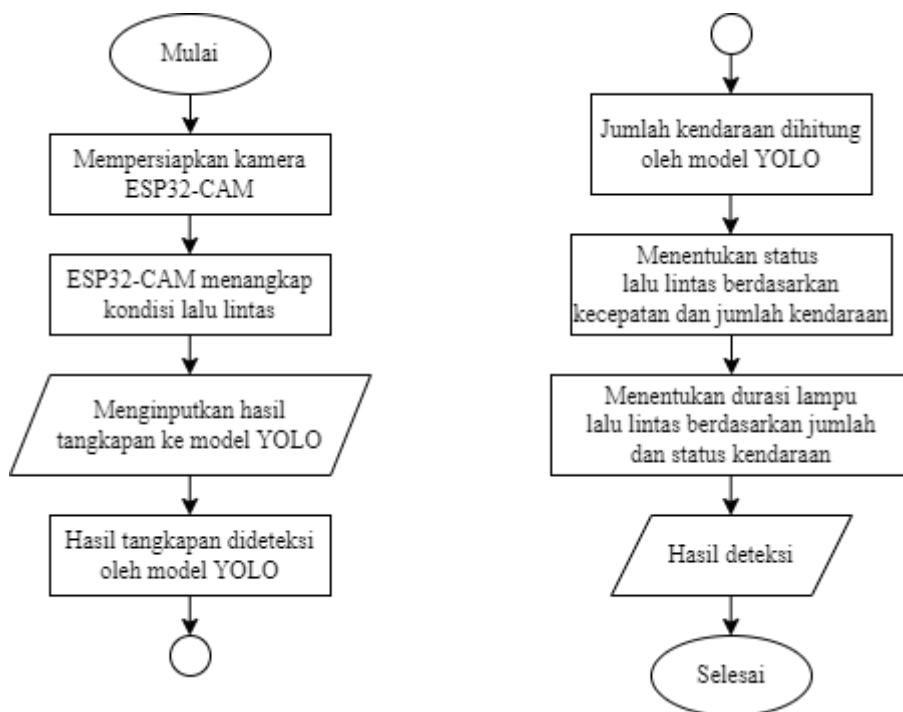
Diagram alir di atas menggambarkan tahapan dalam sistem deteksi dan pengaturan lalu lintas menggunakan model YOLO, yang dibagi menjadi dua bagian Pengguna dan Sistem. Pertama pengguna menghidupkan kamera terlebih dahulu. Kemudian setelah kamera diaktifkan maka pengguna dapat memasuki program dan kemudian memulai sistem pendekripsi. Setelah sistem dimulai, kamera yang telah aktif akan mengambil video kondisi lalu lintas saat itu. Video tersebut kemudian dipanggil oleh model YOLO untuk dideteksi. Pada model YOLO, video akan dideteksi objek-objek di dalamnya, seperti mobil, sepeda motor dan truk. Setelah deteksi selesai, sistem melakukan perhitungan kendaraan yang berada pada jalan. Setelah dideteksi

jumlah kendaraan dan kecepatan kendaraan, sistem menentukan status dari lalu lintas tersebut, yang juga digunakan untuk menghitung durasi lampu lalu lintas yang ideal sesuai dengan kondisi jalan tersebut. Hasil analisa dari sistem akan dikirimkan dalam bentuk video kepada pengguna. Berdasarkan hasil analisis ini, hasil deteksi dari sistem bertujuan untuk mengoptimalkan aliran lalu lintas, seperti mempercepat atau memperlambat durasi lampu hijau dan merah sesuai kebutuhan lalu lintas saat itu.

### 3.3 Flowchart

*Flowchart* adalah diagram yang memanfaatkan simbol-simbol grafis untuk menggambarkan tahapan-tahapan dalam suatu proses atau alur kerja. Simbol-simbol tersebut dihubungkan dengan panah yang menunjukkan urutan dan arah aliran. *Flowchart* mempermudah visualisasi proses secara sederhana, meningkatkan pemahaman, serta digunakan untuk menganalisis atau merancang sistem.

#### 3.3.1 Flowchart Sistem

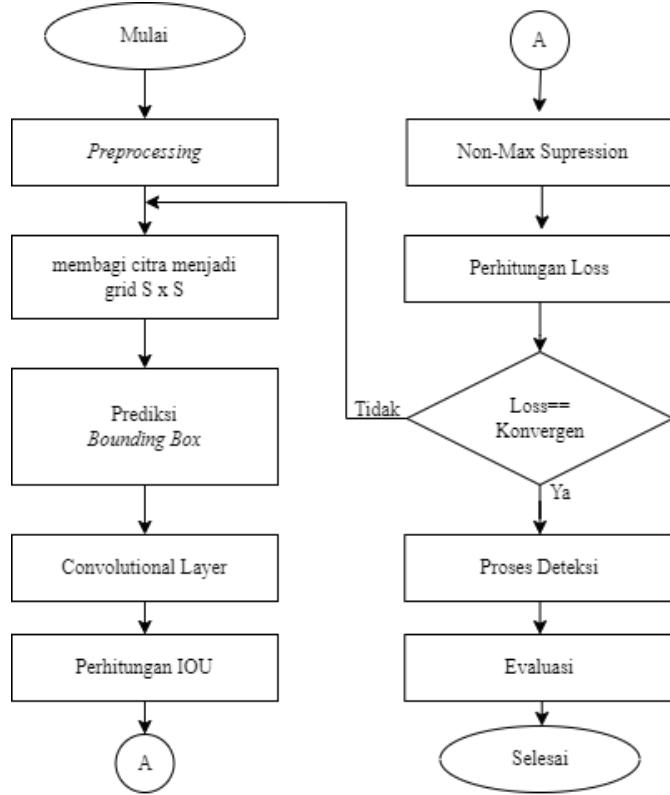


**Gambar 3.4** Flowchart Sistem

Diagram alir sistem ini menggambarkan tahapan operasional sistem secara rinci. Tahap pertama dimulai dengan mempersiapkan kamera pada JPO Jalan Sisingamangaraja, Medan. Kemudian kamera dihidupkan terlebih dahulu agar dapat menangkap kondisi lalu lintas pada jalan. Setelah kamera menangkap kondisi

lalu lintas, hasil dari tangkapan tersebut di-*inputkan* kepada model YOLO untuk di deteksi. Selanjutnya, sistem melakukan analisis pada gambar yang telah diambil untuk pada model YOLO yang bertujuan untuk mendeteksi jumlah kendaraan yang ada. Kemudian dilakukan juga analisa status lalu lintas dengan menghitung kecepatan dan jumlah kendaraan. Status lalu lintas ditentukan dengan kecepatan dan jumlah kendaraan yang terdeteksi. Untuk menghitung kecepatan rumus yang digunakan pada penelitian ini yaitu dengan mengukur jarak tempuh kendaraan tiba di kotak 1 dikurang waktu kendaraan tiba di kotak 2, kemudian jarak tersebut dibagi dengan waktu tempuh dalam meter per detik. Hasil dari perhitungan tersebut dikali dengan 3,6 untuk mendapatkan waktu dalam kilometer per jam. Kemudian dari kecepatan tersebut didapati status lalu lintas. Apabila kendaraan melaju kencang maka status lalu lintas adalah lancar, Apabila kendaraan melaju dibawah 20km/jam maka status lalu lintas adalah sedang. Untuk kondisi padat hanya dapat terpenuhi apabila kecepatan rata-rata kendaraan dibawah 10km/jam dan terdapat lebih dari 20 kendaraan yang terdeteksi. Setelah proses deteksi selesai, hasil analisis gambar tersebut digunakan untuk menghitung durasi lampu lalu lintas yang ideal guna mengoptimalkan lalu lintas. Perhitungan untuk durasi lampu lalu lintas yang digunakan sebagai rumus contoh pada simulasi penelitian ini adalah setiap mobil dan truk yang terdeteksi akan mengurangi durasi lampu merah selama 3 detik sedangkan pada sepeda motor hanya akan mengurangi durasi selama 1 detik setiap 3 sepeda motor yang terdeteksi. Informasi ini kemudian digunakan untuk mengatur durasi lampu lalu lintas sesuai dengan kondisi kepadatan yang terdeteksi. Setelah itu, sistem mengirimkan hasil dari deteksinya yang berupa video dengan jumlah kendaraan, status lalu lintas dan durasi yang ideal untuk lampu lalu lintas berdasarkan jumlah dan status kendaraan yang berada pada jalan.

### 3.3.2 Flowchart Model YOLO



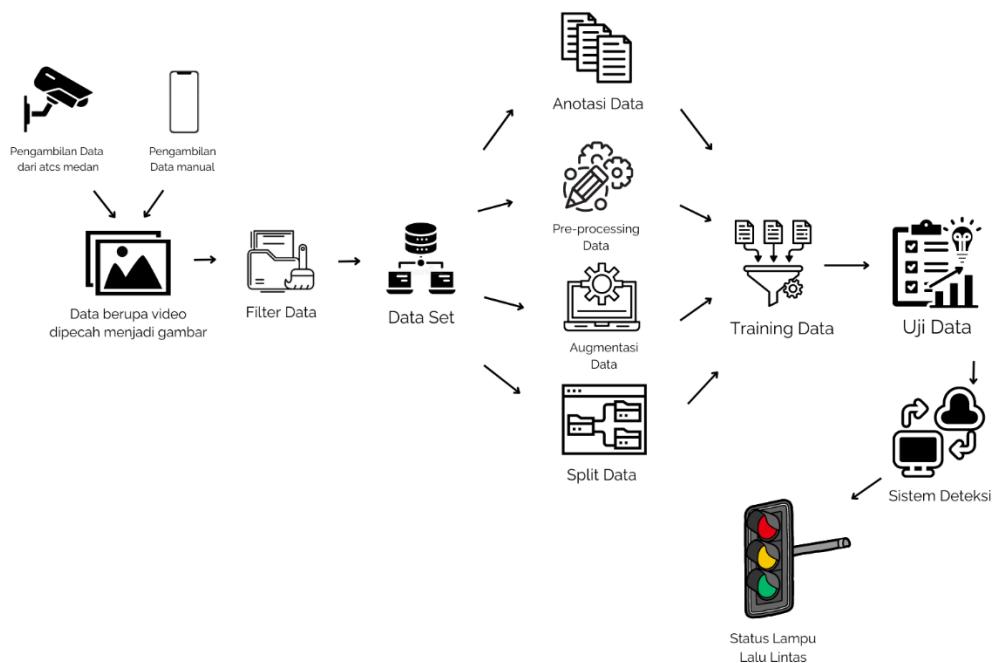
**Gambar 3. 5 Flowchart Model YOLO**

Diagram alur diatas merupakan diagram alur deteksi objek menggunakan metode YOLO yang dimulai dengan tahap preprocessing, di mana citra dipersiapkan untuk deteksi objek, termasuk normalisasi dan perubahan ukuran gambar. Citra kemudian dibagi menjadi grid berukuran  $S \times S$  untuk memudahkan deteksi di setiap grid. Setiap grid menghasilkan prediksi bounding box untuk menemukan objek di dalamnya. Fitur-fitur citra diekstraksi menggunakan konvolusi. Nilai *Intersection over Union* (IoU) antara *bounding box* prediksi dan *ground truth* dihitung untuk mengukur akurasi. Seleksi *non-maximal* dilakukan untuk menghapus *bounding box* yang tumpang tindih dan mempertahankan yang paling relevan. Nilai *loss* dihitung untuk mengevaluasi performa deteksi, dan apabila nilai tidak *konvergen* (stabil), proses kembali ke pembagian citra menjadi *grid*. Deteksi objek kemudian dilakukan dengan memanfaatkan *bounding box* yang telah disesuaikan. Akhirnya, hasil deteksi dievaluasi untuk mengukur kinerja sistem deteksi sebelum proses diakhiri.

### 3.4 Perancangan Sistem

Perancangan sistem merupakan penentuan proses dan data yang diperlukan oleh sistem untuk implementasi akhir sistem. Proses ini melibatkan penetapan arsitektur sistem, pemilihan data yang sesuai, serta perencanaan rinci untuk semua komponen sistem. Tujuan utamanya adalah menciptakan pandangan terperinci tentang interaksi antarbagian sistem dan lingkungannya, sehingga memungkinkan pengembangan sistem yang efisien, handal, dan mudah dipelihara.

#### 3.4.1 Perancangan Model

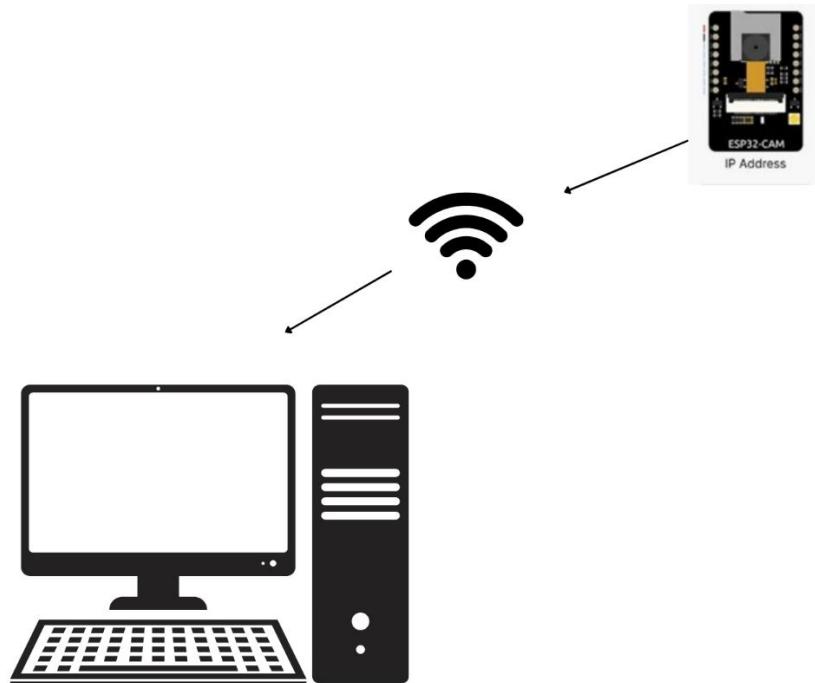


**Gambar 3. 6 Perancangan Model**

Perancangan Model dimulai dari pengumpulan data hingga sistem mendapatkan model dapat bekerja sesuai dengan kebutuhan. Pengambilan data ini dilakukan dengan 2 cara yaitu pengambilan data dari ATCS (*Area Traffic Control System*) Medan, yang merupakan *website* pemerintah yang menampilkan video CCTV pada jalan Medan secara siaran langsung yang dapat diakses oleh publik. Kemudian adapula pengambilan data yang diambil sendiri secara langsung menggunakan kamera gawai yang kemudian akan digunakan sebagai *dataset*. Setelah terkumpul, gambar tersebut akan dipilih terlebih dahulu gambar mana yang dapat digunakan dan mana gambar yang tidak dapat digunakan. Sebelum *dataset* dilatih dilakukan terlebih dahulu persiapan terhadap *dataset* tersebut, yakni dengan melakukan

pelabelan terhadap data set yang dilakukan pada platform Roboflow. Selain pelabelan manual *pre-processing* dan augmentasi juga dilakukan terhadap *dataset*. Setelah dilakukan persiapan dengan langkah-langkah tersebut, *dataset* kemudian dilatih menggunakan *Google Colab*. Setelah dilakukan *training* terhadap *dataset*, *dataset* akan dievaluasi terhadap hasil dari *training* tersebut. Setelah didapatkan hasil terbaik model melalui *training* dan pengujian *dataset*, model itulah yang akan digunakan untuk memproses pendekripsi kendaraan. Kemudian model sistem akan mengirimkan hasil keluaran sistem yang berupa durasi lampu lalu lintas yang ideal sesuai dengan jumlah kendaraan yang berada di jalan dan status dari lalu lintas tersebut.

### 3.4.2 Perancangan Alat



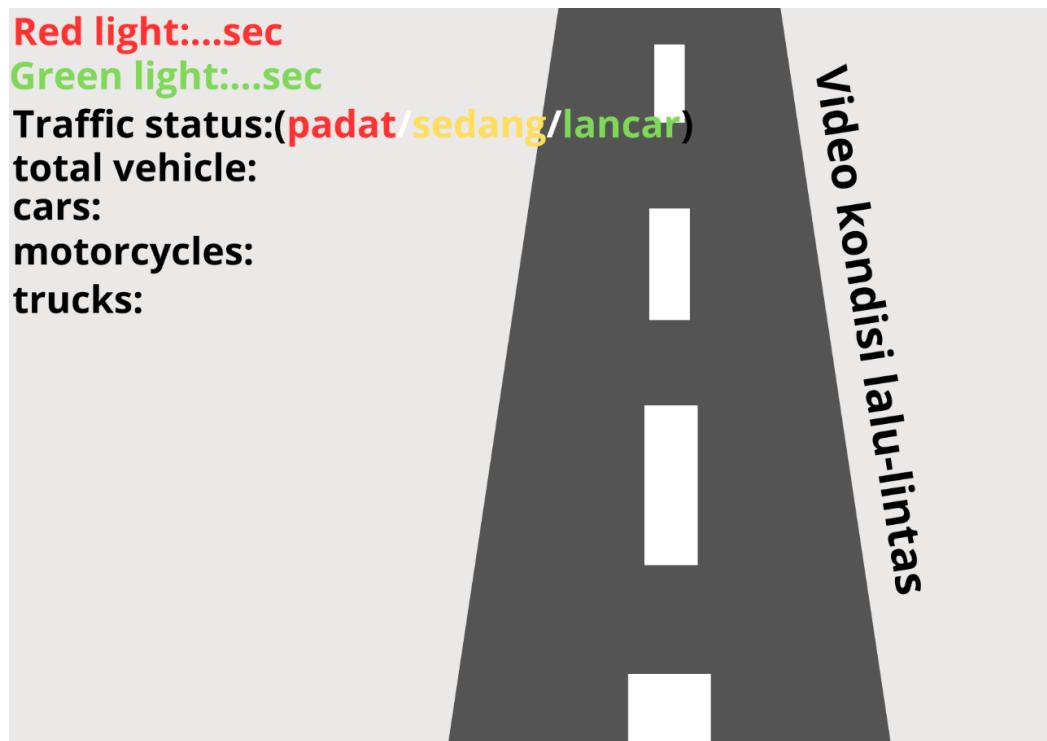
**Gambar 3. 7 Perancangan Alat**

Gambar diatas menunjukkan alat dan kerja sistem pengaturan lalu lintas berbasis IoT. Pertama, kamera ESP32-CAM digunakan untuk menangkap citra kendaraan di persimpangan jalan dan mengirimkan data tersebut ke dalam server lokal. Data kemudian diterima oleh model YOLO. Model YOLO menganalisis data citra untuk mendekripsi dan menghitung jumlah kendaraan. Berdasarkan analisis tersebut, model akan mengirimkan hasil analisa yang berupa jumlah kendaraan yang sedang

berhenti, status lalu lintas pada kondisi jalan dan durasi ideal lampu lalu lintas yang sesuai dengan kondisi lalu lintas tersebut.

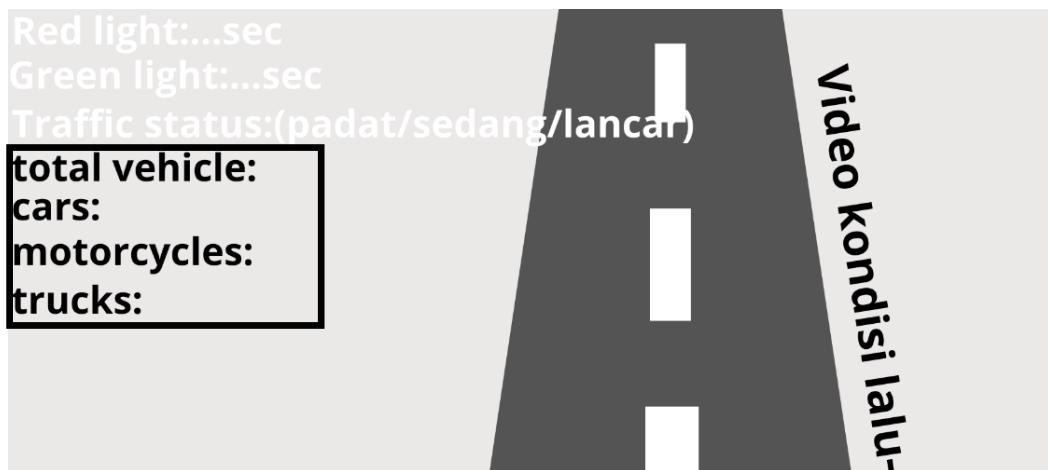
### 3.4.3 Perancangan Interface

Rancangan *interface* sistem memerlukan pendekatan yang menawarkan tampilan yang sederhana. Hal ini bertujuan untuk memberikan kenyamanan serta mempermudah pengguna dalam berinteraksi dengan sistem tersebut.



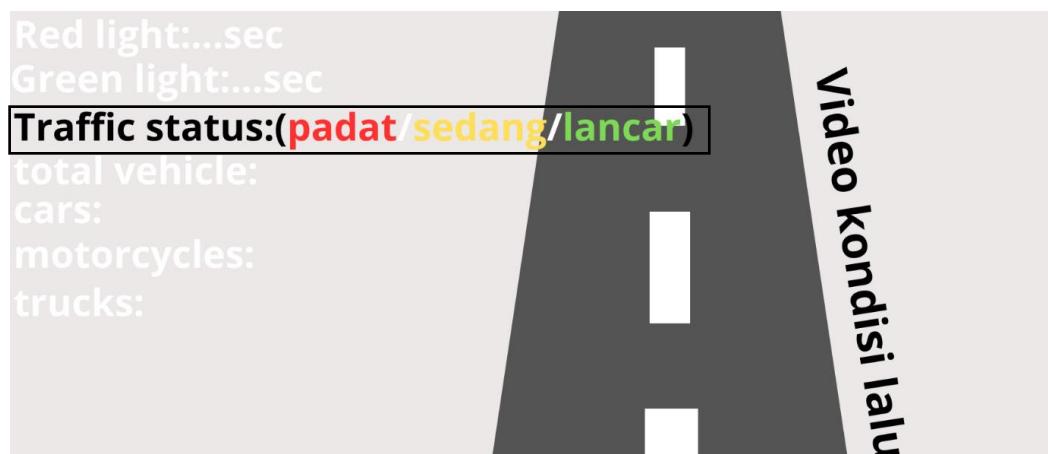
**Gambar 3. 8 Perancangan *Interface***

Gambar 3.8 di atas merupakan gambaran sederhana tampilan dari hasil deteksi sistem. *Interface* ini dirancang dengan tujuan agar pengguna dapat dengan mudah memahami hasil yang diberikan oleh sistem. Hasil yang ditunjukkan oleh sistem berupa video keadaan lalu lintas dengan tampilan berisi kendaraan pada lalu lintas yang sedang dideteksi dalam video. Kemudian adapula tampilan berupa durasi lampu lalu lintas sesuai dengan hasil deteksi, status dari lalu lintas, dan total kendaraan yang sedang terdeteksi beserta jumlahnya sesuai dengan kelasnya masing-masing.



**Gambar 3. 9** *Output Total Kendaraan*

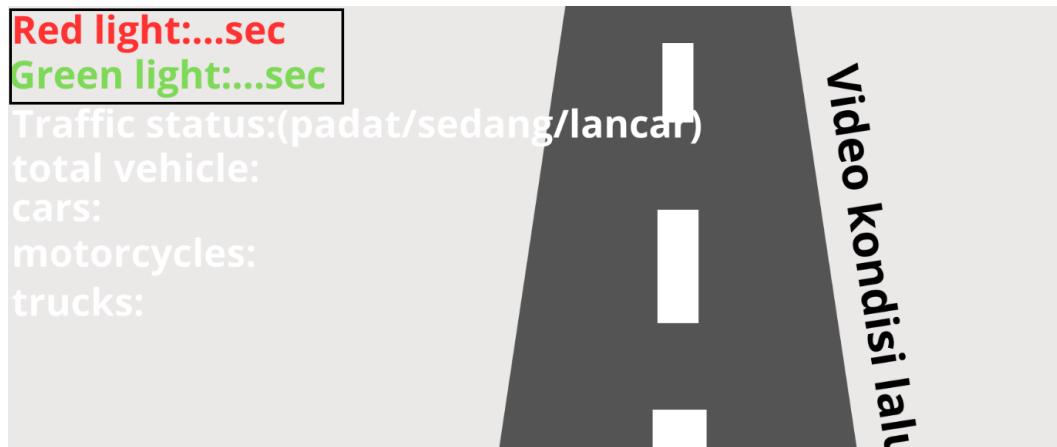
Pada gambar 3.9 di atas merupakan hasil *output* yang menunjukkan jumlah kendaraan yang telah dideteksi. Tiap kelas dari kendaraan yang dideteksi akan ditampilkan yang berupa mobil pada tampilan *cars*, sepeda motor pada tampilan *motorcycles* dan truk pada tampilan *trucks* sesuai jumlahnya yang terdeteksi. Kemudian total dari kendaraan berhenti yang dideteksi oleh sistem akan ditampilkan pada “*Total vehicle*”.



**Gambar 3. 10** *Output Status Lalu Lintas*

Pada gambar 3.10 di atas merupakan hasil dari pendeksi status lalu lintas. Yaitu antara status padat, status sedang, atau status lancar. Penentuan status dari lalu lintas yang digunakan pada penelitian ini, ditentukan dari kecepatan objek yang telah dideteksi. Apabila terdapat beberapa kendaraan yang bergerak cepat maka status lalu lintas yang tertera adalah “lancar”. Sebaliknya apabila terdapat kendaraan yang bergerak lambat maka status lalu lintas yang tertera adalah “sedang”. Status “padat”

akan ditampilkan apabila kendaraan yang terdeteksi bergerak lambat dan terdapat lebih dari 20 kendaraan yang terdeteksi. Status dari lalu lintas yang dideteksi dapat dilihat pada tampilan “*Traffic status*”.



Kemudian setelah didapati hasil dari deteksi, Maka sistem akan menampilkan *output* yaitu berupa durasi ideal lampu merah dan lampu hijau berdasarkan kondisi yang telah ditentukan.

## **BAB 4**

### **IMPLEMENTASI DAN PENGUJIAN SISTEM**

#### **4.1 Implementasi sistem**

##### **4.1.1 Pengujian Sistem Spesifikasi Perangkat Keras**

Untuk dapat menerapkan dan menguji implementasi YOLO dalam memprediksi kemacetan lalu lintas secara *realtime*, diperlukan perangkat keras dengan spesifikasi sebagai berikut.:

1. Processor AMD® Ryzen 7 series 5700U
2. Memory RAM 8 GB
3. SSD dengan kapasitas 500 GB
4. GPU AMD Radeon *Graphic*

Untuk mendukung persyaratan perangkat keras yang diperlukan, penulis juga memanfaatkan layanan *cloud* gratis dari Google yang disebut *Google Colab*. Dalam *Google Colab*.

##### **4.1.2 Spesifikasi Perangkat Lunak**

Spesifikasi perangkat lunak yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Sistem Operasi Windows 11 Home 64 bit *operating system*.
2. *Google Colab*
3. *Python3*
4. *Library*: warnings, os, shutil, numpy (np), pandas (pd), matplotlib.pyplot as plt, seaborn (sns), cv2, yaml, PIL.Image, ultralytics.YOLO, IPython.display.Video.
5. Visual Studio Code

#### **4.2 Pengumpulan Dataset**

Pengumpulan dataset merupakan langkah krusial dalam penelitian ini untuk memastikan keberhasilan implementasi YOLO (You Only Look Once) dalam mendeteksi kemacetan lalu lintas secara *real-time*. Sumber data yang digunakan

mencakup rekaman dari kamera CCTV lalu lintas yang diambil dari platform situs web pemerintah dan rekaman video langsung oleh peneliti. Pengambilan video dilakukan di berbagai lokasi dengan kepadatan lalu lintas yang bertujuan untuk memperkaya variasi dataset dan memastikan model dapat mendeteksi dari sudut pandang simulasi yang sudah ditentukan. Semua sumber data ini memberikan keragaman yang diperlukan untuk menciptakan model yang dapat bekerja maksimal.



**Gambar 4. 1** Contoh *dataset* dari ATCS Medan

Pada gambar 4.1 di atas merupakan contoh *dataset* yang diambil dari *website* ATCS kota Medan. ATCS (*Area Traffic Control System*) merupakan website dinas perhubungan kota medan yang menampilkan tangkapan CCTV kota Medan yang terletak pada jalan secara langsung. *Website* ini dapat diakses oleh Masyarakat secara publik. Akan tetapi *website* ini hanya memberikan tangkapan dari kamera CCTV pada jalan secara langsung saja. Pada penelitian ini, data yang diambil dari ATCS Medan merupakan rekaman layar oleh peneliti pada *website* ATCS di beberapa lokasi. Rekaman diambil pada Jalan Zainul Ariffin, Medan pada siang hari dengan cuaca cerah dengan durasi 18 menit, pada Jalan Diponegoro, Medan pada siang hari dengan cuaca hujan dengan durasi 10 menit. Pada Jalan Pengadilan, Medan pada siang hari dengan cuaca cerah dengan durasi 21 menit dan pada Jalan Jendral Ahmad Yani pada siang hari dengan cuaca mendung dengan durasi 15 menit. Pengambilan data pada ATCS Medan memiliki keterbatasan seperti hasil

tangkapan dari kamera CCTV menjadi buram ketika hujan dan objek tidak terlihat jelas pada malam hari. Kemudian agar data yang diambil bervariasi peneliti juga mengambil data secara manual menggunakan kamera pada gawai.



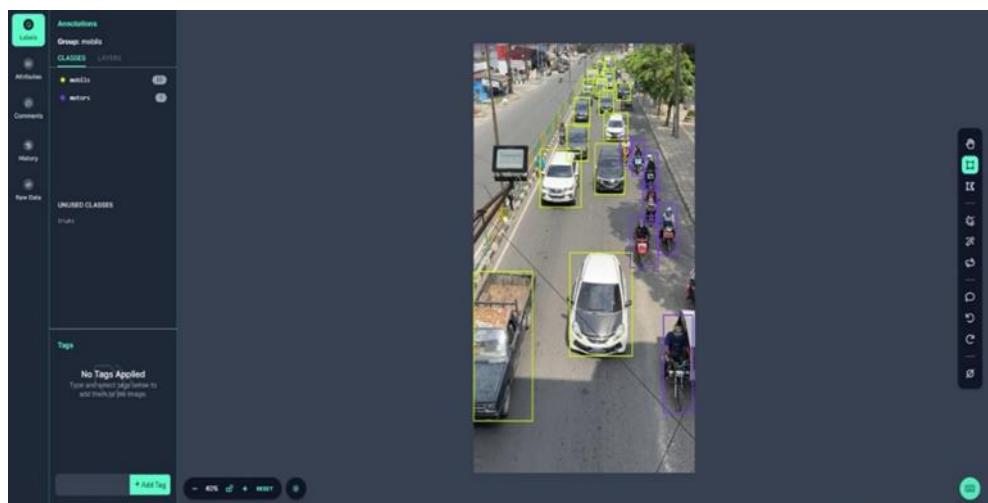
**Gambar 4. 2** Contoh *dataset* dari gawai peneliti

Pada gambar 4.2 di atas merupakan contoh *dataset* yang diambil langsung oleh peneliti. *Dataset* yang diambil berupa video dengan durasi 20 - 60 menit pada beberapa lokasi yang berbeda. Pengambilan *dataset* ini diambil pada Jalan Jamin Ginting, Medan di siang dengan cuaca cerah dan malam hari dengan cuaca hujan, Jalan Gatot Subroto, Medan di pagi hari dengan cuaca cerah. Jalan Jendral Katamso, Medan di siang hari dengan cuaca hujan, dan Jalan Sisingamangaraja, Medan di siang hari dengan cuaca cerah dan malam hari dengan cuaca cerah. Total video yang diambil sebanyak 14 potongan video dengan durasi rata-rata sekitar 40 menit.

Kriteria pemilihan data mencakup variasi waktu pengambilan (pagi, siang, malam) untuk mengakomodasi perbedaan pencahayaan, serta kondisi cuaca yang beragam (cerah dan hujan) untuk memastikan model mampu bekerja dalam berbagai situasi. Kemudian untuk menyesuaikan format data dengan input YOLO semua data yang didapat diubah menjadi format mp4. Dengan dataset yang komprehensif dan representatif ini, diharapkan model YOLO yang dikembangkan mampu mendeteksi kemacetan lalu lintas secara real-time dengan akurasi tinggi, menjadikannya alat yang efektif dalam manajemen lalu lintas.

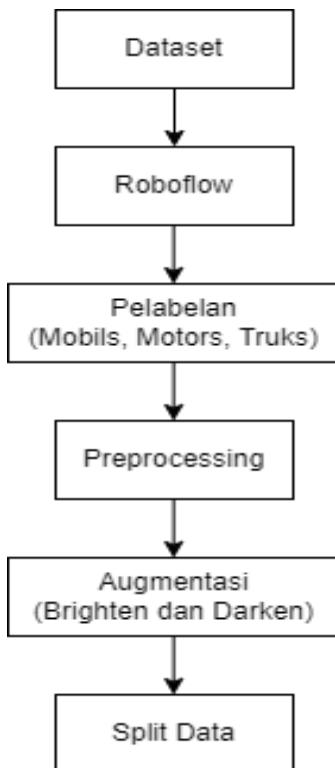
### 4.3 Pelabelan Dataset

Pelabelan dataset merupakan langkah penting dalam penelitian ini untuk memastikan model YOLO (You Only Look Once) dapat mendeteksi kemacetan lalu lintas secara real-time dengan akurasi tinggi. Proses pelabelan melibatkan penandaan setiap objek yang relevan dalam dataset dengan informasi yang sesuai, sehingga model dapat belajar mengenali pola-pola yang berbeda. Kriteria pelabelan yang digunakan meliputi jenis kendaraan seperti mobil, motor, dan truk.



**Gambar 4. 3** Pelabelan Kelas mobil, motors, truk

Platform yang digunakan pada proses pelabelan adalah Roboflow. Dalam proses pelabelan, Roboflow telah menyediakan fitur semi-automatis yang mempermudah dan mempercepat pelabelan dataset. Roboflow menawarkan antarmuka pengguna yang intuitif, memungkinkan peneliti untuk menggambar bounding box di sekitar objek dengan cepat dan akurat.

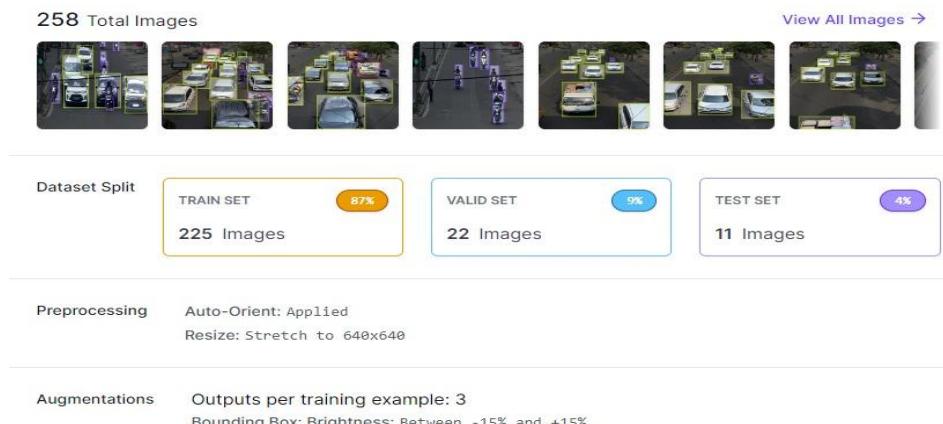


**Gambar 4. 4** Proses persiapan *dataset*

Proses pelabelan *dataset* dengan Roboflow dimulai dengan pengunggahan data ke platform. Video yang telah dipecah menjadi *frame* individu diunggah ke Roboflow, sehingga setiap *frame* siap untuk diberi label. Peneliti kemudian menggunakan alat gambar dalam Roboflow untuk menggambar bounding box di sekitar setiap kendaraan dan memberi label yang menunjukkan jenis kendaraan. Pelabelan dilakukan dengan menggunakan bounding box di sekitar kendaraan yang akan diidentifikasi oleh sistem. Setiap kendaraan diberi kelas sesuai dengan jenis kendaraan tersebut,yaitu berupa mobils untuk mobil, motors untuk sepeda motor, dan truks untuk truk.

Setelah pelabelan awal selesai, dilakukan pemeriksaan dan validasi untuk memastikan konsistensi dan akurasi. Frame yang telah diberi label diperiksa ulang untuk mengurangi kesalahan dan memastikan semua objek telah dilabeli dengan benar. Hal ini dilakukan dengan tujuan objek dapat ditandai dengan tepat, dan Memberikan label yang sesuai dan dapat diterapkan untuk memberikan informasi yang jelas tentang jenis objek yang terdeteksi.

#### 4.4 Preprocessing, Augmentasi dan split dataset



**Gambar 4. 5** Proses *Preprocessing, Augmentasi* dan *Split Dataset*

Pada dataset yang sudah diberi label dilakukan proses preprocessing yang bertujuan untuk menyiapkan data agar sesuai dengan format atau persyaratan yang diperlukan oleh model. Pada penelitian ini preprocessing yang dilakukan adalah melakukan *resize* gambar menjadi 640x640. Augmentasi dilakukan dengan tujuan untuk meningkatkan keragaman-keragaman data latihan, sehingga model dapat belajar dari beragam situasi atau kondisi. Augmentasi yang dilakukan pada dataset penelitian ini meliputi *brightness*, dan *darken*. Setelah proses diatas selesai didapatkan 258 total gambar. kemudian gambar tersebut dibagi menjadi tiga bagian set yaitu Train set 225 gambar, Valid Set 22 gambar dan Test set sebanyak 11 gambar, pembagian ini dipilih dengan mempertimbangkan jumlah dataset. Dataset yang telah divalidasi kemudian diekspor dari Roboflow dalam format yang kompatibel dengan YOLO, seperti format YOLOv8, untuk digunakan dalam pelatihan model. Dengan menggunakan Roboflow, proses pelabelan dataset menjadi lebih efisien dan akurat, yang pada akhirnya akan menghasilkan model YOLO yang mampu mendeteksi kemacetan lalu lintas secara *real-time* dengan tingkat akurasi yang tinggi.

## 4.5 Training Dataset

Setelah dilakukan proses persiapan *dataset*, kemudian dilakukan *training dataset* pada *Google Colab*.

### 4.5.1 Mengunduh Library Ultralytics

```
!pip install ultralytics -q
```

**Gambar 4. 6** Kode untuk Mengunduh *Library Ultralytics*

Gambar 4.6 di atas merupakan proses mengunduh *Library Ultralytics* sehingga dapat mengakses fungsionalitasnya dan menggunakan alat dan model yang disediakan.

### 4.5.2 Import YOLO

```
from PIL import Image
from ultralytics import YOLO
from IPython.display import Video
```

**Gambar 4. 7** Kode untuk Import YOLO

Gambar 4.7 di atas merupakan proses mengimpor YOLO dari *Library Ultralytics* yang sebelumnya telah dideklarasikan. Pada baris ini juga terdapat pemanggilan pyplot. Pyplot adalah submodul dari Matplotlib yang memungkinkan pembuatan visualisasi data dengan Python secara sederhana. Ini menyediakan fungsi untuk membuat plot, mengatur penampilan plot, mengelola sumbu dan label, menambahkan anotasi, serta menyimpan plot. Pyplot membantu membuat grafik seperti *line plot*, *scatter plot*, *bar plot*, *histogram*, dan lainnya dengan mudah dan cepat.

### 4.5.3 Inisialisasi Model YOLO

```
# Load a pretrained YOLOv8n model from Ultralytics
model = YOLO('yolov8n.pt')
```

**Gambar 4. 8** Kode untuk menginisialisasi Model YOLO

Gambar 4.5 di atas mengacu pada proses memuat model YOLOv8n yang telah di-*training* sebelumnya dari pustaka Ultralytics. Dengan menggunakan `from ultralytics import YOLO`, kita mengimpor kelas YOLO dari pustaka tersebut.

Kemudian, dengan baris `model = YOLO('yolov8n.pt')`, model YOLOv8n yang telah di-*training* sebelumnya dimuat dari file 'yolov8n.pt'. Ini memungkinkan penggunaan model tersebut untuk deteksi objek dalam gambar atau video dengan menggunakan arsitektur dan bobot yang telah ditentukan sebelumnya.

#### 4.5.4 Proses Mengakses *Dataset*

```
# Define the dataset_path
dataset_path = '/content/drive/My Drive/dataset'
```

**Gambar 4. 9** Kode untuk Mengakses Dataset

Gambar 4.9 di atas merupakan proses pengaksesan *dataset* yang sebelumnya sudah di-*upload* pada Google Drive yang kemudian dapat digunakan untuk melatih model atau melakukan berbagai tugas pengolahan data lainnya. Melalui *path* ini akan diarahkan ke folder tempat penyimpanan *dataset*.

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

nc: 3
names: ['mobil', 'motor', 'truk']
```

**Gambar 4. 10** File *data.yaml*

#### 4.5.5 Proses Mendefinisikan Jumlah *Class* pada *Dataset*

```
# Define the dataset_path
dataset_path = '/content/drive/My Drive/dataset'

# Set the path to the YAML file
yaml_file_path = os.path.join(dataset_path, 'data.yaml')

# Load and print the contents of the YAML file
with open(yaml_file_path, 'r') as file:
    yaml_content = yaml.load(file, Loader=yaml.FullLoader)
    print(yaml.dump(yaml_content, default_flow_style=False))
```

**Gambar 4. 11** Kode untuk Mendefinisikan Jumlah *Class* pada *Dataset*

Gambar 4.11 di atas bertujuan untuk membaca file YAML yang berisi informasi tentang jumlah kelas (*num\_classes*) yang ada dalam *dataset*. Yang pertama

mengimpor modul ‘yaml’, yang memungkinkan untuk membaca dan menulis data YAML dalam Python. Kemudian *file* YAML yang ada pada Google Drive dibuka dan mengonversinya menjadi struktur data Python. Kemudian mengakses nilai yang terkandung dalam *file* YAML setelah dibaca dan mengubahnya menjadi *string* dan disimpan dalam variabel ‘*num\_classes*’.

#### **4.5.6 Proses Training Dataset**

```
results = model.train(
    data=yaml_file_path,
    epochs=100,
    imgsz=640,
    device=0,
    patience=50,
    batch=32,
    optimizer='auto',
    lr0=0.0001,
    lrf=0.1,
```

**Gambar 4. 12** Kode untuk *Training Dataset*

Gambar 4.12 di atas merupakan proses pelatihan model deteksi objek menggunakan arsitektur YOLOv8n dengan menggunakan *library Ultralytics*. Dalam proses ini, sebuah objek model YOLOv8n dibuat dengan menyediakan *path* menuju *file* model yang telah dilatih sebelumnya. Selanjutnya, metode `train()` dipanggil pada objek model tersebut dengan menyediakan *path* menuju *file* YAML yang berisi konfigurasi *dataset*, parameter-parameter pelatihan seperti jumlah *epochs*, *patience*, ukuran *batch*, dan ukuran gambar. Proses pelatihan dilakukan menggunakan *dataset* yang telah ditentukan dan hasilnya disimpan dalam variabel `results`. Kode ini memastikan bahwa model dilatih sesuai dengan konfigurasi yang ditentukan, dan hasil pelatihan dapat digunakan untuk evaluasi lebih lanjut atau untuk penerapan praktis dalam tugas deteksi objek.

#### **4.5.7 Tuning Hyperparameter**

Pada penelitian ini, penyetelan *hyperparameter* dilakukan dengan kombinasi berbagai nilai jumlah *epochs*, yakni 100, 150, dan 200 *epochs* dengan *learning rate* 0.01. Kombinasi-kombinasi ini dieksplorasi secara cermat untuk menemukan model terbaik dan optimal. Hasil *tuning hyperparameter* dapat diperhatikan dalam tabel di bawah ini:

**Tabel 4. 1** Training Epoch 100, Learning Rate 0.01

<b>Epoch</b>	<b>box_loss</b>	<b>cls_loss</b>	<b>dfl_loss</b>	<b>Box(P)</b>	<b>Box(R)</b>	<b>mAP</b>
1/100	1.285	0.7471	1.155	0.731	0.881	0.829
2/100	1.233	0.7317	1.12	0.793	0.849	0.779
3/100	1.271	0.7461	1.139	0.701	0.884	0.762
4/100	1.261	0.7472	1.138	0.768	0.841	0.825
5/100	1.246	0.726	1.135	0.702	0.892	0.745
6/100	1.236	0.7483	1.125	0.675	0.876	0.76
7/100	1.309	0.7423	1.147	0.74	0.854	0.775
8/100	1.27	0.7453	1.145	0.727	0.879	0.746
9/100	1.28	0.7302	1.133	0.675	0.907	0.733
10/100	1.262	0.7328	1.137	0.69	0.88	0.713
...	...	...	...	...	...	...
91/100	0.992	0.541	1.023	0.769	0.839	0.787
92/100	0.9866	0.5332	1.013	0.732	0.899	0.792
93/100	0.9864	0.5335	1.019	0.723	0.899	0.789
94/100	0.9504	0.5316	1.007	0.729	0.889	0.772
95/100	0.9709	0.5238	1.017	0.739	0.884	0.767
96/100	0.9504	0.5184	1.016	0.736	0.88	0.766
97/100	0.9705	0.5204	1.009	0.729	0.887	0.766
98/100	0.951	0.5176	1.014	0.73	0.886	0.763
99/100	0.9145	0.4961	0.9955	0.735	0.887	0.767
100/100	0.9458	0.5049	1.004	0.735	0.884	0.77

**Tabel 4. 2** Summary Training Epoch 100, Learning Rate 0.01

<b>Class</b>	<b>Box(P)</b>	<b>Box(R)</b>	<b>mAP</b>
All	0.726	0.848	<b>0.849</b>
Mobils	0.872	0.825	0.900
Motors	0.873	0.718	0.817
Trucks	0.433	1.000	0.830

**Tabel 4. 3** Training Epoch 150, Learning Rate 0.01

<b>Epoch</b>	<b>Box_Loss</b>	<b>Class_Loss</b>	<b>DFL_Loss</b>	<b>Box(P)</b>	<b>Box(R)</b>	<b>mAP</b>
1/150	3.009	3.42	2.387	0.00419	0.0388	0.012
2/150	2.362	2.182	1.885	0.592	0.178	0.190
3/150	2.219	1.824	1.732	0.739	0.368	0.403
4/150	2.055	1.613	1.627	0.446	0.526	0.557
5/150	1.963	1.511	1.574	0.558	0.551	0.579
6/150	1.906	1.462	1.524	0.630	0.602	0.614
7/150	1.901	1.416	1.510	0.797	0.653	0.718
8/150	1.844	1.352	1.485	0.594	0.745	0.701
9/150	1.813	1.289	1.454	0.930	0.605	0.743
10/150	1.749	1.243	1.423	0.728	0.543	0.693
...	...	...	...	...	...	...
141/150	0.8694	0.4778	0.9769	0.735	0.876	0.753
142/150	0.8647	0.4717	0.9775	0.730	0.875	0.750
143/150	0.8589	0.4708	0.9783	0.737	0.868	0.743
144/150	0.8399	0.4577	0.9636	0.736	0.877	0.745
145/150	0.8480	0.4630	0.9695	0.744	0.880	0.749
146/150	0.8483	0.4662	0.9739	0.730	0.867	0.738
147/150	0.8305	0.4465	0.9637	0.740	0.867	0.750
148/150	0.8259	0.4466	0.9578	0.730	0.859	0.740
149/150	0.8881	0.4974	0.9725	0.736	0.864	0.750
150/150	0.8397	0.4521	0.9593	0.725	0.886	0.759

**Tabel 4. 4** Summary Training Epoch 150, Learning Rate 0.01

<b>Class</b>	<b>Box(P)</b>	<b>Box(R)</b>	<b>mAP50</b>
all	0.749	0.816	0.835
mobils	0.893	0.761	0.904
motors	0.858	0.688	0.847
truks	0.498	1.000	0.753

**Tabel 4. 5 Training Epoch 200, Learning Rate 0.01**

<b>Epoch</b>	<b>Box_Loss</b>	<b>Class_Loss</b>	<b>DFL_Loss</b>	<b>Box(P)</b>	<b>Box(R)</b>	<b>mAP</b>
1/200	1.091	0.6227	1.062	0.661	0.888	0.725
2/200	1.014	0.5957	1.027	0.697	0.762	0.732
3/200	1.064	0.6112	1.048	0.629	0.878	0.774
4/200	1.068	0.6230	1.050	0.648	0.881	0.770
5/200	1.095	0.6334	1.066	0.598	0.856	0.759
6/200	1.092	0.6474	1.060	0.640	0.767	0.711
7/200	1.157	0.6506	1.076	0.665	0.834	0.697
8/200	1.153	0.6641	1.097	0.655	0.882	0.706
9/200	1.177	0.6619	1.089	0.683	0.875	0.709
10/200	1.176	0.6772	1.096	0.695	0.886	0.727
...	...	...	...	...	...	...
191/200	0.8072	0.4408	0.9464	0.763	0.843	0.753
192/200	0.8139	0.4432	0.9441	0.748	0.856	0.748
193/200	0.7944	0.4359	0.9363	0.660	0.897	0.709
194/200	0.8580	0.4833	0.9609	0.653	0.890	0.711
195/200	0.7877	0.4326	0.9416	0.702	0.877	0.744
196/200	0.7802	0.4324	0.9429	0.733	0.880	0.752
197/200	0.7851	0.4241	0.9431	0.747	0.869	0.754
198/200	0.7995	0.4327	0.9367	0.743	0.849	0.752
199/200	0.7689	0.4211	0.9320	0.747	0.850	0.754
200/200	0.8295	0.4904	0.9642	0.752	0.850	0.753

**Tabel 4. 6 Summary Training Epoch 200, Learning Rate 0.01**

<b>Class</b>	<b>Box(P)</b>	<b>Box(R)</b>	<b>mAP50</b>
all	0.678	0.878	0.798
mobils	0.824	0.872	0.887
motors	0.795	0.815	0.809
truks	0.414	0.949	0.696

Kombinasi *epoch* dan *learning rate* dalam pelatihan model bertujuan untuk mengoptimalkan kinerja model melalui penyesuaian parameter pelatihan. *Epoch* menentukan jumlah siklus lengkap melalui *dataset* dan *learning rate* mengatur seberapa besar bobot model diperbarui. Kombinasi yang tepat antara keduanya penting untuk memastikan model belajar secara efektif dan efisien. Pada penelitian ini teknik yang digunakan berupa *random search* atau secara acak. Setelah melakukan *tuning hyperparameter*, ditemukan kombinasi optimal pada *epoch* 100 dengan *learning rate* 0.01. Kombinasi ini menghasilkan nilai mAP tertinggi di antara kombinasi lainnya, yaitu sebesar 0.849 untuk semua kelas seperti yang terlihat pada tabel 4.2. Tabel tersebut menampilkan hasil evaluasi model deteksi objek berdasarkan tiga metrik yaitu *Box Precision* (Box(P)), *Box Recall* (Box(R)), dan *mean Average Precision* (mAP). Kolom "Class" menunjukkan kategori objek yang dievaluasi, yaitu "All" (semua kelas), "Mobils" (mobil), "Motors" (sepeda motor), dan "Truks" (truk). *Box Precision* mengukur ketepatan prediksi model, dengan nilai tertinggi pada "Motors" (0.873) dan terendah pada "Truks" (0.433). *Box Recall* mengukur sensitivitas model dalam mendekripsi objek, dengan nilai tertinggi pada "Truks" (1.000) dan terendah pada "Motors" (0.718). *Mean Average Precision* (mAP) memberikan gambaran kinerja keseluruhan model, dengan nilai tertinggi pada keseluruhan kelas 0.849. Kinerja terbaik model terlihat pada kombinasi *epoch* 100 dengan *learning rate* 0.01. Oleh karena itu, model yang dihasilkan dari pelatihan tersebut dipilih untuk menjadi model sistem.

#### 4.6 Evaluasi Model

Setelah melakukan *training* dengan *epoch* 100 dan *learning rate* 0.01. Kombinasi ini menghasilkan nilai mAP yaitu sebesar 0.849 untuk semua kelas seperti yang terlihat pada gambar 4.13. Oleh karena itu, model yang dihasilkan dari pelatihan tersebut dipilih untuk menjadi model sistem.

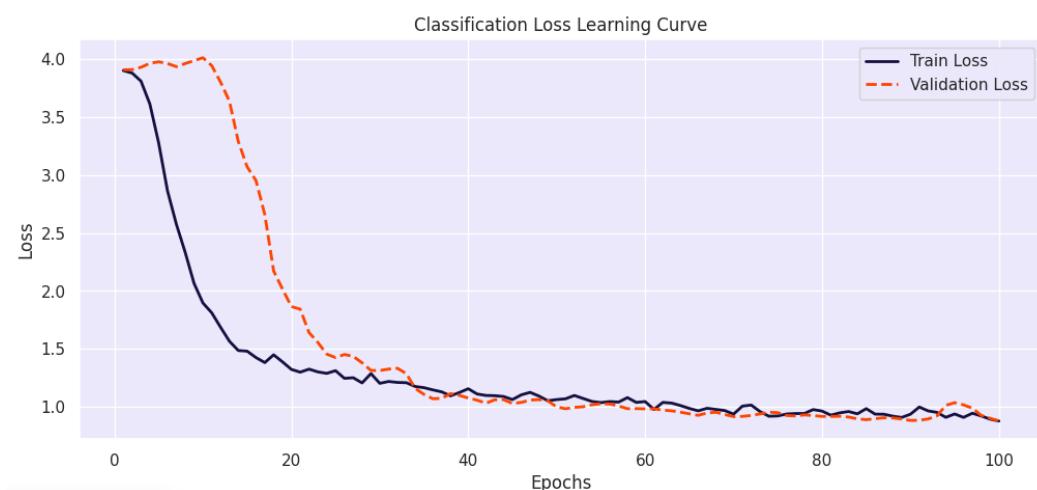
Metric	Value
metrics/precision(B)	0.690
metrics/recall(B)	0.835
metrics/mAP50(B)	0.820

Gambar 4. 13 Kode untuk *Training Dataset*

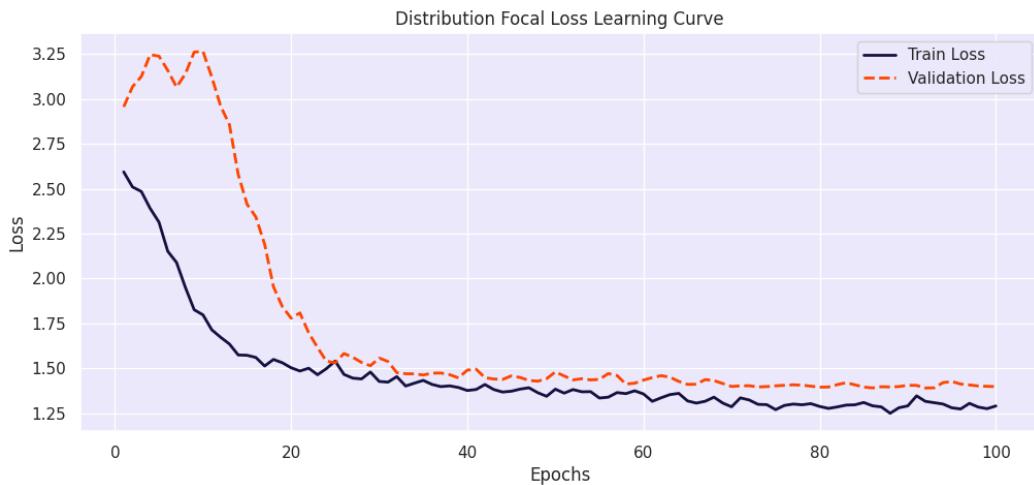
Berdasarkan proses pelatihan model pada *epoch* 100 dan *learning rate* 0.01 disajikan hasil evaluasi dari model yang berhasil dibangun. Grafik yang dihasilkan memberikan gambaran visual tentang performa model secara menyeluruh. Grafik tersebut mencakup beberapa metrik penting seperti *box loss*, *class loss*, dan *dfl loss*, yang ditunjukkan pada gambar 4.14, 4.15 dan 4.16.



**Gambar 4. 14** Grafik *Box Loss*

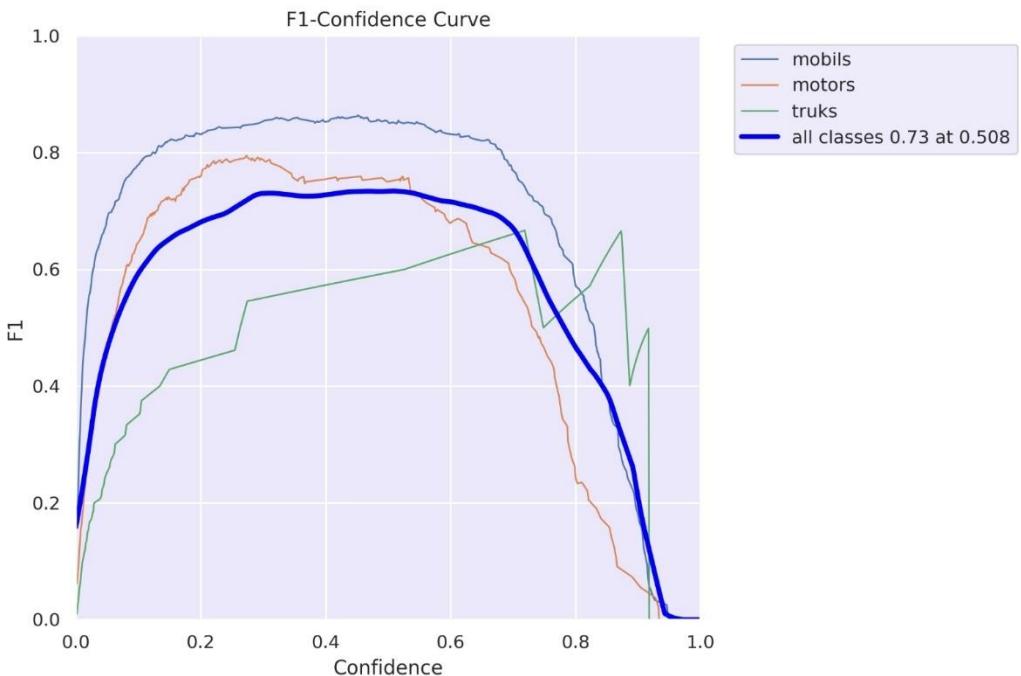


**Gambar 4. 15** Grafik *Classification Loss*



**Gambar 4. 16** Grafik *Distribution Focal Loss*

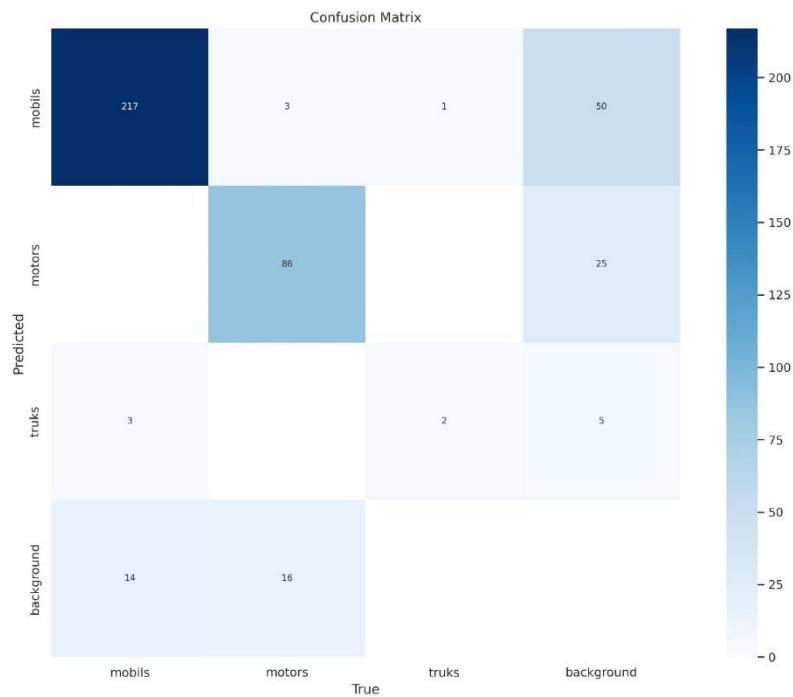
Secara umum, grafik yang disajikan menunjukkan hasil dari *training* data. Grafik untuk *train* mencakup data *box\_loss*, *cls\_loss*, dan *dfl\_loss*. Setiap grafik menunjukkan penurunan di setiap *epoch*-nya. Penurunan grafik ini mengindikasikan seberapa efektif grafik dalam memproses video yang diinput. Misalnya, *box\_loss* mengukur seberapa baik data dicocokkan dengan *bounding box*, *cls\_loss* menunjukkan efektivitas dalam mendeteksi kelas objek, dan *dfl\_loss* menilai kualitas data dari klasifikasi yang terdeteksi. Ini menandakan bahwa sistem mampu mempelajari tugas yang diberikan dengan baik. Pada grafik diatas menunjukkan bahwa model memiliki performa yang memuaskan dengan metrik seperti *box\_loss*, *class\_loss* dan *dfl\_loss* menunjukkan kinerja yang baik. Ketika kita melakukan evaluasi hasil untuk setiap kelas secara spesifik, terlihat adanya variasi yang tidak jauh berbeda dengan kelas lainnya. Selain menggunakan *box\_loss*, *class\_loss* dan *dfl\_loss* penelitian ini juga menggunakan *F1-Confidence Score* sebagai evaluasi yang terlihat pada Gambar 4.17 dibawah.



**Gambar 4. 17** *F1-Confidence score* model

Grafik yang ditampilkan adalah kurva *F1-Confidence score* yang menunjukkan hubungan antara kepercayaan (*confidence*) dan skor F1. Skor F1 dihitung sebagai rata-rata harmonis dari presisi dan recall, yang memberikan bobot yang sama untuk kedua metrik ini. Grafik menampilkan untuk tiga kelas berbeda yaitu "mobil," "motors," dan "truks," serta gabungan dari semua kelas. Pada sumbu horizontal (x) adalah tingkat kepercayaan, sedangkan sumbu vertikal (y) adalah skor F1, yang merupakan rata-rata harmonik dari presisi dan *recall*, digunakan untuk mengukur kinerja model klasifikasi. Kurva-kurva tersebut menggambarkan bagaimana performa model berubah seiring dengan peningkatan tingkat kepercayaan. Garis biru yang lebih tebal merepresentasikan kinerja gabungan dari semua kelas, dengan skor F1 tertinggi sebesar 0.73 pada tingkat kepercayaan 0.508. Hal ini menunjukkan bahwa pada tingkat kepercayaan tersebut, model mencapai keseimbangan terbaik antara presisi dan *recall* secara keseluruhan. Fungsi dari grafik ini untuk mengilustrasikan kinerja model klasifikasi yang dikembangkan, serta untuk menentukan tingkat kepercayaan optimal yang memaksimalkan skor F1. Fungsi skor F1 adalah untuk mengukur kinerja model klasifikasi dengan mempertimbangkan keseimbangan antara presisi dan *recall*. Presisi adalah proporsi prediksi positif yang benar-benar positif (*True Positives* / (*True Positives* + *False Positives*)).

*Positives)), sedangkan recall adalah proporsi kasus positif yang benar-benar terdeteksi sebagai positif ( $True\ Positives / (True\ Positives + False\ Negatives)$ ). Sehingga dapat digunakan untuk pengambilan keputusan dalam implementasi model tersebut. Setelah melakukan pelatihan model YOLOv8 dengan menggunakan data *training*, langkah selanjutnya adalah melakukan evaluasi menggunakan testing set yang terdiri dari 11 gambar. Dalam tahap evaluasi ini, *confusion matrix* digunakan untuk membandingkan hasil prediksi model dengan hasil sebenarnya. Evaluasi juga melibatkan pengukuran *accuracy* (akurasi), *precision* (presisi), dan *recall* (keberhasilan) untuk mengevaluasi kinerja model. Hasil evaluasi, yang direpresentasikan dalam bentuk *confusion matrix*, dari pemodelan menggunakan algoritma YOLOv8 dapat dilihat pada Gambar 4.18.*



**Gambar 4. 18 Confusion matrix**

Evaluasi kinerja menggunakan *confusion matrix* memanfaatkan empat konsep untuk menggambarkan hasil dari proses klasifikasi, yaitu:

*True Positive* (TP) Merupakan jumlah kasus positif yang benar-benar diklasifikasikan sebagai positif oleh model.

*True Negative* (TN) Merupakan jumlah kasus negatif yang benar-benar diklasifikasikan sebagai negatif oleh model.

*False Positive* (FP) Merupakan jumlah kasus negatif yang salah diklasifikasikan sebagai positif oleh model.

*False Negative* (FN) Merupakan jumlah kasus positif yang salah diklasifikasikan sebagai negatif oleh model.

### ***Accuracy***

*Accuracy* (akurasi) adalah ukuran umum untuk mengevaluasi kinerja dari sebuah model klasifikasi. Ini mengukur seberapa sering model tersebut membuat prediksi yang benar secara keseluruhan. akurasi dihitung dengan rumus:

$$\text{Accuracy} = \frac{\text{Total TP}}{\text{Total data}}$$

$$\text{Accuracy} = \frac{217 + 86 + 2 + 0}{271 + 111 + 10 + 30} = \frac{305}{422} = 0.722$$

### ***Precision***

*Precision* (presisi) adalah ukuran yang menggambarkan seberapa akurat model dalam mengklasifikasikan kasus positif. Precision merupakan proporsi dari data yang secara tepat diklasifikasikan sebagai positif dari keseluruhan data yang diprediksi sebagai positif. Presisi dihitung dengan rumus:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

*Precision* untuk kelas “mobil”

$$\text{TP} = 217$$

$$\text{FP} = 0 + 3 + 1 + 50 = 54$$

*Precision* untuk kelas “sepeda motor”

$$\text{TP} = 86$$

$$\text{FP} = 0 + 0 + 0 + 25 = 25$$

*Precision* untuk kelas “ truk”

$$\text{TP} = 2$$

$$\text{FP} = 3 + 0 + 0 + 5 = 8$$

*Precision* untuk kelas “ background”

$$\text{TP} = 0$$

$$\text{FP} = 14 + 16 + 0 + 0 = 30$$

$$\text{Precision} = \frac{217 + 86 + 2 + 0}{271 + 111 + 10 + 30} = \frac{305}{422} = 0.722$$

### ***Recall***

*Recall* adalah ukuran keberhasilan sistem dalam mendapatkan prediksi yang relevan. Berikut rumus untuk menghitung *recall*.

$$\text{Recall} = \frac{TP}{TP+FN}$$

*Recall* untuk kelas “mobil”

$$TP = 217$$

$$FP = 0 + 0 + 3 + 50 = 53$$

*Recall* untuk kelas “sepeda motor”

$$TP = 86$$

$$FP = 3+0+0+16= 19$$

*Recall* untuk kelas “ truk”

$$TP = 2$$

$$FP = 3+0+0+5 =8$$

*Recall* untuk kelas “ background”

$$TP = 0$$

$$FP = 0+25+5+0 =30$$

$$\text{Recall} = \frac{217+86+2+0}{270+115+10+30} = \frac{305}{425} = 0.718$$

### ***F1-score***

*F1-score* adalah ukuran yang membandingkan dari precision dan recall yang berguna untuk mengevaluasi kinerja model klasifikasi, terutama ketika kelas-kelas tidak seimbang.

$$F1\text{-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$F1\text{score} = 2 \times \frac{0.722 \times 0.717}{0.722 + 0.717} = \frac{0.517}{1.439} = 0.713$$

## **4.7 Proses Deteksi Model YOLO**

Berikut ini proses deteksi Model YOLO yang melalui beberapa tahapan sehingga didapatkan hasil akhir sebuah objek yang terdeteksi.

### **4.7.1 Resize citra**

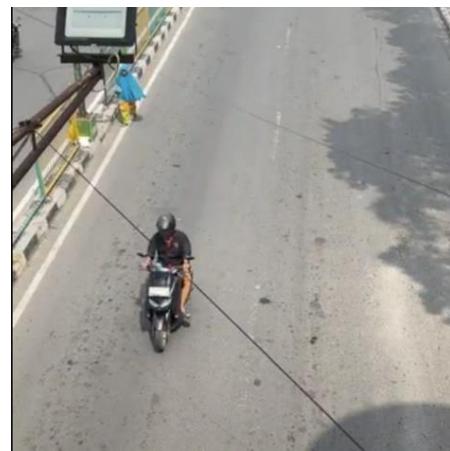
Sebelum mengolah data citra dengan arsitektur YOLO, ukuran citra yang di-input disesuaikan terlebih dahulu dengan *resize*. Melakukan *resize* data berguna untuk menyamakan ukuran citra yang beragam pada masukan. Kemudian video yang telah dimasukan akan dipotong tiap *frame*. Pada penelitian ini tiap video dipisah

menjadi 30 *Frame* per detik. Tiap *frame* akan dideteksi satu per satu oleh model YOLO.



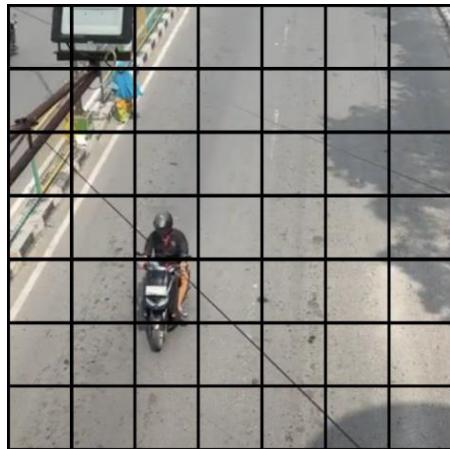
**Gambar 4. 19** Gambar *Input*

Gambar 4.19 adalah Frame yang telah dipisah menjadi 1 gambar dengan ukuran 1280 x 1024 *pixel*, Sebelum di proses lebih lanjut gambar akan di ubah menjadi ukuran 640 x 640 seperti pada gambar 4.20



**Gambar 4. 20** Gambar setelah *Resize* 640 x 640

Setelah didapatkan citra berukuran 640 x 640 dengan 3 channel proses selanjutnya adalah membagi citra kedalam kotak-kotak dengan ukuran 7 x 7 seperti yang ditunjukkan pada gambar 4.21



**Gambar 4. 21 Grid Cell 7 x 7**

kotak-kotak ini disebut sel grid dan setiap sel bertanggung jawab untuk memprediksi apakah terdapat objek di dalamnya atau tidak. Jika ada, akan diberi nilai 1 jika tidak diberi nilai 0. Kotak yang memiliki nilai 1 akan menghasilkan kotak pembatas (Bounding Box). Setiap sel terdiri dari beberapa kotak pembatas dengan 7 komponen pada setiap kotak ( $bx$ ,  $by$ ,  $bw$ ,  $bh$ , kepercayaan,  $pc0$ ,  $pc1$ ).

$bx$  = Merupakan koordinat horizontal (atau sumbu x)

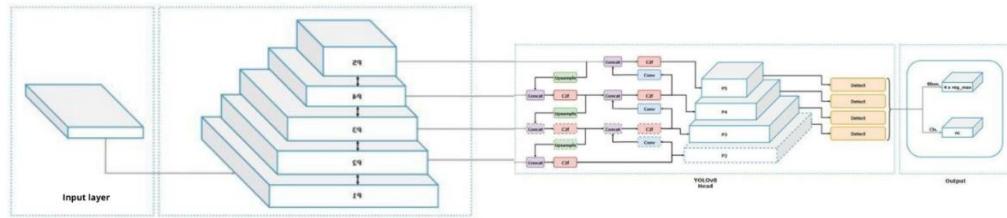
$by$  = Merupakan koordinat vertikal (atau sumbu y)

$bw$  = Merupakan lebar (width) dari kotak pembatas

$bh$  = Merupakan tinggi (height) dari kotak pembatas

$pc0$  dan  $pc1$  = adalah probabilitas atau skor yang diberikan oleh model untuk menunjukkan seberapa yakin model tersebut bahwa objek yang dideteksi adalah dari kelas tertentu.

#### 4.7.2 Arsitektur Yolov8



**Gambar 4. 22** Arsitektur YOLO

Berikut ini arsitektur YOLOv8 yang digunakan dalam penelitian ini yang di ambil dari *summary* saat pelatihan model.

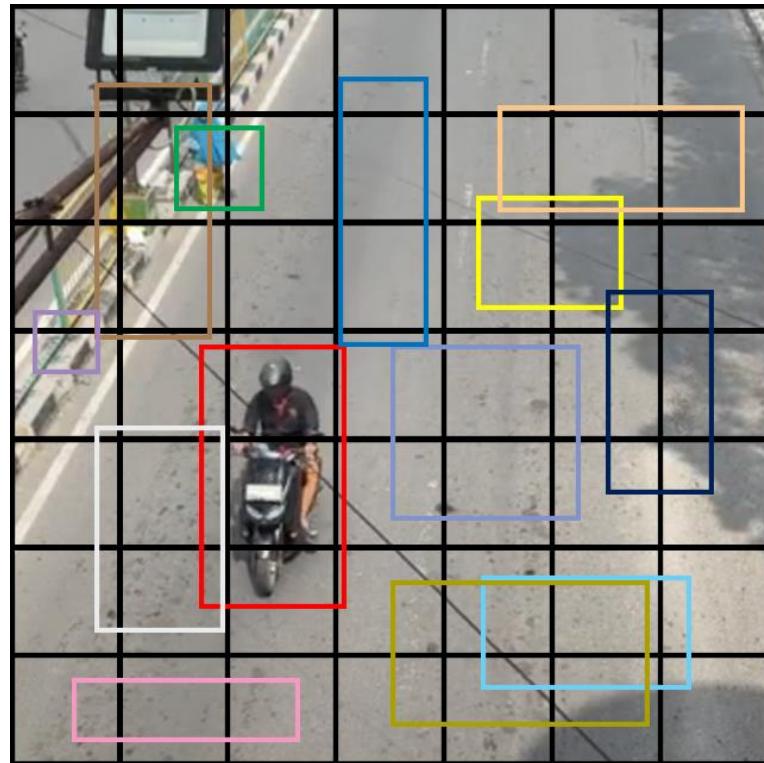
**Tabel 4. 7** Layer pada Model YOLOv8

Layer Type and Modules	In Channels	Out Channels	Kernel Size	Stride	Padding
Conv + BatchNorm2d	3	16	3x3	2	1
Conv + BatchNorm2d	16	32	3x3	2	1
C2f + BatchNorm2d + ModuleList (1x Bottleneck)	32	32	1x1	1	0
Conv + BatchNorm2d	32	64	3x3	2	1
C2f + BatchNorm2d + ModuleList (2x Bottleneck)	64	64	1x1	1	0
Conv + BatchNorm2d	64	128	3x3	2	1
C2f + BatchNorm2d + ModuleList (2x Bottleneck)	128	128	1x1	1	0
Conv + BatchNorm2d	128	256	3x3	2	1
C2f + BatchNorm2d + ModuleList (1x Bottleneck)	256	256	1x1	1	0
SPPF + BatchNorm2d + MaxPool2d (5x5, stride=1, padding=2)	256	128	1x1	1	0
Conv + BatchNorm2d	128	256	1x1	1	0

Upsample (scale_factor=2.0, mode='nearest')	-	-	-	-	-
Concat	-	-	-	-	-
C2f + BatchNorm2d + ModuleList (1x Bottleneck)	384	128	1x1	1	0
Upsample (scale_factor=2.0, mode='nearest')	-	-	-	-	-
Concat	-	-	-	-	-
C2f + BatchNorm2d + ModuleList (1x Bottleneck)	192	64	1x1	1	0
Conv + BatchNorm2d	64	64	3x3	2	1
Concat	-	-	-	-	-
C2f + BatchNorm2d + ModuleList (1x Bottleneck)	192	128	1x1	1	0
Conv + BatchNorm2d	128	128	3x3	2	1
Concat	-	-	-	-	-
C2f + BatchNorm2d + ModuleList (1x Bottleneck)	384	256	1x1	1	0
Detect	-	-	-	-	-

Berikut ini adalah penjelasan singkat mengenai arsitektur YOLO diatas. Pertama, gambar *input* berukuran 640 x 640 piksel dengan 3 saluran warna dimasukkan ke dalam model. Kemudian, Struktur jaringan ini dimulai dengan lapisan konvolusi pertama yang terdiri dari 3 *input channels* dan 16 *output channels* dengan *kernel* berukuran 3x3, *stride* 2, dan *padding* 1, diikuti dengan *batch normalization* dan fungsi aktivasi SiLU. *batch normalization* membuat pelatihan lebih efisien dan stabil dengan menormalisasi *input* untuk setiap lapisan, yang membantu dalam konvergensi lebih cepat. Sedangkan Fungsi SiLU adalah fungsi aktivasi yang membantu jaringan belajar nilai non-linearitas. Lapisan konvolusi ini diulang dengan konfigurasi yang berbeda pada lapisan-lapisan berikutnya, meningkatkan jumlah *output channels* secara bertahap, dari 32 hingga 256. Lapisan-lapisan C2f (CSP Bottleneck dengan 2 konvolusi) menggabungkan beberapa lapisan konvolusi, *batch normalization*, dan *Bottleneck modules* yang masing-masing terdiri dari

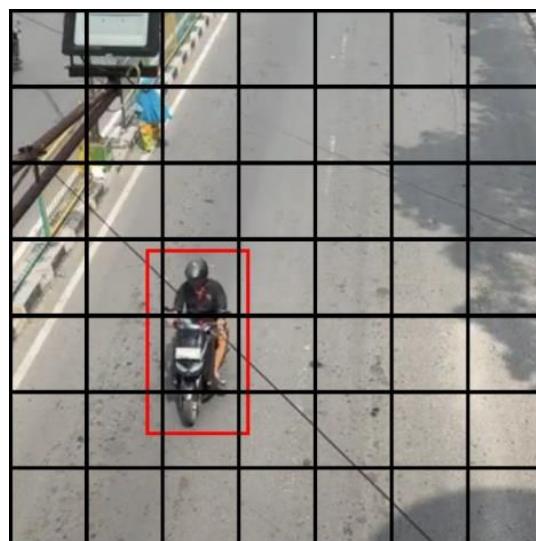
beberapa lapisan konvolusi tambahan. Misalnya, lapisan ketiga memiliki 32 *input channels* dan *output channels*, *kernel* 1x1, *stride* 1, dan *padding* 0, diikuti dengan *batch normalization* dan fungsi aktivasi SiLU. Pada lapisan ketujuh, terdapat lapisan SPPF (*Spatial Pyramid Pooling Fast*) yang mengkombinasikan informasi dari berbagai ukuran filter dengan pooling ukuran 5x5, *stride* 1, dan *padding* 2, diikuti dengan *batch normalization* dan fungsi aktivasi SiLU. Setelah itu, lapisan konvolusi dan *upsampling* digunakan untuk meningkatkan resolusi input dengan faktor skala tertentu, yang kemudian digabungkan dengan *input* dari lapisan sebelumnya menggunakan lapisan *Concat*. Deteksi objek dilakukan pada lapisan akhir, yang terdiri dari beberapa lapisan konvolusi dengan *batch normalization* dan fungsi aktivasi SiLU, yang disusun secara sekuensial. Seluruh jaringan ini dirancang untuk memproses gambar *input* melalui berbagai tahap konvolusi, normalisasi, aktivasi, *pooling*, dan penggabungan, sehingga menghasilkan deteksi objek yang akurat pada *output* akhir. Lapisan konvolusi bertanggung jawab untuk ekstraksi fitur, sementara lapisan C2f mengkombinasikan informasi dari berbagai skala untuk meningkatkan kemampuan jaringan dalam mengenali pola. Lapisan SPPF memungkinkan penggabungan informasi spasial dari berbagai ukuran filter, yang berguna untuk deteksi objek multi-skala. Lapisan *Upsample* berfungsi meningkatkan resolusi fitur yang diekstrak, memungkinkan jaringan untuk menangkap detail yang lebih halus. Semua komponen ini bekerja sama untuk membentuk jaringan yang kompleks namun efisien dalam mendeteksi dan mengenali objek dalam gambar.



**Gambar 4. 23** Pencarian *Bounding Box*

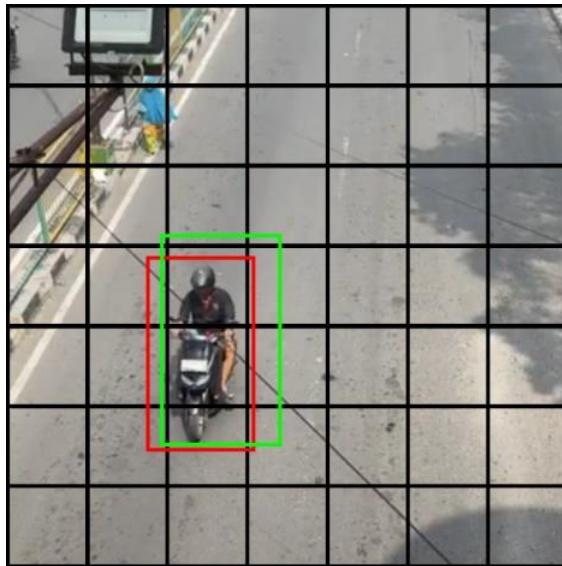
Proses pencarian *bounding box* terlihat dalam gambar 4.23, menunjukkan bahwa setiap sel *grid* bertanggung jawab untuk mencari *anchor box* dengan berbagai ukuran. Diketahui titik yang dilihat adanya objek terdapat pada kordinat Titik (x,y) = (197,402), Lebar (w) = 124, Tinggi (h) = 222

Untuk potongan gambar *bounding box* dengan koordinat (197,402) dapat dilihat pada gambar 4.24



**Gambar 4. 24** Prediksi *Bounding Box*

Gambar 4.24 Merupakan contoh objek yang didapatkan oleh *Bounding Box*. *Bounding box* yang digunakan adalah yang berwarna Merah pada ilustrasi dalam Gambar 4.25



**Gambar 4. 25** Prediksi IOU

Setelah ditemukan hasil prediksi model, dilakukan perhitungan IOU. langkah-langkah perhitungan *Intersection over Union* (IOU) dapat dihitung dengan cara sebagai berikut:

$$\text{area } (BB_{prediksi} \cap BB_{groundTruth}) = (138 \times 241) = 33258$$

$$\begin{aligned} \text{area } (BB_{prediksi} \cup BB_{groundTruth}) &= (124 \times 222) + (138 \times 241) = \\ 27528 + 33258 &= 60786 \end{aligned}$$

$$IOU = \frac{\text{area } (BB_{prediksi} \cap BB_{groundTruth})}{\text{area } (BB_{prediksi} \cup BB_{groundTruth})}$$

$$IOU = \frac{33258}{60786} = 0.54$$

Data *bounding box* yang diperoleh akan digunakan, Kemudian nilai 0.54 yang didapat akan digunakan sebagai skor kepercayaan (*confidence score*) pada *bounding box*. nilai akhir dari prediksi (kelas dari skor kepercayaan) akan didapat dengan mengalikan skor kepercayaan *bounding box* dengan probabilitas kelas. Probabilitas untuk setiap kelas menunjukkan kemungkinan bahwa objek pada *bounding box* termasuk dalam kelas tertentu, contohnya kelas "motors". Nilai probabilitas ini berkisar antara 0 hingga 1, di mana 1 menunjukkan bahwa objek pada *bounding box* termasuk dalam kelas tersebut, dan 0 menunjukkan sebaliknya.

## 4.8 Pengujian Sistem

Tahap pengujian dilakukan terhadap sistem dan berfokus terhadap pengujian pada sistem secara keseluruhan berdasarkan rancangan dan implementasi yang dilakukan. Ketentuan yang digunakan dalam pengujian ini meliputi:

1. Model YOLO dapat memanggil hasil tangkapan dari kamera.
2. Sistem bisa berjalan secara optimal sesuai dengan kondisi yang telah ditetapkan.
3. Menganalisis kinerja sistem berdasarkan hasil dari deteksi sistem.

### 4.8.1 Persiapan Pengujian sistem

Pada penelitian ini, sistem dikembangkan menggunakan metode deteksi objek YOLOv8. Sistem ini dirancang dengan menggunakan kamera esp32-cam untuk menangkap kondisi jalan secara langsung. Kemudian kondisi jalan yang ditangkap oleh kamera akan dikirimkan kepada sistem. Sistem dijalankan pada server lokal. Simulasi sistem pendekripsi kemacetan dilakukan pada Jalan Sisingamangaraja, Medan. Sistem mengambil data kondisi lalu lintas dari kamera ESP32-CAM.



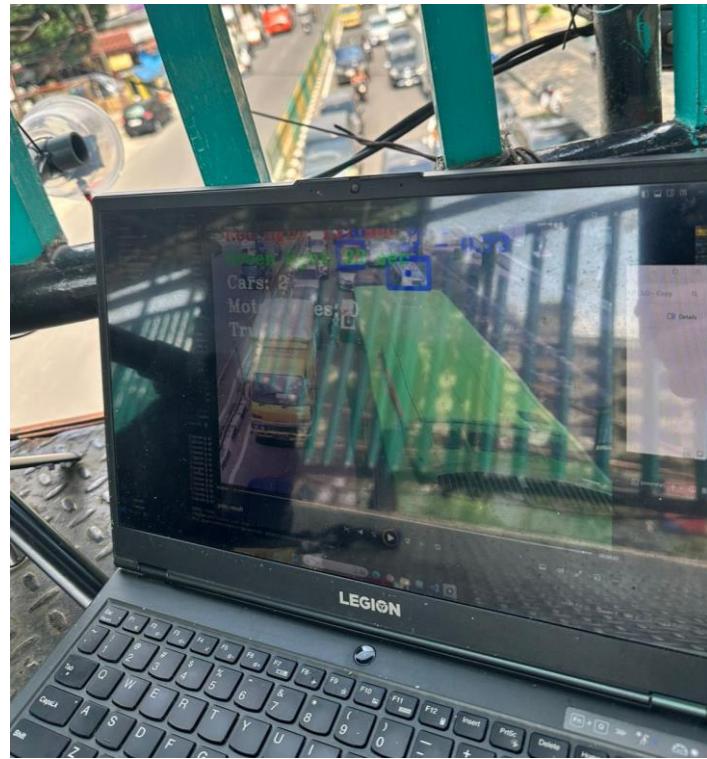
**Gambar 4. 26 Modul ESP32-CAM**

ESP32-CAM adalah modul kamera yang memungkinkan integrasi kemampuan pengolahan gambar dan konektivitas Wi-Fi/Bluetooth dalam satu perangkat kecil. Pada penelitian ini, ESP32-CAM berfungsi untuk menangkap kondisi lalu lintas pada jalan raya.



**Gambar 4. 27 Pengambilan kondisi lalu lintas**

Pada gambar di atas dilakukan peletakan kamera ESP32-CAM pada jembatan penyebrangan orang (JPO) pada Jalan Sisingamangaraja, Medan. JPO pada Jalan Sisingamangaraja berjarak 18 meter dari lampu merah. Simulasi Pengujian dilakukan ketika lampu lalu lintas berwarna hijau dan kendaraan yang dideteksi dalam posisi bergerak yang bertujuan untuk mengukur kecepatan kendaraan dan menentukan status lampu lalu lintas. Lokasi ini dipilih dikarenakan memiliki pijakan untuk purwarupa sistem dan jalannya penelitian tidak merugikan pengemudi yang sedang menggunakan jalan umum.

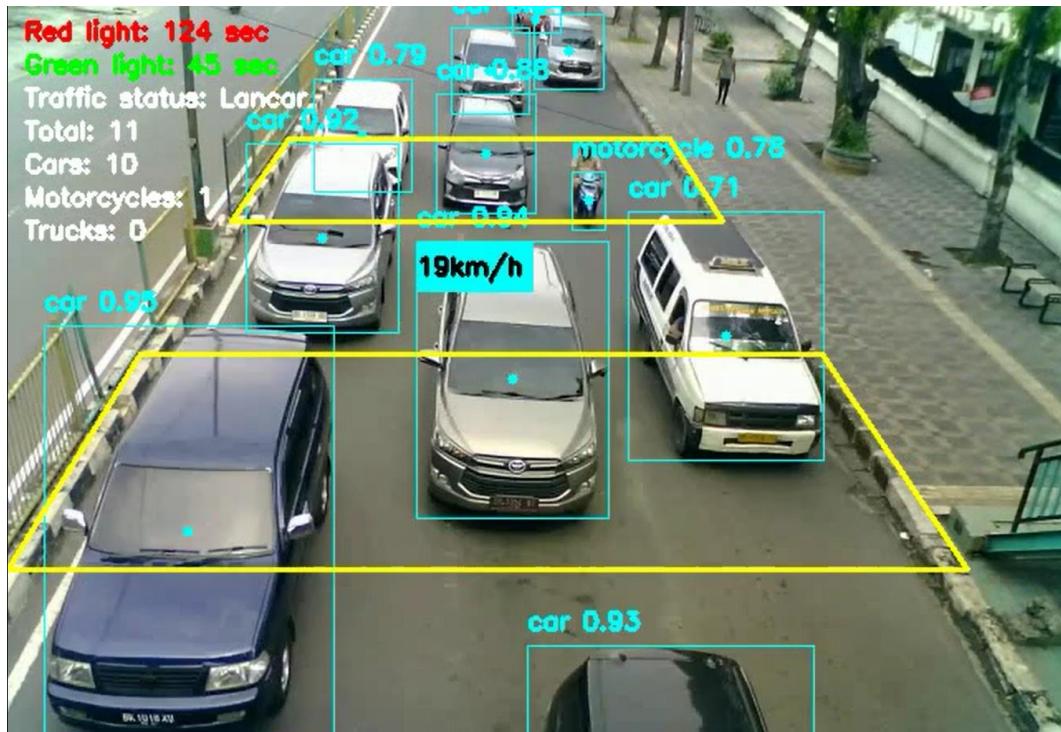


**Gambar 4. 28** *Output* dari sistem

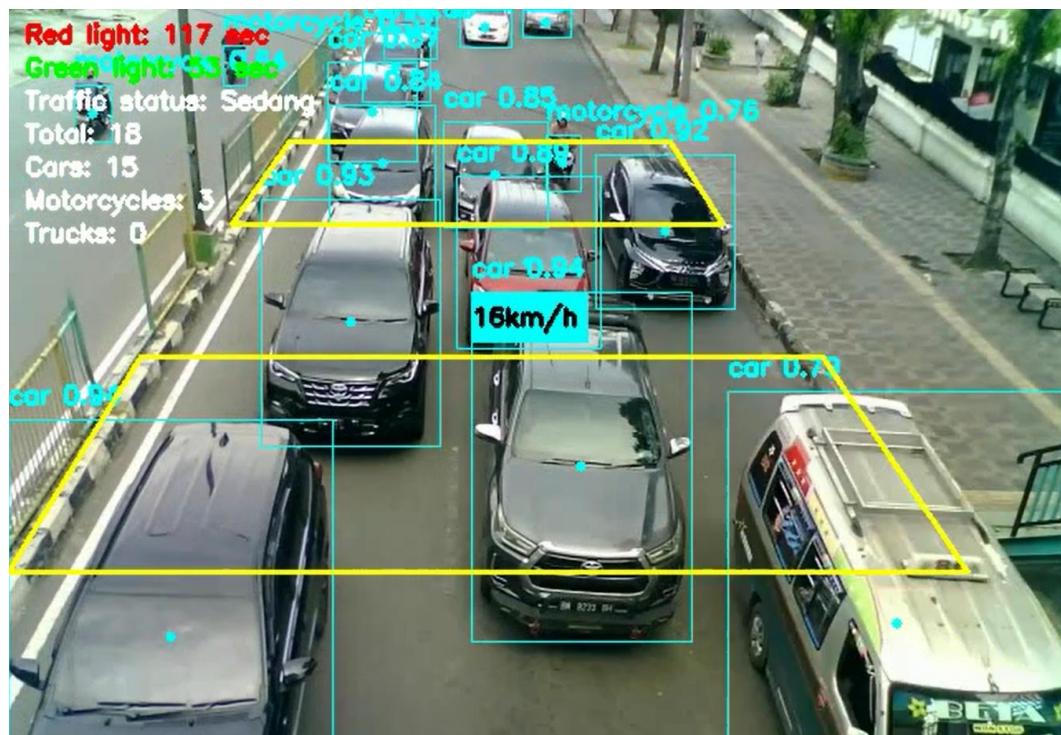
Kemudian setelah kondisi lalu lintas ditangkap kamera, sistem akan memanggil hasil tangkapan kamera untuk mendeteksi jumlah kendaraan yang ada. Setelah itu sistem akan mengeluarkan *output* berupa hasil dari deteksi serta durasi ideal lampu lalu lintas berdasarkan jumlah kendaraan yang sedang berhenti pada jalan dalam sudut pandang kamera, secara *real-time*. Pada penelitian ini konsep *real-time* yang ditawarkan yaitu pemrosesan yang langsung dilakukan oleh sistem saat hasil kamera telah ditangkap dan hasil dari sistem tidak diproses dalam waktu yang cukup lama.

#### **4.8.2 Hasil Pengujian Sistem**

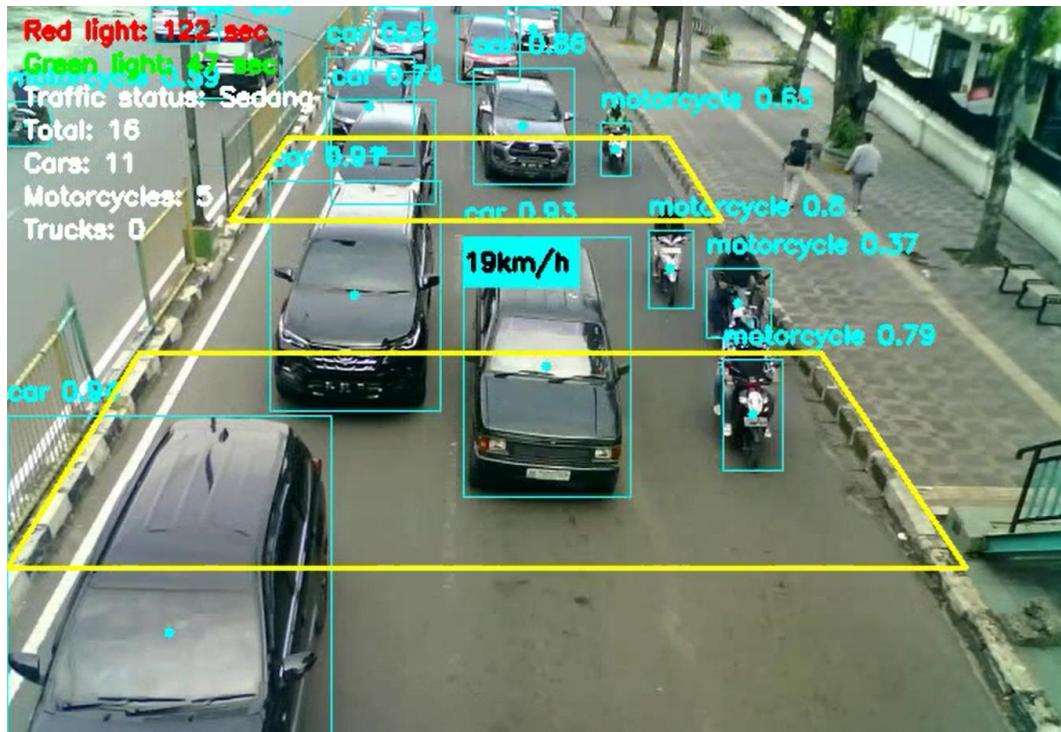
Tahap ini menampilkan hasil yang telah dideteksi oleh sistem guna memeriksa keandalan dari sistem yang telah dikembangkan. Tahap ini menganalisis kinerja sistem dengan melihat bagaimana produktivitas sistem secara keseluruhan.



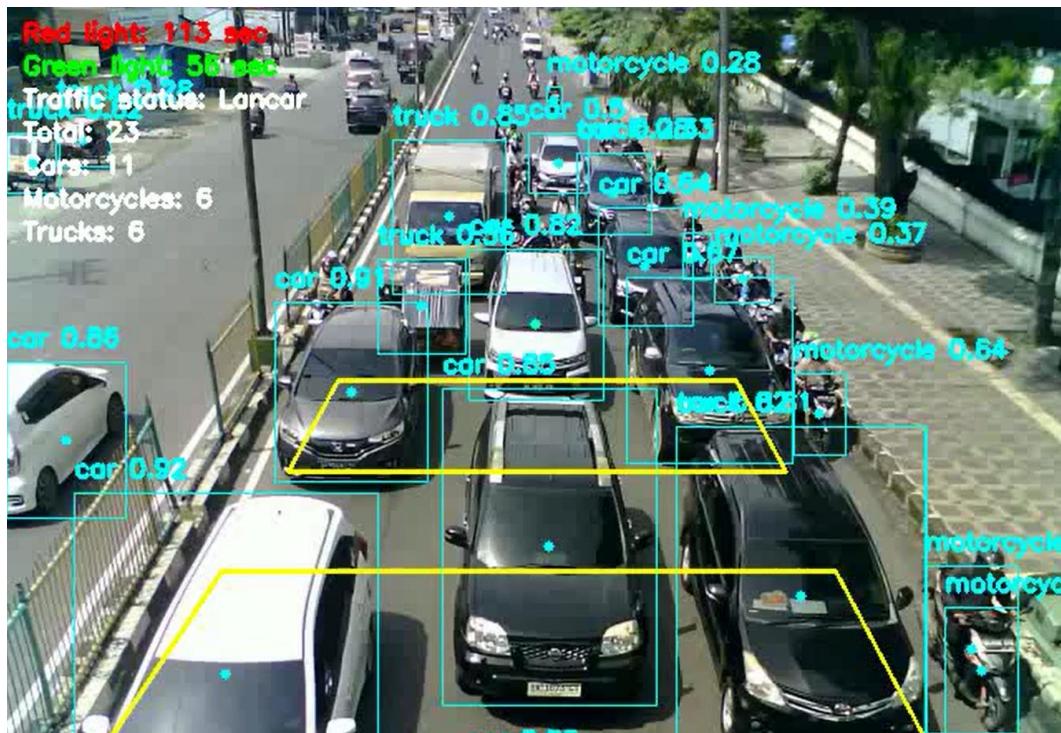
Gambar 4. 29 Hasil Uji 1



Gambar 4. 30 Hasil Uji 2



Gambar 4. 31 Hasil Uji 3

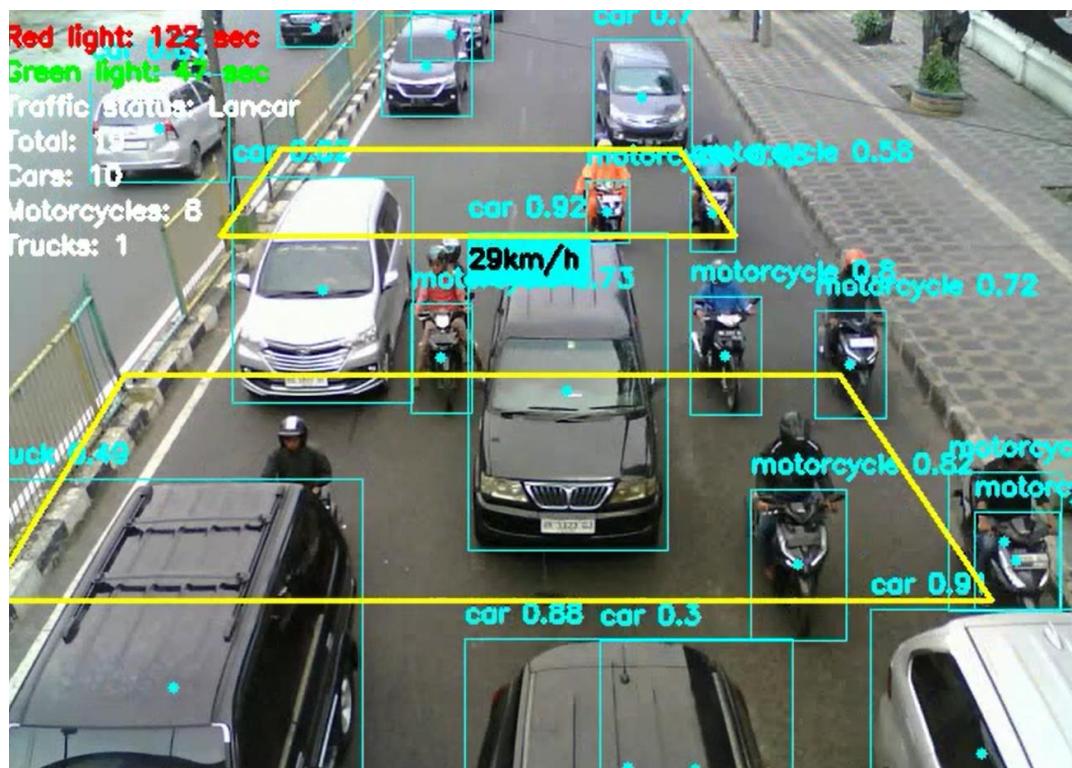


Gambar 4. 32 Hasil Uji 4

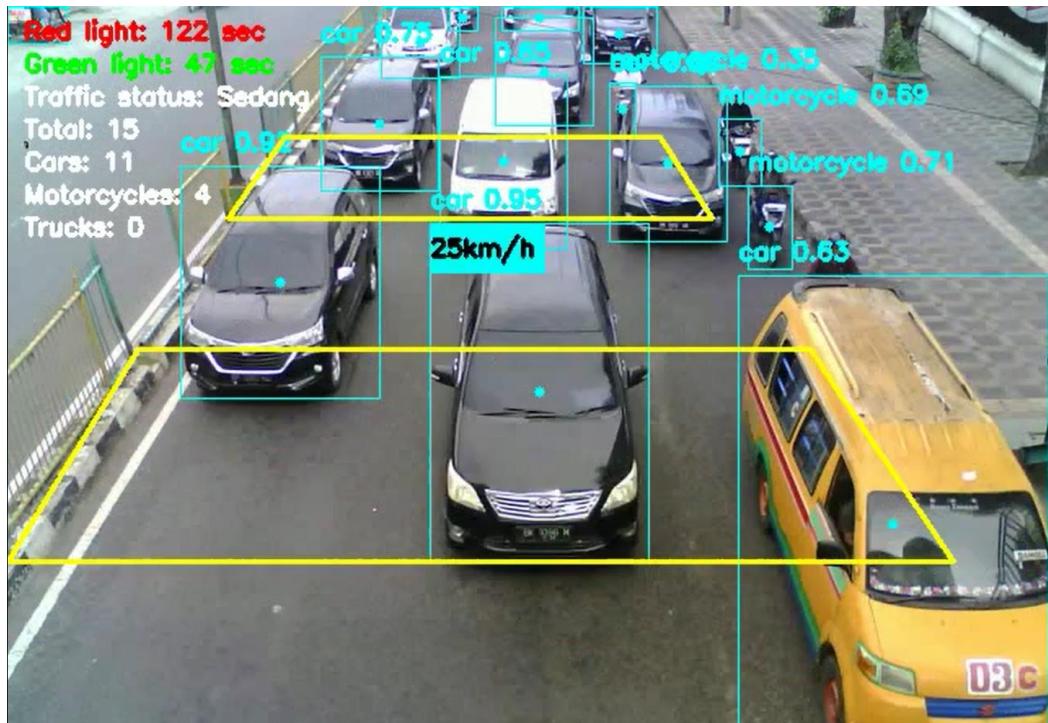
Pada gambar 4.32 di atas terdapat hasil dari deteksi kendaraan, akan tetapi sistem tidak dapat memperbarui status lalu lintas dikarenakan kendaraan dalam posisi berhenti.



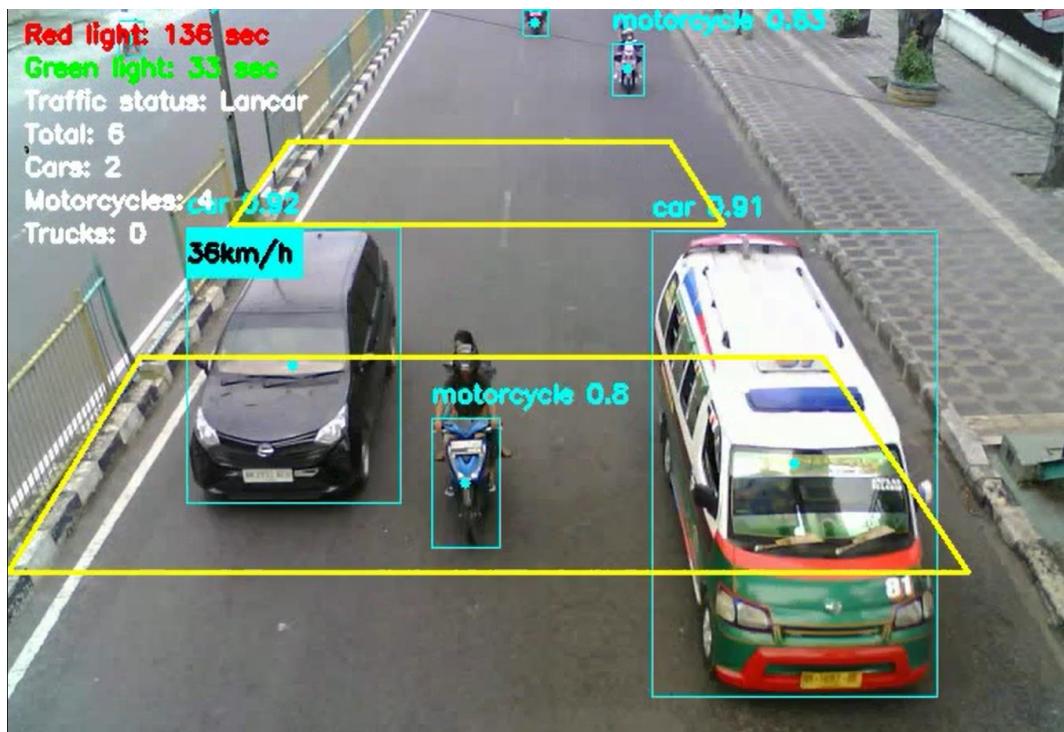
Gambar 4. 33 Hasil uji 5



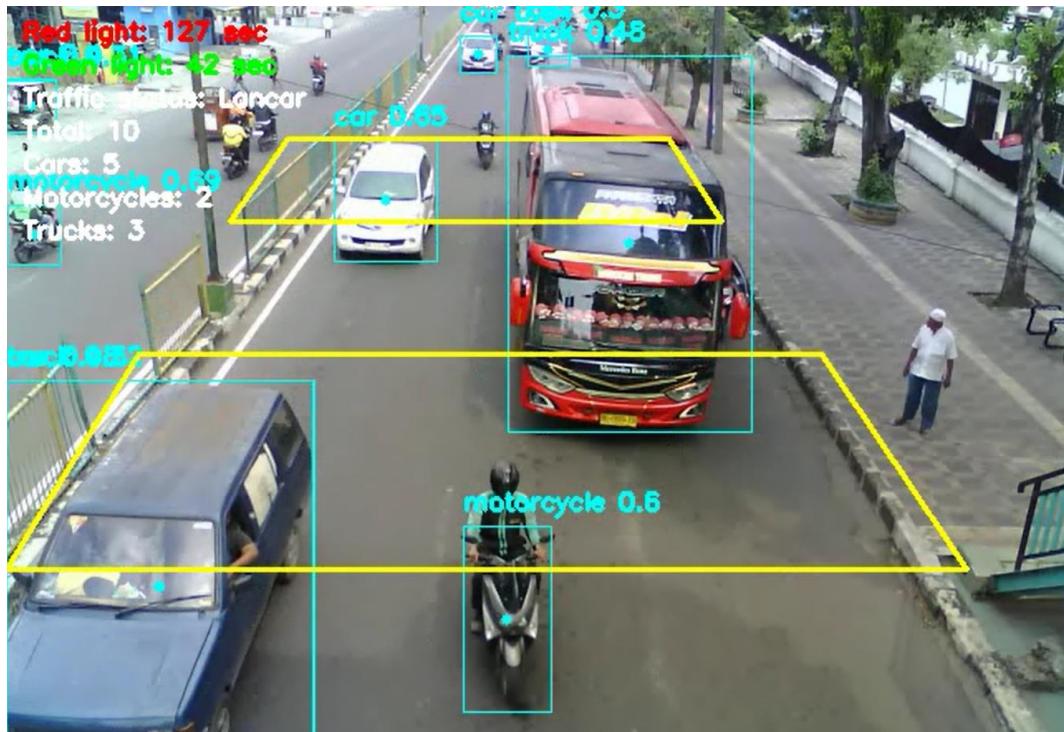
Gambar 4. 34 Hasil Uji 6



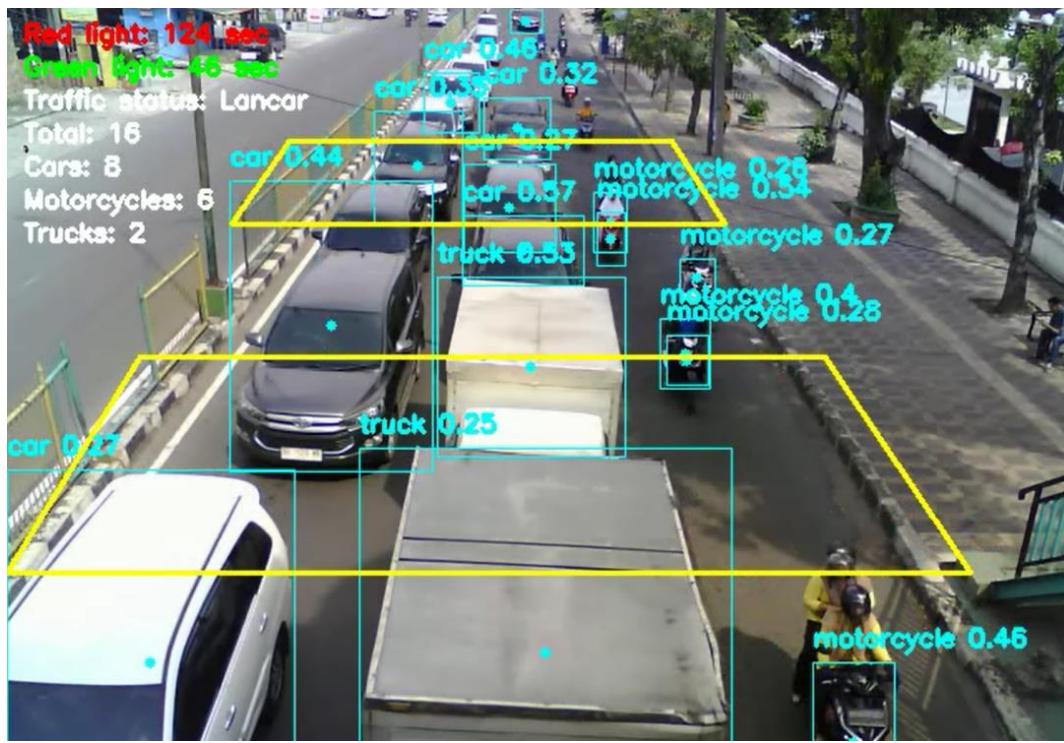
Gambar 4. 35 Hasil Uji 7



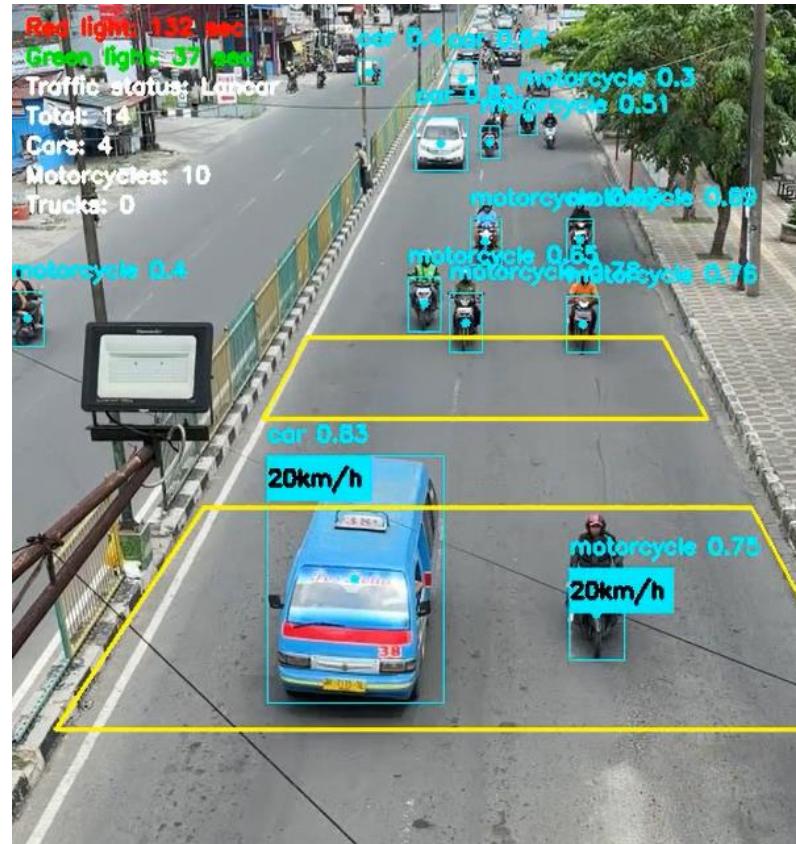
Gambar 4. 36 Hasil Uji 8



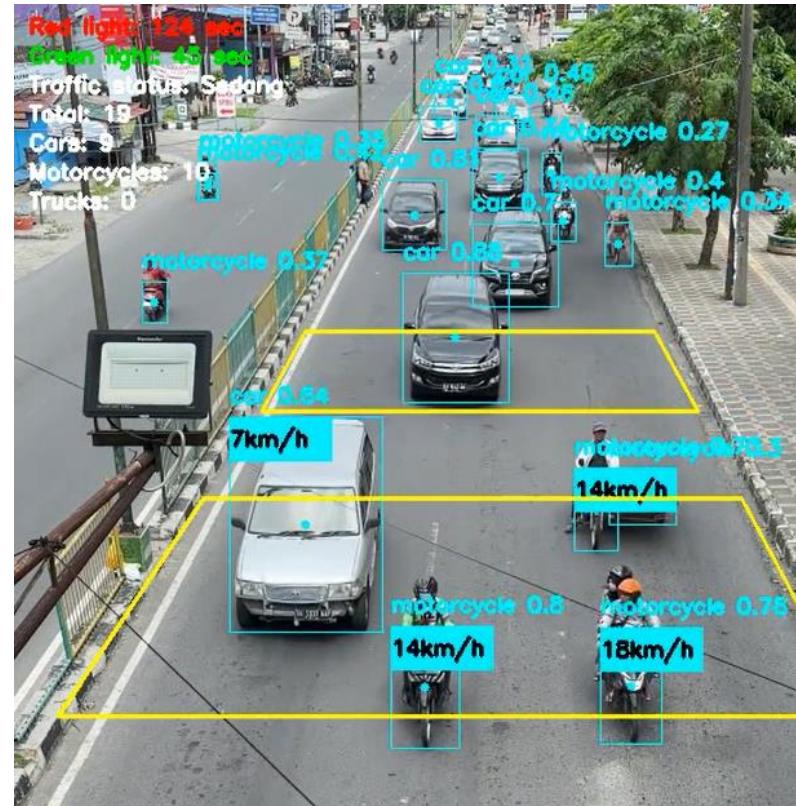
Gambar 4. 37 Hasil Uji 9



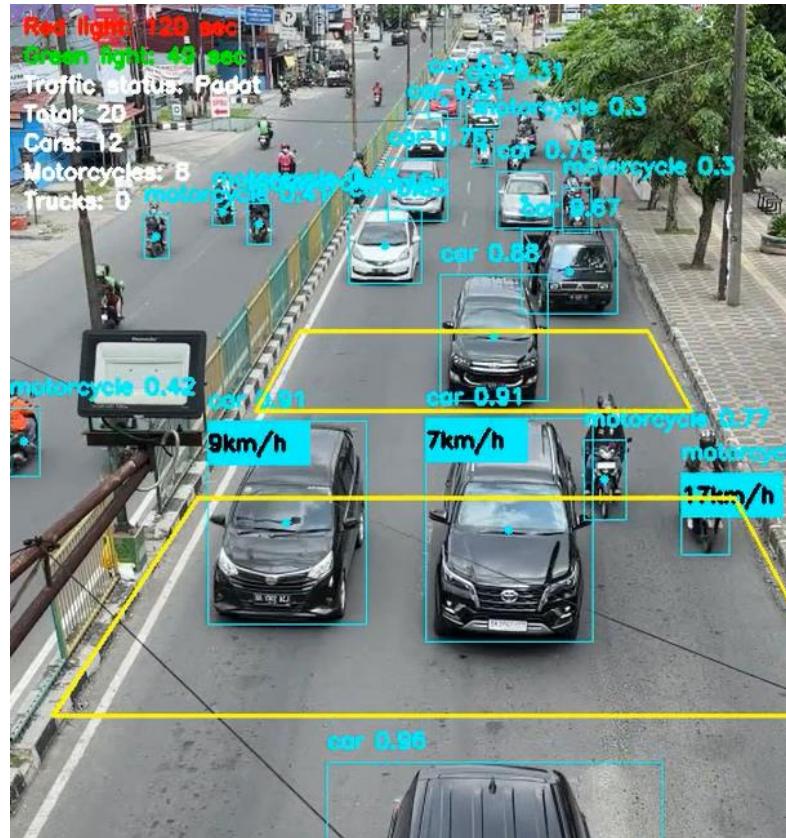
Gambar 4. 38 Hasil Uji 10



Gambar 4.39 Hasil Sampel 1



Gambar 4.40 Hasil Sampel 2



**Gambar 4. 41 Hasil Sampel 3**

Gambar di atas merupakan hasil deteksi dari algoritma YOLO (You Only Look Once) yang digunakan untuk menganalisa kendaraan pada jalan raya. Hasil deteksi ini digunakan untuk mengatur durasi lampu lalu lintas secara dinamis berdasarkan jumlah kendaraan dan status kendaraan yang terdeteksi. Kemudian ada pula hasil keluaran dari sistem yang berupa status lalu lintas berdasarkan kecepatan pada jalan. Parameter yang digunakan sebagai tolak ukur dari kemacetan pada penelitian ini berdasar dari ketentuan MKJI (Manual Kapasitas Jalan Indonesia). Menurut MKJI, suatu persimpangan dapat dikatakan terjadi kemacetan apabila volume suatu persimpangan jalan telah dipenuhi kendaraan dan kecepatan kendaraan mendekati 0 KM/J. MKJI (Manual Kapasitas Jalan Indonesia) adalah manual untuk kegiatan analisis, perencanaan, perancangan, dan operasi fasilitas lalu lintas jalan, yang merupakan produk hasil penelitian yang dilakukan oleh Direktorat jendral bina marga jalan kota, di beberapa lokasi yang dianggap mewakili kondisi karakteristik lalu lintas di wilayah Indonesia. Pada penelitian ini ketentuan yang digunakan untuk menentukan status macet adalah apabila terdapat terdapat lebih dari 20 kendaraan yang tertangkap oleh kamera dan kecepatan rata-rata kendaraan

dibawah 10 KM/J. Ketentuan lebih dari 20 kendaraan ini dikarenakan keterbatasan sudut pandang kamera dan resolusi yang diberikan oleh ESP32 hanya sebesar 2MP dan resolusi gambar terbesar pada 1280 x 1024 piksel. Pada gambar, kendaraan yang terdeteksi diberikan bounding box dengan label dan confidence score. Tampilan *display* di sebelah kiri atas gambar menunjukkan jumlah kendaraan yang terdeteksi, serta beberapa informasi tambahan terkait dengan pengaturan lampu lalu lintas.

Algoritma YOLO mendeteksi berbagai jenis kendaraan seperti mobil, truk, dan sepeda motor, kemudian menghitung total kendaraan yang berada pada jalan, serta menganalisa status lalu lintas. Dalam contoh gambar di atas, kita dapat melihat bahwa ada beberapa mobil, sepeda motor dan truk, yang terdeteksi. Akan tetapi ada beberapa kendaraan yang tidak dapat di deteksi oleh sistem. Faktor yang menyebabkan objek tidak dapat terdeteksi dengan baik adalah sebagai berikut:

1. resolusi kamera dari ESP-32 CAM yang cukup rendah yaitu hanya 2MP.
2. Pencahayaan yang kurang baik dikarenakan pencahayaan yang terlalu tinggi atau terlalu rendah juga mempengaruhi hasil dari deteksi sistem ini.
3. Objek kendaraan yang sedang berhenti berimpitan satu sama lain sehingga objek dideteksi oleh sistem sebagai *background*.
4. Posisi objek tidak sesuai atau terpotong oleh sudut pandang kamera ESP32-CAM.

Sistem lalu lintas berbasis deteksi kendaraan seperti ini dapat membantu dalam pengaturan lalu lintas yang lebih efisien. Sensor kamera yang terhubung ke algoritma deteksi kendaraan akan memantau dan menganalisis jumlah kendaraan dan status lalu lintas secara *real-time*. Berdasarkan analisis ini, sistem kontrol lalu lintas dapat menyesuaikan durasi lampu merah secara otomatis, mengoptimalkan aliran lalu lintas, dan mengurangi kemacetan. Dengan demikian, waktu tempuh kendaraan dapat diminimalisir, dan efisiensi lalu lintas dapat ditingkatkan.

Penggunaan teknologi seperti ini sangat berguna terutama di kota-kota besar dengan tingkat lalu lintas yang tinggi, di mana manajemen lalu lintas yang efisien diperlukan untuk mendukung mobilitas yang lancar dan mengurangi penumpukan kendaraan.

## **BAB 5**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

Berikut merupakan kesimpulan penelitian menggunakan YOLOv8 dalam pendekripsi kemacetan.

1. Penelitian ini menunjukkan bahwa algoritma YOLO (You Only Look Once) efektif dalam mendekripsi berbagai jenis kendaraan seperti mobil, sepeda motor, dan truk dengan hasil evaluasi mAP sebesar 0.849 untuk semua kelas dan nilai F1 confidence score sebesar 0.713.
2. YOLO mampu mendekripsi kendaraan dengan tingkat akurasi yang tinggi, meskipun terdapat beberapa kesalahan klasifikasi minor. Terutama jika kendaraan saling tumpang tindih dengan kendaraan lain dikarenakan keterbatasan posisi kamera ketika pengambilan video.
3. Penggunaan teknologi YOLO dalam manajemen lalu lintas dapat meningkatkan mobilitas dengan menganalisa kecepatan rata-rata kendaraan dan kepadatan jumlah kendaraan.
4. Sistem ini sangat berguna terutama di kota-kota besar dengan tingkat lalu lintas yang tinggi, membantu dalam pengaturan lalu lintas yang lebih efisien dan mengurangi penumpukan kendaraan.

#### **5.2 Saran**

Berikut adalah beberapa saran yang dapat dijadikan pertimbangan dalam penelitian selanjutnya:

1. Perlu dilakukan validasi menyeluruh terhadap model YOLOv8 yang diterapkan, baik melalui simulasi maupun uji lapangan, guna memverifikasi kinerja dan ketepatan deteksi kemacetan lalu lintas secara langsung.
2. Diperlukan pengumpulan data yang representatif dan beragam untuk melatih model YOLOv8 dengan baik. Data yang baik akan meningkatkan kemampuan model dalam mendekripsi berbagai situasi kemacetan lalu lintas.
3. Skripsi ini dapat mempertimbangkan integrasi model YOLOv8 dengan sistem manajemen lalu lintas yang ada untuk memperoleh informasi yang lebih komprehensif dan dapat digunakan secara praktis oleh pengelola lalu lintas.

4. Skripsi ini memiliki potensi untuk memulai penelitian lebih lanjut dalam pengembangan teknologi deteksi kemacetan lalu lintas yang lebih maju dan efisien, serta implementasi solusi tersebut dalam lingkup yang lebih luas.

## DAFTAR PUSTAKA

- Adji, W. A. K., Amalia, A., Herriyance, H., & Elizar, E. (2021). Abnormal Object Detection in Thoracic X-Ray Using You only Look Once (YOLO). *2021 International Conference on Computer System, Information Technology, and Electrical Engineering, COSITE 2021*, 118–123. <https://doi.org/10.1109/COSITE52651.2021.9649500>
- Anggraini, M. D., Kusrini, K., & Al Fatta, H. (2022). Social Distancing Detection Finding Optimal Angle With Yolo V3 Deep Learning Method. *Jurnal Teknik Informatika (Jutif)*, 3(5), 1449–1455. <https://doi.org/10.20884/1.jutif.2022.3.5.390>
- Arrahma, S. A., & Mukhaiyar, R. (2023). Pengujian Esp32-Cam Berbasis Mikrokontroler ESP32. *JTEIN: Jurnal Teknik Elektro Indonesia*, 4(1), 60–66.
- Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., & Pandya, S. (2021). *Cnn7.Pdf*. 1–28.
- HermaTelkomwan, M. I., Tritoasmoro, I. I., Ibrahim, N., Elektro, F. T., Universitas, Only, Y., Once, L., Online, S., & Tracking, R. (2021). *Traffic Light Control Based on Vehicle Density Using the Yolo Method*. 8(1), 198–205.
- Indra, D., Herman, H., & Budi, F. S. (2023). Implementasi Sistem Penghitung Kendaraan Otomatis Berbasis Computer Vision. *Komputika : Jurnal Sistem Komputer*, 12(1), 53–62. <https://doi.org/10.34010/komputika.v12i1.9082>
- Jupiyandi, S., Saniputra, F. R., Pratama, Y., Dharmawan, M. R., & Cholissodin, I. (2019). Pengembangan Deteksi Citra Mobil untuk Mengetahui Jumlah Tempat Parkir Menggunakan CUDA dan Modified YOLO. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, 6(4), 413. <https://doi.org/10.25126/jtiik.2019641275>
- Khairunnas, K., Yuniarno, E. M., & Zaini, A. (2021). Pembuatan Modul Deteksi Objek Manusia Menggunakan Metode YOLO untuk Mobile Robot. *Jurnal Teknik ITS*, 10(1). <https://doi.org/10.12962/j23373539.v10i1.61622>
- Kusuma, T. A. A. H., Usman, K., & Saidah, S. (2021). People Counting for Public Transportations Using You Only Look Once Method. *Jurnal Teknik Informatika (Jutif)*, 2(1), 57–66. <https://doi.org/10.20884/1.jutif.2021.2.2.77>

- Nugroho, B. A., & Djaksana, Y. M. (2022). Implementasi Mikrokontroler Arduino Uno dan Multi Sensor Pada Tempat Sampah. *Jurnal Scientia Sacra: Jurnal Sains*, 2(4), 70–77. <http://pijarpemikiran.com/index.php/Scientia>
- Prasanta, M. R., Pranata, M. Y., Firnanda, M. A., & Sendari, S. (2022). Rancang Bangun Quadcopter Drone Untuk Deteksi Api Menggunakan YOLOv4. *Cyclotron*, 5(1). <https://doi.org/10.30651/cl.v5i1.10013>
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-Decem*, 779–788. <https://doi.org/10.1109/CVPR.2016.91>
- Rofii, F., Priyandoko, G., Fanani, M. I., & Suraji, A. (2021). Vehicle Counting Accuracy Improvement By Identity Sequences Detection Based on Yolov4 Deep Neural Networks. *Teknik*, 42(2), 169–177. <https://doi.org/10.14710/teknik.v42i2.37019>
- Saputra, C. F. M., & Sulistyo, W. (2024). Alat Keamanan Depan Rumah Berbasis Internet of Things (IoT) Menggunakan ESP32-CAM yang Terintegrasi dengan Face Detection dan Telegram. *Jurnal JTIK (Jurnal Teknologi Informasi Dan Komunikasi)*, 8(1), 179–187. <https://doi.org/10.35870/jtik.v8i1.1259>
- Sarosa, M., & Muna, N. (2021). Implementasi Algoritma You Only Look Once ( Yolo ) Untuk Implementation of You Only Look Once ( Yolo ) Algorithm for. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, 8(4), 787–792. <https://doi.org/10.25126/jtiik.202184407>
- Selay, A., Andgha, G. D., Alfarizi, M. A., Bintang, M. I., Falah, M. N., Khaira, M., & Encep, M. (2022). Karimah Tauhid, Volume 1 Nomor 6 (2022), e-ISSN 2963-590X. *Karimah Tauhid*, 1(2963-590X), 861–862.
- Terven, J., Córdova-Esparza, D. M., & Romero-González, J. A. (2023). A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. *Machine Learning and Knowledge Extraction*, 5(4), 1680–1716. <https://doi.org/10.3390/make5040083>
- Turyadi, I. U. (2021). Analisa Dukungan Internet of Things (IoT) terhadap Peran Intelejen dalam Pengamanan Daerah Maritim Indonesia Wilayah Timur.

- Jurnal Teknologi Dan Manajemen Informatika*, 7(1), 29–39.  
<https://doi.org/10.26905/jtmi.v7i1.6040>
- Usen, Y. A., & Hayat, C. (2023). Design and Build Vehicle Plate Detection System Using You Only Look Once Method Based on Android. *Jurnal Teknik Informatika (Jutif)*, 4(4), 807–818.  
<https://doi.org/10.52436/1.jutif.2023.4.4.791>
- Vishwakarma, A. (2024). a Review: Machine Learning Algorithms. *Data Science: Practical Approach with Python & R*, October, 162–175.  
<https://doi.org/10.58532/nbennurch299>