



**UNIVERSITAS INDONESIA**

**Kuantifikasi Kepadatan Lalu Lintas pada Ruas Jalan Di  
Persimpangan dengan Metode Deteksi dan Pelacakan Objek  
Kendaraan Menggunakan YOLOv4, CSRT/KCF, dan OpenCV**

**SKRIPSI**

**Diajukan sebagai salah satu syarat untuk memperoleh gelar  
Sarjana Teknik**

**Muhammad Alfi Aldolio  
1806200015**

**FAKULTAS TEKNIK  
PROGRAM STUDI TEKNIK KOMPUTER**

**DEPOK  
JUNI 2022**



**UNIVERSITAS INDONESIA**

**Kuantifikasi Kepadatan Lalu Lintas pada Ruas Jalan Di  
Persimpangan dengan Metode Deteksi dan Pelacakan Objek  
Kendaraan Menggunakan YOLOv4, CSRT/KCF, dan OpenCV**

**SKRIPSI**

**Diajukan sebagai salah satu syarat untuk memperoleh gelar  
Sarjana Teknik**

**Muhammad Alfi Aldolio**

**1806200015**


**FAKULTAS TEKNIK  
PROGRAM STUDI TEKNIK KOMPUTER**

**DEPOK  
JUNI 2022**

## HALAMAN PERNYATAAN ORISINALITAS

**Skripsi ini adalah hasil karya saya sendiri,  
dan semua sumber baik yang dikutip maupun dirujuk  
telah saya nyatakan dengan benar.**

**Nama : Muhammad Alfi Aldolio**  
**NPM : 1806200015**

**Tanda Tangan : **  
**Tanggal : 9 Juni 2022**

## LEMBAR PENGESAHAN

Skripsi ini diajukan oleh:

Nama : Muhammad Alfi Aldolio

NPM : 1806200015

Program Studi : Teknik Komputer

Judul Skripsi : Kuantifikasi Kepadatan Lalu Lintas pada Ruas Jalan Di Persimpangan dengan Metode Deteksi dan Pelacakan Objek Kendaraan Menggunakan YOLOv4, CSRT/KCF, dan OpenCV

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana mata kuliah Skripsi pada Program Studi Teknik Komputer Fakultas Teknik Universitas Indonesia.

### DEWAN PENGUJI

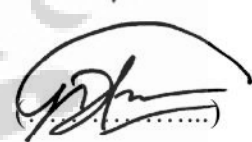
Pembimbing : Dr. Ruki Harwahu, ST. MT. MSc

()

Penguji 1 : Dr. Eng. Mia Rizkinia, S.T, M.T.

()

Penguji 2 : I Gde Dharma Nugraha, S.T., M.T., Ph.D

()

Ditetapkan di : Fakultas Teknik

Tanggal : 4 Juli 2022

## KATA PENGANTAR

Segala puji bagi Allah SWT yang telah memberikan banyak kenikmatan salah satunya adalah kesempatan untuk diri saya sebagai penulis berkembang dan mempelajari ilmu pengetahuan di bidang yang penulis minati. Rasa syukur juga dihanturkan atas kesempatan untuk berkuliah di Universitas Indonesia sebagai Mahasiswa Teknik Komputer yang kemudian berhasil membantu penulis untuk menyusun dan menulis buku skripsi berjudul **“Kuantifikasi Kepadatan Lalu Lintas pada Ruas Jalan Di Persimpangan dengan Metode Deteksi dan Pelacakan Objek Kendaraan Menggunakan YOLOv4, CSRT/KCF, dan OpenCV”**.

Tujuan penulis membuat buku skripsi ini adalah untuk memenuhi syarat menyelesaikan Program Sarjana (S1) Jurusan Teknik Komputer pada semester genap tahun ajaran 2021/2022 sebagai seorang mahasiswa Teknik Komputer di Fakultas Teknik Universitas Indonesia.

Dengan selesainya laporan ini, penulis juga ingin menyampaikan rasa terima kasih kepada:

- 1) Kedua orang tua penulis yang selalu memberikan dukungan dalam bentuk apapun.
- 2) Prof. Ruki Harwahu yang telah memberikan bimbingan selama proses pembuatan laporan ini.
- 3) Dr.Eng. Mia Rizkinia, S.T., M.T. sebagai pembimbing akademis yang telah memberikan informasi serta arahan selama penulis berkuliah di Universitas Indonesia.
- 4) Seluruh dosen pengajar di Teknik Komputer.
- 5) Rekan serta sahabat yang menjadi teman diskusi serta pelepas jenuh.

Depok, Juni 2022



Muhammad Alfi A

**HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI KARYA  
ILMIAH UNTUK KEPENTINGAN AKADEMIS**

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan dibawah ini:

Nama : Muhammad Alfi Aldolio  
NPM : 1806200015  
Program Studi : Teknik Komputer  
Departemen : Teknik Elektro  
Fakultas : Teknik  
Jenis Karya : Skripsi

demikian perkembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia Hak Bebas Royalti Non-Eksklusif (Non-Exclusive Royalty-Free Right) atas karya ilmiah saya yang berjudul:

**Kuantifikasi Kepadatan Lalu Lintas pada Ruas Jalan Di Persimpangan  
dengan Metode Deteksi dan Pelacakan Objek Kendaraan Menggunakan  
YOLOv4, CSRT/KCF, dan OpenCV**

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non-Eksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan mempublikasikan skripsi saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di: Depok

Pada tanggal: 7 Juli 2022

Yang menyatakan



(MUHAMMAD ALFI ALDOLIO)

## ABSTRAK

Nama : Muhammad Alfi A  
Program Studi : Teknik Komputer  
Judul : Kuantifikasi Kepadatan Lalu Lintas pada Ruas Jalan Di  
Persimpangan dengan Metode Deteksi dan Pelacakan Objek  
Kendaraan Menggunakan YOLOv4, CSRT/KCF, dan OpenCV

Kamera digunakan oleh pengelola jalur lalu lintas kendaraan di jalan besar pada beberapa kota-kota di Indonesia untuk tujuan pengawasan, pengendalian, hingga pengambilan bukti pelanggaran atau kronologi kecelakaan pada ruas jalan tersebut. Namun, penggunaan kamera ini masih dalam tahap menerima data yang kemudian tindakan selanjutnya dilakukan secara manual. Sedangkan, banyak kebutuhan yang dapat dipenuhi dengan adanya perkembangan teknologi pada bidang komputasi. Salah satunya adalah penggunaan kamera tersebut dapat dimaksimalkan dengan mengimplementasikan algoritma pembelajaran mesin untuk menentukan jalur lalu lintas mana yang menjadi prioritas pada persimpangan dengan kuantifikasi kepadatan kendaraan pada ruas jalan.

Pada penelitian ini sistem kuantifikasi kepadatan kendaraan melalui data gambar dikaji dengan menggunakan algoritma untuk mendeteksi objek kendaraan pada seperti YOLOv4 yang merupakan *state-of-the-art* dalam algoritma pendeteksian karena memiliki akurasi yang lebih baik dan juga lebih cepat dibandingkan dengan arsitektur deteksi objek lainnya. Selain itu, diimplementasikan juga algoritma pelacakan objek kendaraan seperti CSRT/KCF sehingga tidak perlu melakukan proses deteksi secara terus-menerus dan dapat mengurangi biaya komputasi. Hasil percobaan pada penelitian ini membuktikan bahwa kombinasi model deteksi dan pelacakan dapat digunakan secara *real-time* maupun interaktif. Walaupun nilai mAP dari model YOLOv4 mengalami penurunan sekitar 20.65%, namun perbedaan antara hasil kuantifikasi kepadatan kendaraan sistem dan nilai aktual masih tidak terlalu jauh yaitu sekitar 1-5%, tergantung dengan jenis model yang digunakan.

Kata Kunci:

*Computer Vision; manajemen lalu lintas, YOLOv4, CSRT, KCF, OpenCV, AI*

## ABSTRACT

Name : Muhammad Alfi A  
Study Program : Computer Engineering  
Title : *Quantification of Traffic Density on Road Sections at Crossroads with Vehicle Object Detection and Tracking Methods Using YOLOv4, CSRT/KCF, and OpenCV*

*Cameras are used by traffic lane managers on major roads in several cities in Indonesia for the purpose of monitoring, controlling, and collecting evidence of violations or chronology of accidents on those roads. However, the use of this camera is still in the stage of receiving data, then further actions are carried out manually. Meanwhile, many needs can be met with the development of technology in the field of computing. One of them is that the use of the camera can be maximized by implementing machine learning algorithms to determine which traffic lanes are the priority at intersections by quantifying the density of vehicles on the road.*

*In this study, the vehicle density quantification system through image data will be studied using an algorithm to detect vehicle objects such as YOLOv4 which is a state-of-the-art detection algorithm because it has better accuracy and is also faster than other object detection architectures. In addition, vehicle object tracking algorithms such as CSRT/KCF will also be implemented so that there is no need to carry out the detection process continuously and can reduce computational costs. To meet the needs of image data processing from the video as well as the configuration of the AI model, one of the libraries, namely OpenCV, will be used to facilitate the creation and optimization of machine learning models/algorithms. This research proves that the combination of detection and tracking models can be used in real-time or interactively. Although the mAP value of the YOLOv4 model has decreased by about 20.65%, the difference between the system vehicle density quantification results and the actual value is still not too far away, around 1-5%, depending on the type of model used.*

Keywords:

*Computer Vision; traffic management, YOLOv4, CSRT, KCF, OpenCV, AI*



## DAFTAR ISI

HALAMAN PERNYATAAN ORISINALITAS .....	iii
LEMBAR PENGESAHAN .....	iv
KATA PENGANTAR.....	v
ABSTRAK .....	vii
ABSTRACT .....	viii
DAFTAR ISI.....	ix
DAFTAR GAMBAR.....	xi
DAFTAR FORMULA .....	xiii
DAFTAR TABEL .....	xiv
<b>BAB 1 PENDAHULUAN .....</b>	<b>15</b>
1.1. Latar Belakang .....	15
1.2. Rumusan Masalah .....	17
1.3. Tujuan Penelitian .....	17
1.4. Batasan Masalah.....	18
1.5. Metodologi Penelitian .....	18
1.6. Sistematika Penulisan .....	20
<b>BAB 2 LANDASAN TEORI .....</b>	<b>21</b>
2.1. <i>Artificial Intelligence</i> .....	21
2.1.1. <i>Deep Learning Pipeline</i> .....	21
2.1.2. Computer Vision .....	23
2.1.3. You Only Look Once (YOLOv4) .....	26
2.1.4. Channel and Spatial Reliability Tracking (CSRT).....	29
2.1.5. Kernelized Correlation Filter (KCF) .....	30
2.1.6. Euclidean Distance.....	31
2.2. <i>Framework dan Library</i> .....	32
2.2.1. Python .....	32
2.2.2. OpenCV .....	33
2.2.3. CUDA dan cuDNN .....	33
2.3. Metrik Evaluasi .....	33
2.3.1. <i>Intersection over Union (IoU)</i> .....	34
2.3.2. <i>Mean Average Precision (mAP)</i> .....	34
2.3.3. <i>Frame Rate per Second (FPS)</i> .....	36
2.3.4. <i>Traffic Density</i> .....	36

2.4.	Lalu Lintas Kendaraan di Ruas Jalan.....	37
2.4.1.	Jenis-jenis Kendaraan.....	37
2.4.2.	<i>Area of Interests</i> (AoIs) / Region.....	38
<b>BAB 3 PERANCANGAN SISTEM KUANTIFIKASI KEPADATAN LALU LINTAS DENGAN METODE DETEKSI DAN PELACAKAN OBJEK KENDARAAN DI RUAS JALAN .....</b>		<b>39</b>
3.1.	Rancangan Kebutuhan Sistem.....	39
3.1.1.	Kebutuhan untuk Perangkat Keras.....	40
3.1.2.	Kebutuhan untuk Perangkat Lunak.....	40
3.2.	Rancangan Sistem .....	44
3.2.1.	Sistem Deteksi Objek Kendaraan.....	45
3.2.2.	Sistem Pelacakan Objek Kendaraan.....	48
3.2.3.	Metode Kalkulasi Kepadatan pada Ruas Jalan .....	49
3.3.	Rancangan Analisis dan Evaluasi Sistem .....	49
3.3.1.	Evaluasi Akurasi Model dengan mAP .....	50
3.3.2.	Evaluasi Sistem dalam Mengkalkulasi Kepadatan Kendaraan ..	50
3.3.3.	Evaluasi Kecepatan Sistem dalam Memproses Citra dengan FPS	50
<b>BAB 4 IMPLEMENTASI DAN ANALISIS PERFORMA SISTEM KUANTIFIKASI KEPADATAN LALU LINTAS DENGAN METODE DETEKSI DAN PELACAKAN OBJEK KENDARAAN DI RUAS JALAN</b>		<b>51</b>
4.1.	Implementasi Sistem .....	51
4.2.	Pengujian Akurasi Model Deteksi Objek Kendaraan .....	53
4.2.1.	Model YOLOv4 .....	53
4.2.2.	Model YOLOv4-tiny.....	57
4.3.	Pengujian Performa Sistem .....	60
4.3.1.	Skenario 1: Pengujian Sistem Dengan YOLOv4 dan CSRT .....	61
4.3.2.	Skenario 2: Pengujian Sistem Dengan YOLOv4 dan KCF .....	64
4.3.3.	Skenario 3: Pengujian Sistem Dengan YOLOv4-tiny dan CSRT	66
4.3.4.	Skenario 4: Pengujian Sistem Dengan YOLOv4-tiny dan KCF	68
<b>BAB 5 KESIMPULAN DAN SARAN .....</b>		<b>70</b>
<b>DAFTAR PUSTAKA .....</b>		<b>72</b>

## DAFTAR GAMBAR

Gambar 2-1 <i>Deep Learning Pipeline</i> [7] .....	22
Gambar 2-2 Arsitektur Model CNN [9].....	24
Gambar 2-3 Gambar RGB dengan ukuran 4x4 <i>pixel</i> [9] .....	24
Gambar 2-4 Arsitektur YOLO [11].....	26
Gambar 2-5 Hasil deteksi YOLO [11] .....	28
Gambar 2-6 Proses CSRT dalam <i>Tracking</i> Objek [12] .....	29
Gambar 2-7 Jarak antara Dua Titik pada Bidang 2D .....	31
Gambar 2-8 <i>Intersection over Union</i> (IoU) .....	34
Gambar 2-9 Kendaraan di Ruas Jalan.....	37
Gambar 2-10 Region pada Ruas Jalan .....	38
Gambar 3-1 Blok Diagram Sistem.....	39
Gambar 3-2 <i>File</i> Konfigurasi Model YOLOv4 .....	41
Gambar 3-3 Cuplikan Citra dari Kamera CCTV pada Ruas Jalan.....	42
Gambar 3-4 Tampilan Antarmuka Perangkat Lunak LabelImg .....	43
Gambar 3-5 Hasil Anotasi dengan Menggunakan Perangkat Lunak LabelImg ...	43
Gambar 3-6 Diagram Alir Rancangan Dasar Sistem .....	44
Gambar 3-7 Blok Diagram Sistem Deteksi Objek Kendaraan.....	45
Gambar 3-8 Pencegahan Kelemahan Sistem Deteksi Objek dengan <i>Euclidean Distance</i> .....	47
Gambar 3-9 Blok Diagram Sistem Pelacakan Objek Kendaraan.....	48
Gambar 4-1 Citra dari 4 CCTV yang berbeda .....	51

Gambar 4-2 Perbandingan Nilai FP dan TP Objek untuk Model YOLOv4 .....	55
Gambar 4-3 Grafik Nilai AP dan mAP dari Hasil Deteksi YOLOv4 .....	55
Gambar 4-4 Hasil Deteksi YOLOv4 pada Setiap Citra CCTV.....	56
Gambar 4-5 Perbandingan Nilai FP dan TP Objek untuk Model YOLOv4-tiny..	58
Gambar 4-6 Grafik Nilai AP dan mAP dari Hasil Deteksi YOLOv4-tiny .....	58
Gambar 4-7 Hasil Deteksi YOLOv4-tiny pada Setiap Citra CCTV .....	59
Gambar 4-8 Grafik Perbandingan antara Nilai Kepadatan Kendaraan Hasil Prediksi Sistem dan Nilai Aktual untuk Skenario 1 .....	62
Gambar 4-9 Grafik Nilai FPS dan Kepadatan Kendaraan yang Dicuplik setiap 5 detik untuk Skenario 1 .....	63
Gambar 4-10 Grafik Perbandingan antara Nilai Kepadatan Kendaraan Hasil Prediksi Sistem dan Nilai Aktual untuk Skenario 2.....	64
Gambar 4-11 Grafik Nilai FPS dan Kepadatan Kendaraan yang Dicuplik setiap 5 detik untuk Skenario 2 .....	65
Gambar 4-12 Grafik Perbandingan antara Nilai Kepadatan Kendaraan Hasil Prediksi Sistem dan Nilai Aktual untuk Skenario 3.....	66
Gambar 4-13 Grafik Nilai FPS dan Kepadatan Kendaraan yang Dicuplik setiap 5 detik untuk Skenario 3 .....	67
Gambar 4-14 Grafik Perbandingan antara Nilai Kepadatan Kendaraan Hasil Prediksi Sistem dan Nilai Aktual untuk Skenario 4.....	68
Gambar 4-15 Grafik Nilai FPS dan Kepadatan Kendaraan yang Dicuplik setiap 5 detik untuk Skenario 4 .....	69

## DAFTAR FORMULA

Formula 2-1 Rumus <i>Linear Correlation Filter</i> .....	30
Formula 2-2 Rumus <i>Kernelized Correlation Filter</i> (KCF).....	31
Formula 2-3 Perhitungan <i>Euclidean Distance</i> .....	32
Formula 2-4 Rumus <i>Precision</i> .....	35
Formula 2-5 Rumus <i>Recall</i> .....	35
Formula 2-6 Rumus Interpolasi 11 Titik <i>Recall</i> .....	35
Formula 2-7 Rumus <i>Average Precision</i> [19] .....	35
Formula 2-8 Perhitungan <i>Frame Rate per Second</i> .....	36
Formula 2-9 Perhitungan <i>Traffic Density</i> .....	36
Formula 3-1 Perhitungan Koordinat Titik Tengah, Panjang, dan Lebar Objek pada Citra.....	46
Formula 3-2 Perhitungan Titik <i>Anchor</i> dan Titik Pojok Kanan Bawah <i>Bounding Box</i> .....	46

## DAFTAR TABEL

Tabel 4-1 Spesifikasi Perangkat untuk Pengujian Sistem.....	52
Tabel 4-2 Nilai AP dan mAP dari YOLOv4 untuk Setiap CCTV .....	54
Tabel 4-3 Nilai AP dan mAP dari YOLOv4-tiny untuk Setiap CCTV.....	57
Tabel 4-4 Nilai Aktual Kuantifikasi Kepadatan Lalu Lintas untuk Setiap CCTV	60



# BAB 1

## PENDAHULUAN

### 1.1. Latar Belakang

Sistem kuantifikasi kepadatan lalu lintas yang mengandalkan algoritma AI ini dikaji karena adanya perencanaan pemindahan Ibu Kota Indonesia ke Kalimantan Timur yang menerapkan konsep dari *Smart City* [1]. Pengelolaan lalu lintas yang efisien, efektif, dan aman menjadi poin penting dalam pengembangan *Smart City*. Salah satu alat yang dapat digunakan untuk pengelolaan lalu lintas adalah kamera CCTV yang dapat menangkap citra gambar di ruas-ruas jalan persimpangan. Kamera CCTV tersebut dapat digunakan untuk mencegah terjadinya tindak kejahatan di jalan raya, serta untuk memantau volume kendaraan yang masuk atau keluar dari ruas jalan tersebut. Namun, kebanyakan dari sistem ini masih menggunakan cara kerja konvensional, dimana pengawasan dan tindakan dilakukan secara manual oleh pekerja dan pengatur lampu lalu lintas sendiri. Sistem juga masih bekerja berdasarkan waktu tetap, tanpa mempertimbangkan kepadatan lalu lintas pada ruas jalan. [2]

Dengan potensi yang dimiliki oleh komputer sekarang, manajemen lalu lintas dapat diotomatisasi dengan menggunakan *Artificial Intelligence* (AI). AI sendiri adalah sebuah algoritma yang dapat mempelajari pola yang ada pada suatu data sehingga dapat mensimulasikan kemampuan manusia dalam mengambil suatu keputusan serta memecahkan suatu permasalahan yang biasanya ada dalam bentuk regresi, klasifikasi, atau segmentasi. Dalam kasus kuantifikasi kepadatan lalu lintas dengan mengandalkan fasilitas kamera CCTV, maka algoritma AI yang digunakan harus dapat mengolah citra gambar yang ditangkap dalam bentuk video oleh kamera. Hal tersebut membuat algoritma AI dan perangkat komputasi yang digunakan harus dapat memproses gambar dengan cepat.

Salah satu algoritma AI yang banyak digunakan untuk keperluan deteksi objek adalah algoritma/arsitektur *You Only Look Once, Version 4* (YOLOv4). YOLOv4 merupakan *state-of-the-art* dalam bidang deteksi objek pada gambar secara *real-time*. Arsitektur ini menggunakan *neural networks* untuk melakukan

deteksi objek dan menjadi populer karena kecepatan dan juga akurasi. Performa dari YOLOv4 juga dapat ditingkatkan 500 kali lebih cepat bila diproses menggunakan *Graphics Processing Unit* (GPU) NVIDIA melalui CUDA. Pada uji coba dengan *dataset* COCO yang mempunyai 80 label data objek, arsitektur versi YOLOv4-p6 mendapatkan *mean average precision* (mAP) sebesar 72.1% dengan 32 FPS, sedangkan untuk arsitektur YOLOv4-tiny yang lebih mengutamakan pada poin kecepatan deteksi objeknya mendapatkan mAP sebesar 40.2% dengan 330 FPS. [3]

Arsitektur YOLOv4 dapat diimplementasikan dalam dua bahasa pemrograman yaitu C++ dan Python dengan menggunakan *Open-Source Computer Vision Library* (OpenCV) sebagai *backend*. OpenCV merupakan salah satu *framework* yang bersifat sebagai *library* yang dapat mempermudah proses manipulasi gambar dan juga pengolahannya. YOLOv4 terintegrasi dengan OpenCV untuk memproses suatu gambar dan juga melakukan pendeteksian objek. Kombinasi dari keduanya dapat digunakan untuk menjadi entitas utama dalam mendeteksi jenis-jenis kendaraan yang ada pada ruas jalan. Karena pengolahan citra gambar dilakukan dengan cepat maka diperlukan juga proses pelacakan objek pada salah satu bagian dari gambar atau yang biasa disebut sebagai *Area of Interest* (AoI)/region. Region diperlukan untuk memilih bagian dari ruas jalan karena adanya keterbatasan penangkapan citra oleh CCTV. Nantinya, kalkulasi kepadatan lalu lintas hanya mempertimbangkan kendaraan yang ada pada region saja, tidak keseluruhan citra yang ditangkap.

Beberapa versi dari arsitektur YOLOv4 dan juga metode pelacakan seperti *Channel and Spatial Reliability Tracking* (CSRT) atau *Kernelized Correlation Filter* (KCF) digunakan dan diukur performanya dalam mengkalkulasi kepadatan lalu lintas pada ruas jalan sehingga nantinya dapat diwujudkan manajemen lalu lintas secara otomatis dan optimal [4].



## 1.2. Rumusan Masalah

Rumusan masalah dari penelitian ini adalah sebagai berikut:

1. Apakah kuantifikasi kepadatan kendaraan pada suatu ruas jalan dapat dihitung secara otomatis melalui gambar yang ditangkap kamera?
2. Apakah algoritma deteksi dan pelacakan dapat diintegrasikan pada suatu sistem sehingga dapat mengurangi beban komputasi ketika melakukan kuantifikasi kepadatan?
3. Pemilihan algoritma deteksi dan pelacakan mana yang memiliki performa terbaik?

## 1.3. Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Mengimplementasikan metode *image processing* untuk manipulasi serta ekstraksi citra.
2. Mengimplementasikan algoritma AI berbasis arsitektur YOLOv4 untuk mendeteksi objek kendaraan pada ruas jalan dengan metode *transfer learning*.
3. Mengimplementasikan algoritma AI seperti CSRT/KCF untuk pelacakan objek kendaraan pada ruas jalan.
4. Mengkalkulasikan kepadatan lalu lintas pada bagian ruas jalan dari hasil deteksi dan pelacakan objek.
5. Menguji performa dari beberapa kombinasi algoritma deteksi dan pelacakan objek yang telah disebutkan pada suatu citra dengan parameter acuan *mean average precision* (mAP) dan *frame rate/second* (FPS).
6. Menganalisis hasil pengujian masing-masing arsitektur YOLOv4 dengan visualisasi dan pendekatan statistik.
7. Menganalisis hasil pengujian keseluruhan sistem dengan visualisasi dan pendekatan statistik.

#### 1.4. Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Penelitian ini membahas terkait implementasi algoritma AI untuk kuantifikasi kepadatan lalu lintas pada ruas jalan dengan cara deteksi dan pelacakan objek kendaraan pada *Area of Interest* tertentu.
2. Algoritma AI yang digunakan untuk deteksi objek adalah arsitektur YOLOv4 dengan metode pembelajaran *transfer learning*.
3. Algoritma AI yang digunakan untuk pelacakan objek adalah CSRT/KCF.
4. Citra yang digunakan untuk uji coba berasal dari Kamera CCTV yang diambil dari tempat berbagi video yaitu youtube.com.
5. Perangkat keras yang digunakan untuk komputasi adalah GPU NVIDIA seri RTX.
6. Sistem yang digunakan harus dapat dengan cepat mendeteksi objek karena kebutuhan pengolahan citra/gambar.
7. Parameter yang digunakan untuk mengukur performa model adalah mAP untuk akurasi dan FPS untuk kecepatan.
8. Parameter yang digunakan untuk mengukur performa sistem adalah perbandingan kuantifikasi kepadatan lalu lintas antara perhitungan manual dengan hasil prediksi oleh sistem.

#### 1.5. Metodologi Penelitian

Metode yang digunakan selama penelitian ini adalah sebagai berikut:

1. Studi literatur dengan membaca dan memahami berbagai sumber kajian ilmiah yang sudah ada dan sesuai dengan tujuan laporan ini.
2. Studi kasus implementasi algoritma AI yang serupa dengan sistem deteksi dan pelacakan objek pada ruas jalan.
3. Konsultasi dan diskusi dengan dosen pembimbing terkait topik penelitian.
4. Perancangan sistem dengan mengumpulkan data video Kamera CCTV pada ruas jalan dan konfigurasi algoritma AI.

5. Implementasi sistem dengan menggunakan data citra/video yang sudah dikumpulkan.
6. Simulasi proses deteksi dan pelacakan kendaraan pada ruas jalan dengan menggunakan suatu citra dari video.
7. Analisis hasil simulasi dengan menggunakan parameter pengukur performa deteksi dan pelacakan objek yaitu mAP dan FPS serta hasil kuantifikasi kepadatan lalu lintas.



## 1.6. Sistematika Penulisan

Sistematika Penulisan pada penelitian ini dibagi ke dalam 3 bab, yaitu:

### BAB I PENDAHULUAN

Bab ini menjelaskan mengenai latar belakang penelitian, tujuan penelitian, batasan penelitian, metode penelitian, dan sistematika penulisan.

### BAB II DASAR TEORI

Bab ini menjelaskan mengenai dasar teori yang berhubungan dengan penelitian seperti *Artificial Intelligence* (AI), *framework* dan *library* yang digunakan, metrik evaluasi, dan kondisi lalu lintas di ruas jalan.

### BAB III PERANCANGAN SISTEM KUANTIFIKASI KEPADATAN LALU LINTAS DENGAN METODE DETEKSI DAN PELACAKAN OBJEK KENDARAAN DI RUAS JALAN

Bab ini membahas terkait proses perancangan sistem untuk mengkalkulasi kepadatan lalu lintas dengan menggunakan arsitektur YOLOv4 untuk deteksi objek, CSRT/KCF untuk pelacakan objek, dan OpenCV sehingga dapat mengolah citra yang ditangkap oleh kamera CCTV.

### BAB IV IMPLEMENTASI DAN ANALISIS PERFORMA SISTEM KUANTIFIKASI KEPADATAN LALU LINTAS DENGAN METODE DETEKSI DAN PELACAKAN OBJEK KENDARAAN DI RUAS JALAN

Bab ini membahas terkait implementasi dari sistem yang telah dirancang sebelumnya. Bagian ini juga ditampilkan hasil pengujian dari beberapa skenario sistem yang digunakan.

### BAB V KESIMPULAN DAN SARAN

Berisikan kesimpulan serta saran pengembangan untuk penelitian yang telah dilaksanakan.

## BAB 2

### LANDASAN TEORI

#### 2.1. *Artificial Intelligence*

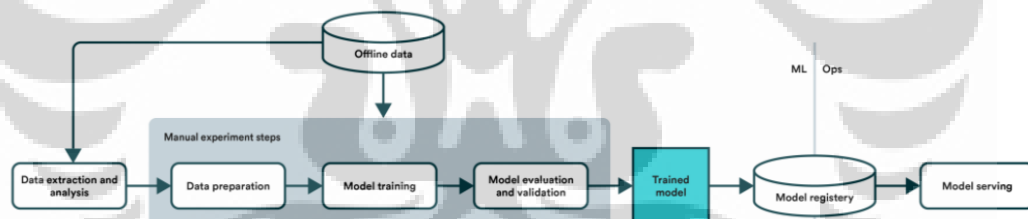
Teknologi khususnya di bidang komputasi telah mengalami perkembangan yang pesat. Hal tersebut dapat mendukung implementasi algoritma yang memerlukan komputasi berat seperti AI dapat digunakan untuk memecahkan suatu permasalahan yang kompleks. AI sendiri merupakan suatu teknologi yang dapat secara otomatis mengambil keputusan seperti manusia tanpa diprogram secara eksplisit oleh manusia itu sendiri. Untuk melakukan hal tersebut, suatu algoritma AI harus dilatih terlebih dahulu untuk dapat mengenali suatu permasalahan dengan cara mempelajari pola-pola yang ada pada suatu data. Jenis-jenis permasalahan yang dapat diselesaikan oleh AI sangat banyak, namun secara umum dapat dibagi menjadi tiga kategori yaitu regresi, klasifikasi, dan *clustering* [5]. Jenis-jenis permasalahan tersebut dapat berkembang sesuai kebutuhan dan jenis data yang diolah. Deteksi dan pelacakan objek itu sendiri tergolong sebagai perpaduan antara klasifikasi dan lokalisasi objek dimana terdapat suatu algoritma AI yang digunakan untuk menentukan dimana letak objek pada suatu citra dan tergolong pada kategori label apakah objek tersebut. [6]

##### 2.1.1. *Deep Learning Pipeline*

*Deep Learning* merupakan salah satu metode pada AI yang dapat meniru cara kerja otak manusia dalam mempelajari dan mengolah informasi. *Deep Learning* menggunakan suatu sistem yang disebut sebagai *Artificial Neural Networks* (ANN) yang terdiri dari *input layer*, *hidden layer*, dan *output layer*. Setiap *layer* memiliki sejumlah neuron yang memiliki *activation function* tertentu untuk mengatasi permasalahan non-linearitas pada data. Setiap neuron antar *layer* saling terhubung dan memiliki bobot, kedua parameter itulah yang disesuaikan saat proses pembelajaran sehingga membentuk suatu model yang dapat menyelesaikan berbagai jenis permasalahan yang telah disebutkan sebelumnya. Proses pembelajaran sendiri dibagi menjadi tiga jenis yaitu:

- *Supervised* yaitu mempelajari fungsi yang memetakan *input* ke *output* berdasarkan contoh pasangan *input-output*.
- *Unsupervised* yaitu mempelajari fungsi berdasarkan data yang diberikan tanpa mengetahui *output* yang diinginkan.
- *Reinforcement* fungsi pembelajaran yang *output* tergantung pada keadaan *input* saat ini dan *input* berikutnya tergantung pada *output* dari *input* sebelumnya.

Proses implementasi AI terdapat beberapa tahapan yang harus dipersiapkan dan dilakukan. Mulai dari pengumpulan *dataset*, *data preprocessing*, pembuatan model, *model fitting*, optimasi, hingga tahap *deployment*. Tahapan yang telah disebutkan tidak hanya dijalankan secara berurutan namun merupakan sebuah siklus yang terus berulang hingga kebutuhan atas permasalahan dapat terpenuhi. Keseluruhan proses inilah yang disebut sebagai *Deep Learning Pipeline* yang jika dilakukan secara manual maka tergambar seperti pada diagram blok berikut:



Gambar 2-1 *Deep Learning Pipeline* [7]

Dapat lihat dari gambar 2-1 bahwa data melalui proses ekstraksi dan analisis jika diperlukan sebelum memasuki tahapan eksperimen dengan model AI yang digunakan. Model yang telah dilatih dan dievaluasi dapat langsung masuk ke *model registry* yang nantinya digunakan sebagai suatu *service* sesuai dengan kebutuhan produksinya. Untuk mempersingkat waktu eksperimen hingga mendapatkan hasil model yang optimal, pelatihan model tidak selalu dimulai dari awal, terdapat metode *transfer learning* yang menggunakan konfigurasi serta

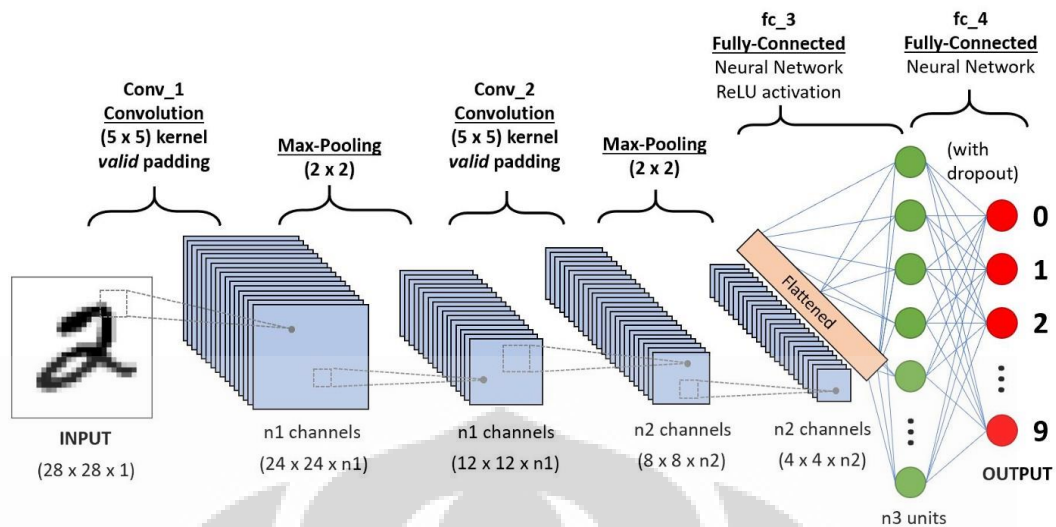
bobot dari model yang sudah ada sebelumnya dan mengaplikasikannya pada permasalahan lain yang serupa. Metode ini dapat digunakan pada jenis model apapun selama terdapat konfigurasi model serta bobot model yang telah dilatih sebelumnya.

### 2.1.2. Computer Vision

Berkembangnya algoritma AI serta kemampuan perangkat keras dalam mengolah data membuat target implementasinya menjadi lebih luas. Sekarang, AI tidak hanya dapat mengolah data tabular saja namun juga data dalam bentuk citra. Kemampuan AI untuk mengolah dan mendapatkan informasi yang penting dari sebuah citra, video, atau apapun yang berbentuk visual menjadi salah satu bidang yang dikenal dengan sebutan *computer vision*.

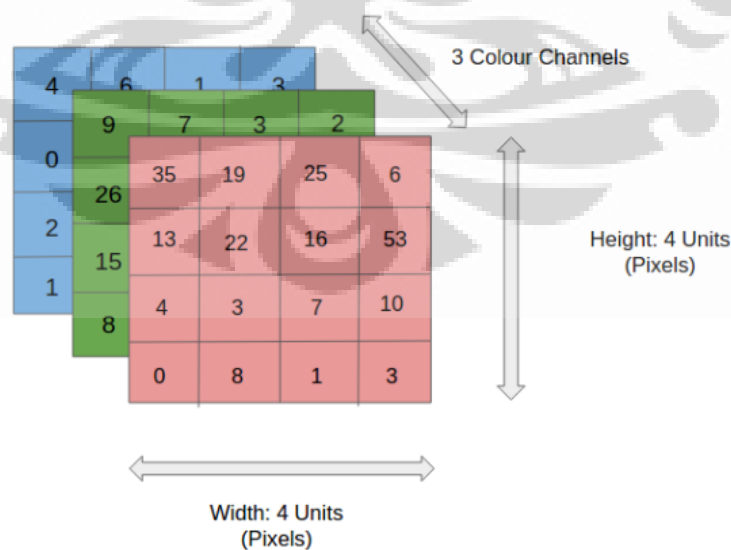
*Computer vision* memiliki prinsip kerja yang sama seperti penglihatan manusia, namun teknologi ini harus dilatih terlebih dahulu untuk mengenali komponen-komponen yang ada pada suatu citra. Meskipun teknologi ini tidak selalu sempurna dalam pengolahan citra, namun *computer vision* membutuhkan waktu lebih sedikit untuk mengolah informasi dari citra dengan menggunakan kamera dan juga algoritma AI dibandingkan dengan mata manusia. Dengan *computer vision* ribuan produk atau aset gambar dapat dianalisis dalam hitungan menit. [8]

Untuk tercapainya kebutuhan *computer vision* maka diperlukan juga algoritma AI yang sesuai. Umumnya terdapat 2 jenis algoritma yang biasa digunakan untuk mengolah data dalam bentuk citra yaitu *Deep Learning* dan *Convolutional Neural Networks* (CNN). Kedua model/algoritma tersebut dapat melihat komponen penting dari suatu gambar digital dengan cara memecah setiap *pixel* yang ada pada gambar menjadi bagian-bagian kecil yang disebut *feature*. *Feature* tersebut digunakan dalam permodelan matematika konvolusi yang kemudian memberikan prediksi terkait apa yang komputer lihat pada suatu citra. Berikut ini merupakan contoh arsitektur model CNN:



Gambar 2-2 Arsitektur Model CNN [9]

CNN dapat menangkap dependensi spasial dan temporal dari suatu citra dengan menerapkan beberapa filter yang relevan. Model CNN juga dapat menurunkan banyaknya parameter ketika mengolah data citra. Hal tersebut membuat CNN menjadi salah satu model algoritma yang memiliki kemampuan sangat baik dalam mengenali citra. Citra atau gambar digital yang biasa dilihat manusia merupakan kumpulan dari *pixel* yang memiliki warna tertentu yang dibentuk menjadi matriks 3 dimensi seperti berikut:

Gambar 2-3 Gambar RGB dengan ukuran  $4 \times 4$  pixel [9]



Masing-masing *pixel* di posisi yang sama dari setiap *channel* RGB ketika digabungkan membentuk warna tertentu. Gambar 2-3 hanyalah contoh sederhana dari suatu citra dengan *channel* RGB, namun umumnya ukuran dari citra tidaklah  $4 \times 4$  *pixel* saja. Contohnya, citra dengan resolusi 4K dapat memiliki matriks  $3840 \times 2160$  *pixel* untuk masing-masing *channel* RGB. Hal ini membuat adanya peningkatan kompleksitas dan kebutuhan untuk mengolah data dalam bidang *computer vision*.

Pada CNN sebelum penyesuaian bobot dan bias pada neuron dilakukan, ada beberapa *layer* yang harus dilalui untuk ekstraksi fitur sehingga dapat mempermudah proses pengolahan citra yang memiliki banyak informasi seperti yang telah dijelaskan sebelumnya. Urutan dari *layer* tersebut adalah *convolution layer*, *pooling layer*, dan yang terakhir adalah *fully-connected layer*. Masing-masing *layer* tersebut memiliki peran sebagai berikut:

#### 1. *Convolution layer*

Bagian ini yang berfungsi untuk mengambil fitur-fitur yang merupakan *pixel-pixel* kecil pada citra dengan menggunakan suatu matriks filter berbentuk dua dimensi yang disebut sebagai *kernel*. *Kernel* tersebut dikalikan secara *dot product* dengan bagian dari citra sehingga menghasilkan matriks baru yang disebut sebagai *feature map*. *Kernel* bergeser sekian *pixel* hingga seluruh *pixel* dari gambar dikalikan. Jauhnya pergeseran *kernel* pada citra disebut sebagai *stride*.

#### 2. *Pooling layer*

Pada bagian ini, ukuran spasial *feature map* hasil konvolusi direduksi lagi sehingga dapat mengurangi kebutuhan komputasi dan juga mengekstrak fitur yang dominan pada matriks. Proses pada *layer* ini adalah mengambil sebagian matriks dari *feature map* dan kemudian mengaplikasikan metode *pooling*. Metode yang digunakan antara lain adalah *average pooling* dan *max pooling*. Pada *average pooling* maka semua nilai pada matriks *pooling* dijumlahkan kemudian dirata-rata menjadi suatu matriks

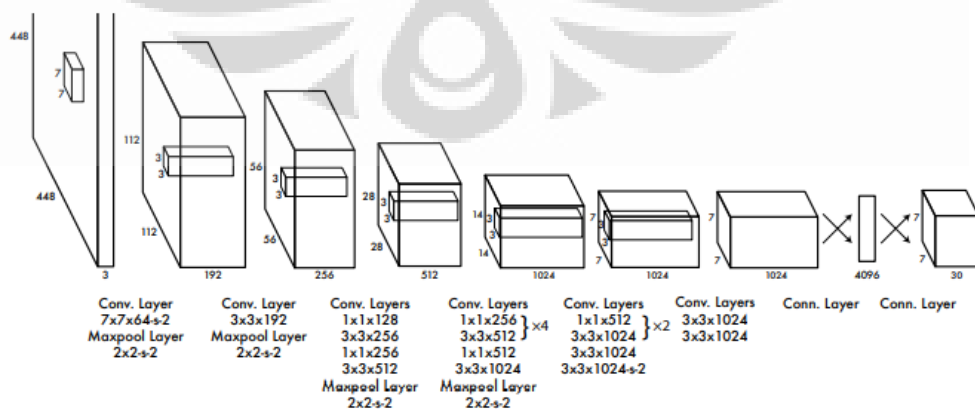
baru lagi. Sedangkan, *max pooling* hanya mengambil nilai *feature* paling besar diantara nilai yang ada pada matriks *pooling*.

### 3. *Fully-connected Layer*

*Layer* ini berisikan ANN yang digunakan untuk penyesuaian bobot dan bias sehingga dapat melakukan prediksi dengan lebih tepat. Hasil dari *pooling* yang sebelumnya dilakukan tidak langsung dimasukkan sebagai *input* dari ANN, melainkan dilakukan proses *flattening* terlebih dahulu. Proses tersebut adalah mengubah matriks berbentuk dua dimensi menjadi satu dimensi dan masing-masing nilai dari matriks tersebut dimasukan ke setiap neuron yang menjadi *input layer* dari ANN. Jumlah neuron pada *output layer* nantinya akan sesuai dengan jumlah label/kelas yang akan diklasifikasikan.

#### 2.1.3. You Only Look Once (YOLOv4)

YOLO merupakan sebuah algoritma yang dapat digunakan untuk melakukan deteksi dan pengenalan berbagai objek pada suatu citra secara *real-time* yang pertama kali dibuat oleh Joseph Redmon [10]. Proses deteksi objek pada YOLO dilakukan dengan cara regresi dengan menentukan probabilitas dari suatu kategori kelas tertentu pada citra yang dideteksi. Algoritma ini hanya memerlukan sekali *forward propagation* melalui banyak *layer* CNN untuk mendeteksi suatu objek. Berikut merupakan gambar dari arsitektur awal YOLO:



Gambar 2-4 Arsitektur YOLO [11]

Dapat dilihat bahwa algoritma deteksi YOLO memiliki 24 *convolutional layer* yang diikuti oleh 2 *fully-connected layer* sebagai *output layer*. Masing-masing *layer convolution* tersebut digunakan untuk melakukan ekstraksi fitur. YOLO juga memiliki arsitektur yang lebih sederhana yaitu dengan 9-*layer convolution* yang dikenal dengan sebutan Fast-YOLO dimana memiliki kecepatan mengelola yang jauh lebih cepat dengan YOLO biasanya, namun mengorbankan akurasi dari prediksi objek pada citra. Dengan metode ini, klasifikasi dan lokalisasi objek pada citra menjadi lebih cepat. Pada YOLO sendiri terdapat tiga proses utama yang dilakukan yaitu:

### 1. *Residual Blocks*

Pada proses ini citra dibagi menjadi beberapa *grid*. Masing-masing *grid* memiliki dimensi  $S \times S$ . Kemudian, setiap kotak yang ada pada *grid* tersebut mendeteksi apakah ada objek yang muncul didalamnya dengan mengecek titik tengah objek.

### 2. *Bounding Boxes Regression*

Setelah mendapat letak objek dari hasil deteksi masing-masing kotak pada *grid*. Algoritma YOLO melakukan komputasi terhadap letak objek dan menghasilkan prediksi dalam bentuk beberapa atribut yaitu:

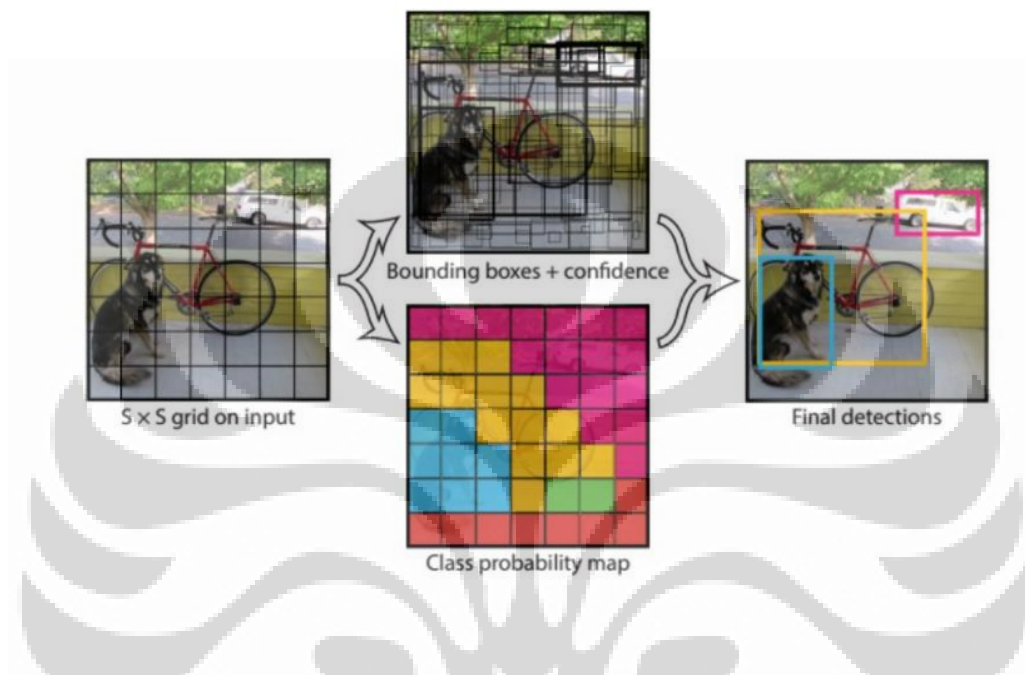
- $p_c$  yang merupakan probabilitas dari hasil prediksi.
- $b_x, b_y$  sebagai koordinat sumbu x dan y untuk titik tengah dari *bounding box*.
- $b_h, b_w$  sebagai panjang dan lebar dari *bounding box*.
- $c$  yang merupakan kelas/kategori objek (contoh: mobil, motor, truk, dll).

Masing-masing atribut tersebut nantinya digunakan untuk klasifikasi dan lokalisasi objek dengan menggambar *bounding box* pada citra.

### 3. *Intersection over Union (IoU)*

Hasil prediksi dari algoritma YOLO sering kali memiliki lebih dari satu *bounding box* yang saling tumpang tindih untuk satu objek yang sama pada citra. Untuk mengatasi permasalahan tersebut digunakan metode IoU yaitu dengan memberikan nilai 1 jika prediksi *bounding box* sama dengan *ground truth*.

Kombinasi dari setiap proses yang telah disebut kan sebelumnya lah yang membuat algoritma YOLO dapat melakukan deteksi objek dengan baik. Berikut ini adalah contoh hasil kombinasi dari proses tersebut pada suatu citra berisikan tiga objek:

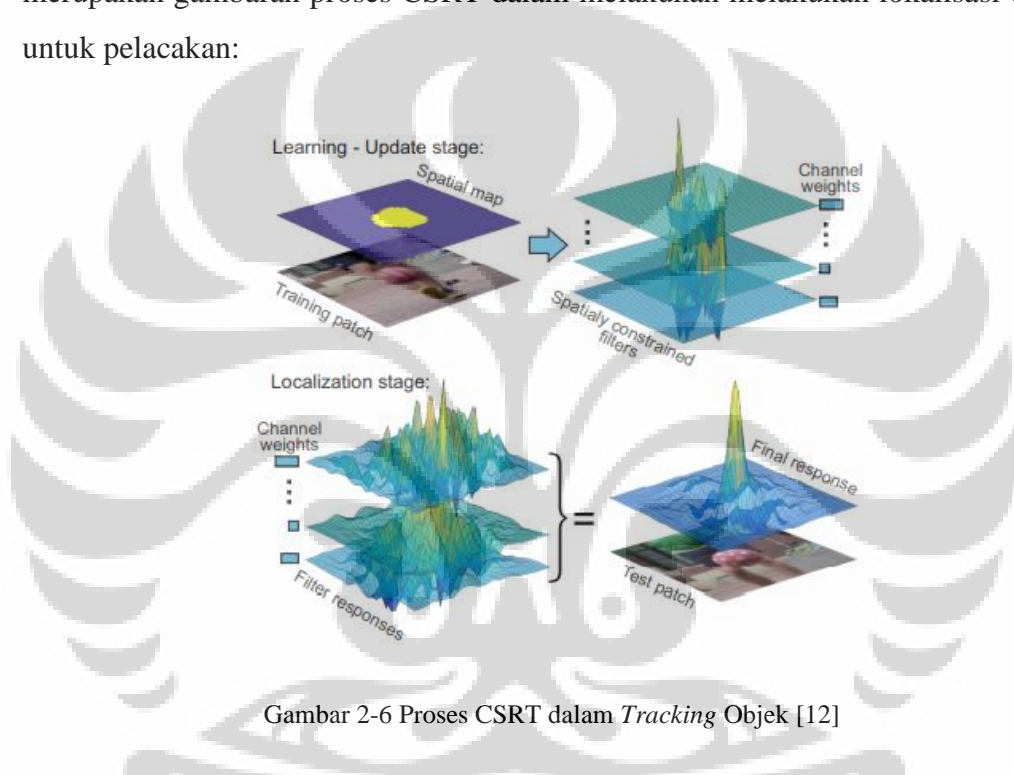


Gambar 2-5 Hasil deteksi YOLO [11]

Algoritma ini kemudian dikembangkan lagi hingga menjadi YOLO version 4 (YOLOv4) oleh AlexeyAB [3]. Arsitektur YOLOv4 terdiri dari tiga bagian yaitu *bag of freebies*, *bag of specials*, dan CSPDarknet53 yang digabungkan dengan YOLOv3. *Bag of freebies* merupakan kumpulan dari beberapa metode yang digunakan untuk meningkatkan *training cost* model, contohnya adalah dengan *data augmentation*. *Bag of specials* merupakan kumpulan dari beberapa metode yang digunakan untuk meningkatkan *inference cost* dengan jumlah yang sedikit, namun dapat meningkatkan akurasi model. Terakhir, CSPDarknet53 merupakan *backbone* untuk *feature extraction* yang dapat digunakan untuk menggabungkan input model saat ini dan sebelumnya dan diteruskan ke *dense layer*.

#### 2.1.4. Channel and Spatial Reliability Tracking (CSRT)

CSRT atau dikenal dengan nama *Channel and Spatial Reliability of Discriminative Correlation Filter Tracking* (CSR-DCF) merupakan algoritma yang dapat digunakan untuk pelacakan objek. Algoritma ini memiliki akurasi yang lebih tinggi dibandingkan beberapa algoritma pelacakan objek lainnya. Namun, kekurangannya adalah membutuhkan komputasi yang lebih besar untuk melakukan proses lokalisasi hingga pelacakan objek secara berkala. Berikut ini merupakan gambaran proses CSRT dalam melakukan lokalisasi objek untuk pelacakan:



Gambar 2-6 Proses CSRT dalam Tracking Objek [12]

CSRT dapat secara otomatis melakukan estimasi *spatial reliability map* untuk membatasi kerja dari *correlation filter*. *Correlation filter* sendiri merupakan *filter* yang digunakan untuk lokalisasi suatu target area dengan akurat berdasarkan korelasi antara komponen warna yang ada di sekitar target. Dengan metode tersebut *correlation filter* dapat fokus untuk melakukan pelacakan pada bagian tertentu saja dan kinerjanya juga lebih optimal walaupun bagian tersebut memiliki bentuk yang tidak beraturan. Seperti dapat dilihat pada Gambar 2-6 bobot *channel reliability* dihitung berdasarkan *correlation filter* yang telah dioptimasi dengan pembatasan area kerja sehingga dapat mengurangi *noise* yang muncul saat proses lokalisasi.

### 2.1.5. Kernelized Correlation Filter (KCF)

Metode yang paling umum digunakan untuk implementasi pelacakan objek yang bergerak adalah dengan menggunakan *correlation filter*, dimana sebuah objek pada gambar dua dimensi dilokalisasi dengan menerapkan filter citra sehingga dapat menghasilkan sebuah respons. Respons tersebut membentuk sebuah *gaussian shape* dengan titik puncak sebagai titik tengah dari objek yang dilokalisasi. Karena *correlation filter* diterapkan dalam *frequency domain* maka memiliki kecepatan komputasi yang dapat dikatakan *real-time*. Berikut ini adalah rumus dari *linear correlation filter*:

$$\min_w \left( \|Xw - y\|^2 + \lambda \|w\|^2 \right)$$

$$w = (X^T X + \lambda I)^{-1} X^T y$$

$$\hat{w} = \frac{\hat{x} \odot \hat{y}}{\hat{x}^* \odot \hat{x} + \lambda}$$

Formula 2-1 Rumus *Linear Correlation Filter*

Dimana  $X$  merupakan nilai *circulant matrix* citra,  $y$  respons yang diinginkan,  $\lambda$  bobot, dan  $w$  linear filter. Dengan rumus diatas, lokalisasi objek dapat dilakukan pada *frequency domain* dengan menggunakan *element-wise operation* sehingga mempercepat kompleksitas waktu algoritma menjadi  $O(n \log n)$ . Namun, *linear correlation filter* tidak selalu bisa diimplementasikan untuk pelacakan objek terutama untuk permasalahan non-linear. Sehingga metode sebelumnya dapat dilanjutkan lagi dengan mengimplementasikan trik kernel yaitu dengan melakukan *mapping* komponen-komponen dari citra ke fungsi kernel. Metode tersebut dikenal dengan *Kernelized Correlation Filter* (KCF) [13] yang menggabungkan tiga metode utama yaitu *circulant matrix*, *kernel function*, dan *ridge regression*. Berikut merupakan perhitungan KCF jika ditulis secara matematis:SS

$$\min_{\alpha} (||K\alpha - y||^2 + \lambda \alpha^T K \alpha)$$

$$\alpha = (K + \lambda I)^{-1} y$$

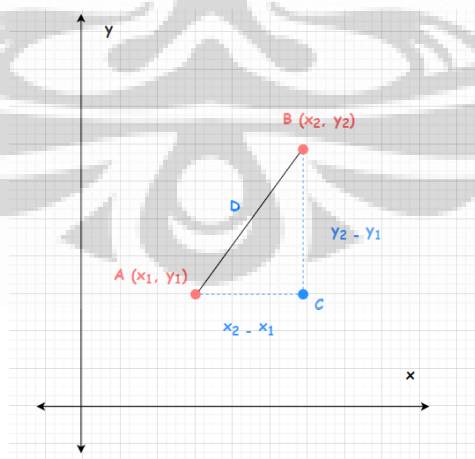
$$\hat{\alpha} = \frac{\hat{y}}{\hat{k}^{xx} + \lambda}$$

Formula 2-2 Rumus *Kernelized Correlation Filter* (KCF)

Tujuan utama dari KCF adalah membuat komponen-komponen data citra yang berupa *pixel* menjadi lebih mudah untuk di segmentasi karena fungsi kernel dapat merubah data-data pada dimensi yang lebih tinggi menjadi *lineary seperable*. Hal tersebut membuat pelacakan objek visual menjadi lebih efisien.

#### 2.1.6. Euclidean Distance

*Euclidean Distance* merupakan sebuah metode perhitungan untuk mengukur jarak antara dua titik pada suatu bidang dua dimensi. Metode ini banyak digunakan pada sistem AI seperti: *clustering* data pada algoritma K-means, klasifikasi data pada algoritma kNN, pelacakan objek dengan *image processing*, dan lain-lain.



Gambar 2-7 Jarak antara Dua Titik pada Bidang 2D

Pada gambar diatas dapat dilihat terdapat dua buah titik A dan B pada suatu diagram cartesius. Untuk mencari jarak antara kedua titik tersebut dapat digunakan perhitungan *Euclidean Distance* dengan pendekatan pythagoras seperti berikut:

$$D^2 = AC^2 + BC^2$$

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Formula 2-3 Perhitungan *Euclidean Distance*

Dengan Formula 2-3 jarak antara titik A dan B dapat dikalkulasi dengan menurunkan rumus pythagoras. Pada penelitian ini *Euclidean Distance* digunakan untuk mengintegrasikan perpindahan antara sistem deteksi dan pelacakan objek kendaraan sehingga dapat mencegah terjadinya *multiple detection* untuk satu objek yang sama.

## 2.2. Framework dan Library

Dalam proses pembuatan sistem deteksi dan pelacakan kendaraan di ruas jalan digunakan juga beberapa *library* dan *framework*. Untuk bahasa pemrograman yang digunakan adalah Python. Selanjutnya, terdapat juga *open-source library* yaitu OpenCV yang digunakan untuk memproses gambar serta implementasi DNN dengan metode *transfer learning*. Karena pemrosesan citra dari video dilakukan dengan cepat sehingga membutuhkan sumber daya komputasi yang besar maka pendeteksian objek dilakukan pada GPU melalui CUDA dan cuDNN.

### 2.2.1. Python

Python merupakan salah satu bahasa pemrograman yang bersifat *high-level* dan mudah dipahami karena lebih mudah digunakan dari pada bahasa pemrograman pada umumnya. Bahasa pemrograman ini dieksekusi dengan menggunakan *interpreter* artinya program dikompilasi per-baris secara berurutan, tidak langsung secara keseluruhan. Python juga sudah mendukung sistem *Object Oriented*



*Programming* (OOP) sehingga program dapat distrukturisasi menjadi sesuatu yang lebih sederhana dan *reuseable*. Python juga memiliki banyak *library* yang mendukung konsep AI atau lebih tepatnya *computer vision*. [14]

### 2.2.2. OpenCV

*Open-source Computer Vision Library* (OpenCV) merupakan sebuah *open-source library* skala besar yang digunakan untuk *computer vision*, *machine learning*, dan *image processing*. [15] OpenCV ditulis dengan menggunakan bahasa program C++, namun dapat juga digunakan pada bahasa Java serta Python. OpenCV dapat digunakan untuk beberapa bidang yaitu deteksi objek, pelacakan objek, dan masih banyak lagi. OpenCV mendukung implementasi DNN terutama fitur *load model* yaitu menggunakan model yang sudah ada beserta *pre-trained configuration*. Selain itu, *library* ini juga dapat diintegrasikan dengan CUDA sehingga dapat melakukan komputasi dengan menggunakan GPU NVIDIA.

### 2.2.3. CUDA dan cuDNN

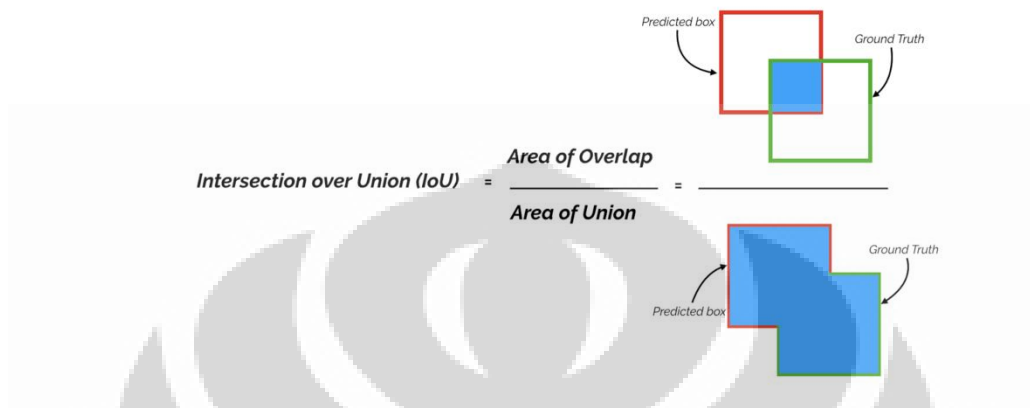
*Compute Unified Device Architecture* (CUDA) merupakan sebuah SDK yang dibuat oleh NVIDIA untuk melakukan komputasi secara *paralel*. Dengan menggunakan *platform* CUDA, eksekusi dan pemrosesan dari suatu program dapat dilakukan di GPU NVIDIA. CUDA juga memiliki *library* untuk mendukung proses komputasi seperti cuBLAS untuk aljabar linier, cuFFT untuk *fourier transform*, cuTENSOR untuk tensor aljabar linear, dan lainnya. [16] CUDA dapat diimplementasikan pada bahasa pemrograman Python, Fortran, dan C/C++.

## 2.3. Metrik Evaluasi

Dalam menilai performa dari sistem yang digunakan untuk deteksi dan pelacakan objek kendaraan di ruas jalan, diperlukan beberapa parameter yang relevan dalam mengukur kemampuan serta kecepatan model dalam klasifikasi dan lokalisasi objek pada suatu citra. Hal tersebut dapat dilakukan dengan mempertimbangkan beberapa metrik yaitu *Intersection over Union* (IoU), *Mean Average Precision*

(mAP), *Frame Rate per Second* (FPS), dan juga *Traffic Density* saat proses pengolahan citra dari video berlangsung.

### 2.3.1. *Intersection over Union (IoU)*



Gambar 2-8 *Intersection over Union (IoU)*

IoU atau biasa juga disebut sebagai *Jaccard Index* merupakan salah satu metrik yang dapat digunakan untuk evaluasi kemampuan deteksi objek. IoU melakukan komputasi terhadap luas area antara *bounding box* hasil deteksi dengan *ground truth* yang saling tumpang tindih dan membandingkannya dengan luas keseluruhan dari keduanya. Validasi IoU ditentukan dengan *threshold* tertentu. Contohnya, jika *threshold* IoU adalah 0.5 maka jika hasil perbandingan sebelumnya lebih atau sama dengan 0.5 maka objek dengan label yang diuji dapat dikatakan sesuai atau *true positive* (TP), sebaliknya jika dibawah 0.5 maka objek dan label tidak sesuai atau *false positive* (FP).

### 2.3.2. *Mean Average Precision (mAP)*

Pada *computer vision*, mAP merupakan parameter yang cukup populer dalam mengevaluasi kemampuan klasifikasi dan lokalisasi suatu objek berdasarkan model AI. Nilai mAP dan AP memiliki makna yang kurang lebih sama, istilah AP biasa digunakan untuk menghitung akurasi rata-rata dari satu kategori objek, sedang mAP adalah akurasi rata-rata dari seluruh AP masing-masing kategori. Metrik mAP memberikan gambaran terkait berapa banyak hasil prediksi dari model yang sesuai/benar. Untuk menghitung nilai dari mAP dibutuhkan dua

komponen yaitu *precision* dan *recall*. *Precision* mengukur seberapa akurat model mendeteksi objek atau berapa persen hasil prediksi yang benar. [17] Sedangkan, *recall* mengukur kemampuan model untuk mengenali objek dari kategori tertentu terhadap keseluruhan objek yang ada pada kategori tersebut. [17] Rumus yang digunakan untuk menghitung *precision* dan *recall* adalah sebagai berikut:

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

Formula 2-4 Rumus *Precision*

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

Formula 2-5 Rumus *Recall*

Untuk perhitungan masing-masing metrik diperlukan nilai *true positive* (TP), *false positive* (FP), dan *false negative* (FN). Selanjutnya untuk mengukur nilai dari mAP dapat digunakan salah satu metode yang dikenal dengan interpolasi 11 titik *recall* [18] dengan titik awal 0 yang bertamah 0.1 hingga titik akhir 1.0 seperti berikut {0, 0.1, 0.2, ..., 0.9, 1.0}. Nilai *precision* yang diambil nantinya adalah nilai yang paling tinggi pada masing-masing titik *recall*. Secara matematis rumus untuk mencari *precision* dari interpolasi 11 titik *recall* adalah sebagai berikut:

$$P_{interp}(R) = \max_i \{P_i : R_i \geq R\}$$

Formula 2-6 Rumus Interpolasi 11 Titik *Recall*

Selanjutnya nilai dari AP untuk masing-masing kategori objek dapat dikalkulasi dengan rata-rata interpolasi *precision* sebelumnya. Secara matematis dapat ditulis sebagai berikut:

$$AP = \frac{1}{11} \times \sum_{r \in \{0,0.1,0.2,...,0.9,1.0\}} P_{interp}$$

Formula 2-7 Rumus *Average Precision* [19]

Untuk nilai mAP nantinya adalah nilai rata-rata dari masing-masing AP yang telah dihitung menggunakan rumus pada Formula 2-7.

### 2.3.3. *Frame Rate per Second (FPS)*

FPS dapat menjadi metrik yang menentukan seberapa cepat sistem ketika memproses citra dan melakukan deteksi serta pelacakan objek pada citra tersebut. FPS digambarkan sebagai berapa banyak *frame* yang terproses dalam satu detik.

$$FPS = \frac{\text{Total frame diproses}}{\text{Total waktu dalam satuan detik}}$$

Formula 2-8 Perhitungan *Frame Rate per Second*

Untuk model YOLOv4 yang dilatih dengan menggunakan COCO *dataset* dan dijalankan menggunakan sistem komputasi GPU NVIDIA RTX 2070 dapat mengolah citra dengan 34 FPS. Dan untuk versi model yang lebih ringan yaitu YOLOv4-tiny dapat mengolah citra hingga 330 FPS, namun akurasi tidak sebaik yang sebelumnya. [3]

### 2.3.4. *Traffic Density*

*Traffic density* merupakan sebuah metrik yang digunakan untuk membandingkan hasil kuantifikasi kepadatan lalu lintas yang dilakukan oleh sistem secara otomatis dengan kuantifikasi kepadatan lalu lintas secara manual pada ruas jalan atau region tertentu.

$$R_i = \frac{\text{Total kendaraan terbobot pada } R_i}{\text{Total kendaraan terbobot setiap region}} \times 100\%$$

Formula 2-9 Perhitungan *Traffic Density*

Bentuk dari metrik *traffic density* sendiri merupakan persentase dari total kendaraan yang telah dibobotkan pada masing-masing region seperti tertulis pada Formula 2-9. Parameter ini dapat digunakan sebagai acuan seberapa akurat sistem dalam melakukan kuantifikasi kepadatan lalu lintas.

## 2.4. Lalu Lintas Kendaraan di Ruas Jalan

Dalam kuantifikasi kepadatan lalu lintas, model AI yang digunakan harus memiliki target objek yang dideteksi dan dilacak. Dalam melakukan hal tersebut diperlukan pemilihan jenis objek yang sesuai yaitu jenis-jenis kendaraan yang menjadi label atau target pendeteksian serta penentuan ruas jalan mana yang dikalkulasikan kepadatan kendaraanya.

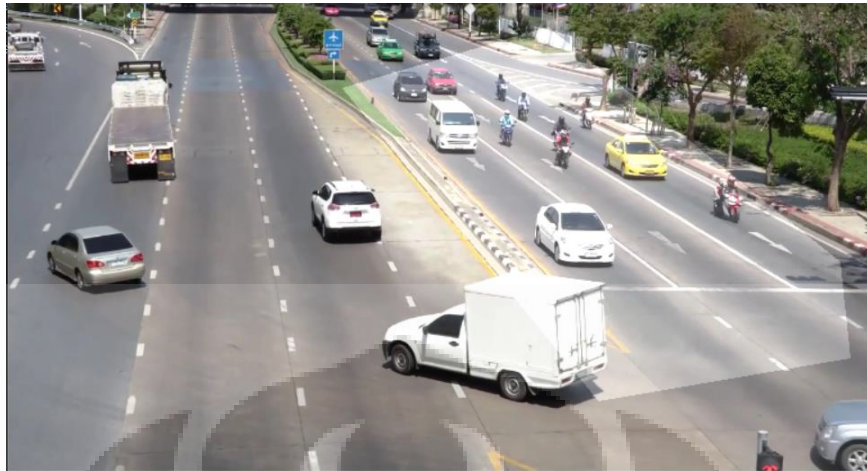
### 2.4.1. Jenis-jenis Kendaraan



Gambar 2-9 Kendaraan di Ruas Jalan

Pada penelitian ini, model AI digunakan untuk mendeteksi kendaraan yang berlalu-lalang pada ruas jalan. Karena model deteksi objek YOLOv4 dilatih dengan menggunakan *dataset* COCO maka terdapat batasan terkait jenis-jenis kendaraan yang dapat dideteksi yaitu: sepeda, motor, mobil, truk, dan bus.

#### 2.4.2. *Area of Interests (AoIs) / Region*



Gambar 2-10 Region pada Ruas Jalan

AoI/region merupakan subset atau bagian tertentu pada citra yang menjadi fokus perhatian utama dari model AI untuk diolah. Pada data citra yang bersifat dua dimensi, region didefinisikan sebagai suatu batasan-batasan pada gambar yang membentuk suatu bidang dua dimensi (persegi, poligon, dll). Contoh dari region dapat dilihat pada Gambar 2-10 yang memiliki bidang poligon transparan dengan warna putih. Pada penelitian ini pemahaman region digunakan untuk menentukan area ruas jalan mana yang dihitung kepadatan kendaraannya dengan mengimplementasikan sistem pendeteksian serta pelacakan objek.

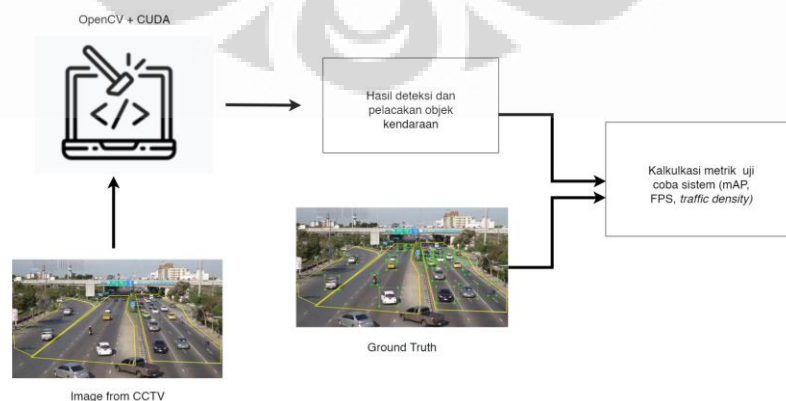
### BAB 3

## PERANCANGAN SISTEM KUANTIFIKASI KEPADATAN LALU LINTAS DENGAN METODE DETEKSI DAN PELACAKAN OBJEK KENDARAAN DI RUAS JALAN

### 3.1. Rancangan Kebutuhan Sistem

Pada bagian ini dibahas mengenai kebutuhan sistem secara keseluruhan dalam mengimplementasikan metode deteksi dan pelacakan kendaraan di ruas jalan. Rancangan kebutuhan dari sistem ini dibuat berdasarkan kajian literatur serta observasi dan uji coba implementasi algoritma AI yang digunakan. Untuk memenuhi rancangan sistem maka diperlukan pemenuhan dua jenis kebutuhan yaitu kebutuhan secara perangkat keras dan perangkat lunak.

Pada kebutuhan perangkat keras dibahas seluruh perangkat yang digunakan dari pengambilan data citra hingga citra tersebut diproses. Sedangkan, kebutuhan perangkat lunak membahas terkait kebutuhan dari berbagai jenis *software* ataupun *library* yang digunakan sebagai perantara antara implementasi program dengan perangkat keras yang digunakan. Sistem yang dibuat sendiri berbentuk suatu program berisikan model AI beserta konfigurasi. Sistem tersebut dapat menerima data video dari Kamera CCTV, kemudian mengolah setiap *frame* dari video tersebut dengan metode deteksi serta pelacakan objek untuk melakukan kalkulasi kepadatan lalu lintas seperti tergambar pada diagram blok berikut:



Gambar 3-1 Blok Diagram Sistem

### 3.1.1. Kebutuhan untuk Perangkat Keras

Pada sistem ini, kebutuhan perangkat keras yang pertama adalah Kamera CCTV untuk mengakses gambar dari suatu ruas jalan. Agar sistem dapat melakukan proses deteksi dan pelacakan dengan maksimal maka citra yang ditangkap oleh kamera setidaknya memiliki resolusi 416x416 dengan kecepatan 48 hingga 60 fps. Namun, karena akses pada Kamera CCTV jalan terbatas dan memerlukan izin terlebih dahulu ke pihak yang berwenang maka sebagai alternatif digunakan mengambil data citra dari platform berbagi video seperti YouTube.

Selain itu, kebutuhan lain yang diperlukan adalah server/komputer yang sudah dilengkapi dengan GPU untuk mengakselerasi proses deteksi dan pelacakan kendaraan pada ruas jalan. GPU yang digunakan pada penelitian ini adalah NVIDIA RTX 3050 Ti. GPU tersebut diintegrasikan dengan CUDA dan cuDNN sehingga dapat meningkatkan performa dari model YOLOv4 dalam mendeteksi kendaraan.

### 3.1.2. Kebutuhan untuk Perangkat Lunak

Dalam melakukan perancangan model AI untuk deteksi dan pelacakan objek, diperlukan instalasi komponen perangkat lunak seperti OpenCV, CUDA & cuDNN, dan konfigurasi serta bobot untuk arsitektur YOLOv4. Selain itu, untuk uji coba sistem diperlukan juga data video hasil rekaman dari suatu Kamera CCTV pada ruas jalan raya. Terakhir adalah *software* LabelImg yang digunakan untuk anotasi data citra sehingga dapat dilakukan pengujian sistem berdasarkan keluaran dari *software* tersebut. Penjelasan terkait fungsi masing-masing perangkat pada sistem adalah sebagai berikut:

#### 1. OpenCV

OpenCV digunakan untuk karena memiliki fungsi dan kemampuan untuk *image processing* yang lebih baik dibandingkan dengan *library* lainnya. Disisi lain, OpenCV juga sudah memiliki fitur untuk implementasi *Deep Neural Network* DNN tanpa melalui proses training, yaitu dengan menggunakan metode *transfer learning*.



## 2. CUDA & cuDNN

Karena OpenCV GPU ditulis dengan menggunakan CUDA maka untuk memproses gambar pada GPU diperlukan juga ekosistem CUDA. Selain itu, instalasi cuDNN juga diperlukan untuk akselerasi pemrosesan model deteksi objek dengan YOLOv4 yang merupakan model (DNN).

## 3. Konfigurasi dan Bobot untuk YOLOv4



```

yolov3-416.cfg
You, 2 weeks ago | 1 author (You)
1 [net]
2 # Testing
3 # batch=1
4 # subdivisions=1
5 # Training
6 batch=64
7 subdivisions=16
8 width=608
9 height=608
10 channels=3
11 momentum=0.9
12 decay=0.0005
13 angle=0
14 saturation = 1.5
15 exposure = 1.5
16 hue=.1
17
18 learning_rate=0.001
19 burn_in=1000
20 max_batches = 500200
21 policy=steps
22 steps=400000,450000
23 scales=1,1,1
24
25 [convolutional]
26 batch_normalize=1
27 filters=32
28 size=3
29 stride=1
30 pad=1
31 activation=leaky
32
33 # Downsample
34
35 [convolutional] You, 2 weeks ago • setup all files and codes
36 batch_normalize=1
37 filters=64
  
```

Gambar 3-2 File Konfigurasi Model YOLOv4

Implementasi dari model YOLOv4 pada sistem dengan metode *transfer learning* membutuhkan dua jenis *file*. Pertama adalah *config file* (.cfg) yang berisikan seluruh konfigurasi terkait *layer* DNN dari YOLOv4. Selanjutnya adalah *weights file* (.weights) yang berisikan bobot dari model YOLOv4 yang telah dilatih dengan menggunakan COCO Dataset yaitu data dengan 80 jenis objek yang paling umum, diantaranya adalah objek-objek kendaraan.

#### 4. Data Video



Gambar 3-3 Cuplikan Citra dari Kamera CCTV pada Ruas Jalan

Video yang digunakan merupakan video pada ruas jalan tertentu, dimana terdapat beberapa jenis kendaraan berlalu-lalang. Video tersebut memiliki rasio 16:9 atau lebih tepatnya berukuran 1280x760. Sebelum diproses, nantinya citra dari video tersebut diperkecil terlebih dahulu menjadi 416x416 sehingga dapat diolah dengan menggunakan model YOLOv4 pada OpenCV.

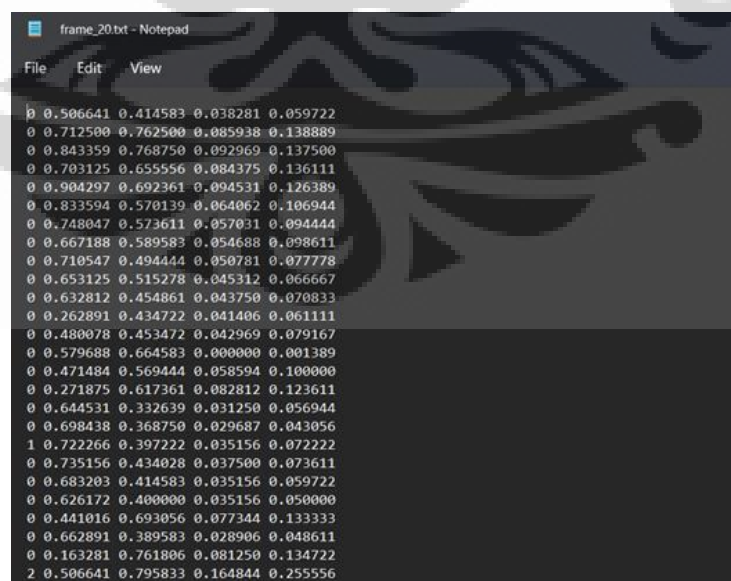
#### 5. LabelImg

LabelImg merupakan perangkat lunak yang bersifat *open source* dan berfungsi untuk melakukan anotasi pada citra. Anotasi sendiri merupakan proses memberikan label pada setiap objek yang ingin dideteksi pada suatu citra sehingga model AI yang digunakan dapat dilatih dengan mempelajari hasil pola pada anotasi. Perangkat lunak ini sangat memudahkan proses anotasi citra untuk keperluan deteksi objek karena dilengkapi dengan *graphical interface* yang mudah digunakan pengguna.



Gambar 3-4 Tampilan Antarmuka Perangkat Lunak LabelImg

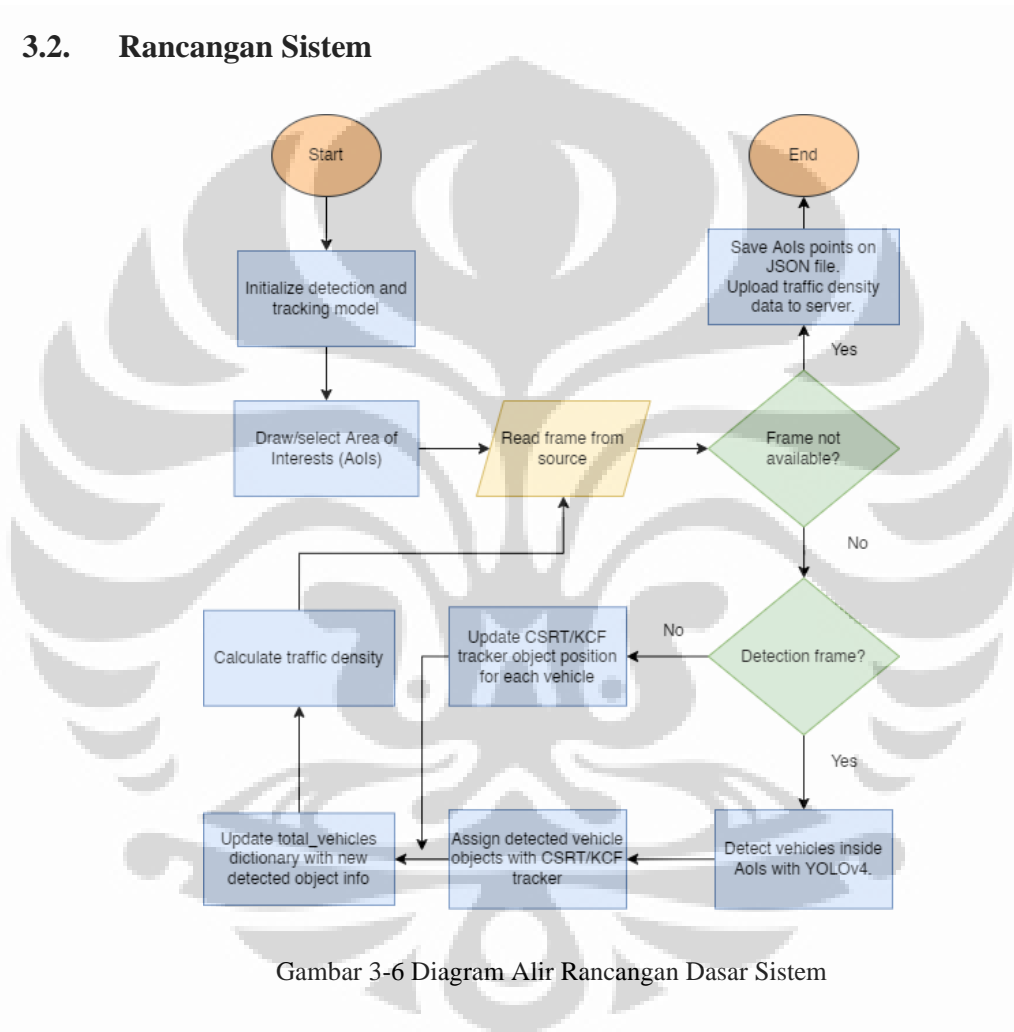
LabelImg juga memiliki beberapa format anotasi seperti VOC XML dan YOLO. Hal tersebut sangat membantu pengguna ketika melatih model AI yang mempunyai format tersendiri sebagai keluarannya seperti YOLOv4. Hasil anotasi data citra yang digunakan dengan LabelImg adalah sebuah *file text* berisikan kategori/*class* objek yang diprediksi beserta sejumlah *file text* berisikan informasi anotasi citra. Jika menggunakan format YOLO maka hasil anotasinya adalah seperti berikut.



Gambar 3-5 Hasil Anotasi dengan Menggunakan Perangkat Lunak LabelImg

Dimana setiap baris pada *file text* tersebut merupakan objek yang dideteksi dengan keterangan informasi secara berurutan adalah sebagai berikut: kelas dari objek, titik tengah sumbu x, titik tengah sumbu y, lebar *bounding box*, tinggi *bounding box*. Nilai selain kelas dari objek merupakan nilai yang telah dinormalisasi.

### 3.2. Rancangan Sistem

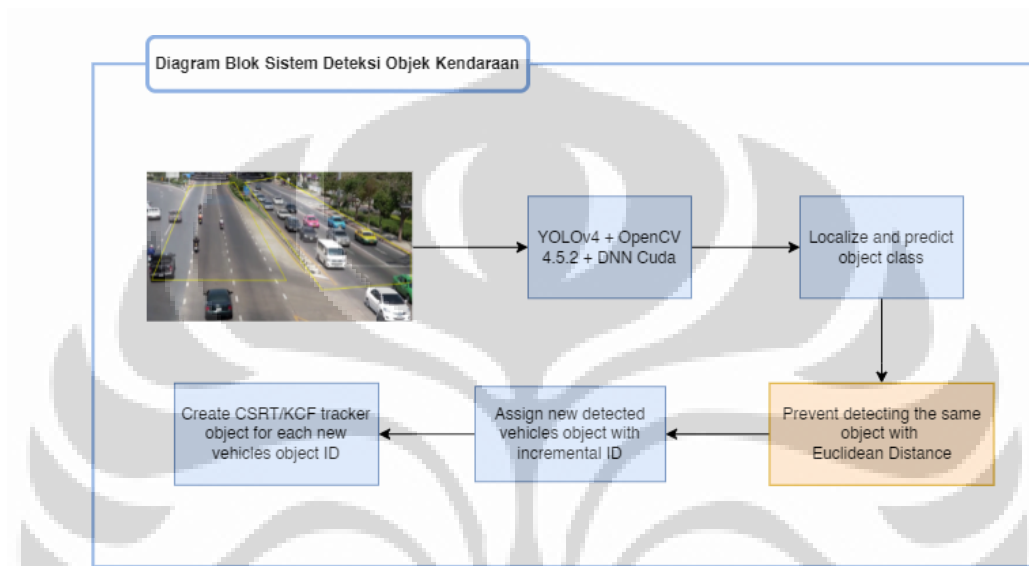


Gambar 3-6 Diagram Alir Rancangan Dasar Sistem

Perancangan dari sistem secara keseluruhan dibagi menjadi empat bagian utama. Pertama adalah memuat seluruh konfigurasi dan model untuk deteksi dan pelacakan objek kendaraan serta menentukan region pada citra yang diolah. Penentuan ini hanya dilakukan sekali saja sebelum sistem deteksi dan pelacakan berlangsung. Selanjutnya adalah perancangan untuk sistem pendeteksian objek dengan menggunakan model YOLOv4 yang hanya dilakukan setiap 80 *frame* sekali saja untuk memberikan *unique identifier* kepada objek kendaraan. Hasil

pendeteksian objek dilanjutkan ke bagian tiga yaitu pelacakan dengan menggunakan algoritma CSRT/KCF sehingga dapat mengurangi beban komputasi. Terakhir adalah proses perhitungan kepadatan lalu lintas pada region yang telah ditentukan sebelumnya dengan menghitung jumlah kendaraan yang telah diberi faktor kali sesuai dengan jenis kendaraannya.

### 3.2.1. Sistem Deteksi Objek Kendaraan



Gambar 3-7 Blok Diagram Sistem Deteksi Objek Kendaraan

Pada sistem deteksi objek kendaraan, terdapat beberapa hal yang harus disiapkan sebelum model deteksi objek dapat digunakan, yaitu: menentukan region dari citra yang ditangkap kamera CCTV, memuat bobot serta konfigurasi untuk model YOLOv4 yang digunakan sebagai detektor, dan *dictionary* untuk menyimpan hasil kuantifikasi *traffic density*. Ketika hal tersebut telah dipenuhi maka model untuk deteksi objek kendaraan sudah dapat digunakan.

Penggunaan model YOLOv4 dalam mendeteksi objek secara bergantian digunakan dengan proses pelacakan objek. Pada penelitian ini sistem deteksi objek hanya dilakukan setiap 80 *frame* sekali untuk mengurangi beban komputasi. Hasil dari pendeteksian objek sendiri sesuai dengan keluaran dari model YOLOv4 yaitu beberapa atribut yang menentukan klasifikasi dan lokalisasi objek berupa probabilitas dari klasifikasi, koordinat titik tengah objek, panjang dan lebar dari

objek, dan kategori/kelas objek ( $p_c, b_x, b_y, b_h, b_w, c$ ) yang dapat digunakan untuk menentukan *bounding box* dari objek. Untuk mengvisualisasikan *bounding box* dari objek dengan OpenCV diperlukan kalkulasi titik *anchor* ( $a$ ) dan titik pojok kanan bawah ( $c$ ) dari *bounding box*. Untuk mendapatkan kedua titik tersebut, komponen  $b_x, b_y, b_h, b_w$  harus disesuaikan dengan lokasinya pada citra. Hal tersebut dapat dilakukan dengan mengalikan komponen tersebut dengan panjang serta lebar dari citra yang diolah kemudian. Secara matematis dapat ditulis seperti berikut:

$$\begin{aligned}x &= b_x \times \text{lebar citra} \\y &= b_y \times \text{panjang citra} \\w &= b_w \times \text{lebar citra} \\h &= b_h \times \text{panjang citra}\end{aligned}$$

Formula 3-1 Perhitungan Koordinat Titik Tengah, Panjang, dan Lebar Objek pada Citra

Setelah itu, kalkulasi kalkulasi titik *anchor* dan titik pojok kanan bawah dari *bounding box* dapat dilakukan dengan menggunakan formula berikut:

$$\begin{aligned}a_x &= x - \frac{w}{2} & a_y &= y - \frac{h}{2} \\c_x &= x + \frac{w}{2} & c_y &= y + \frac{h}{2}\end{aligned}$$

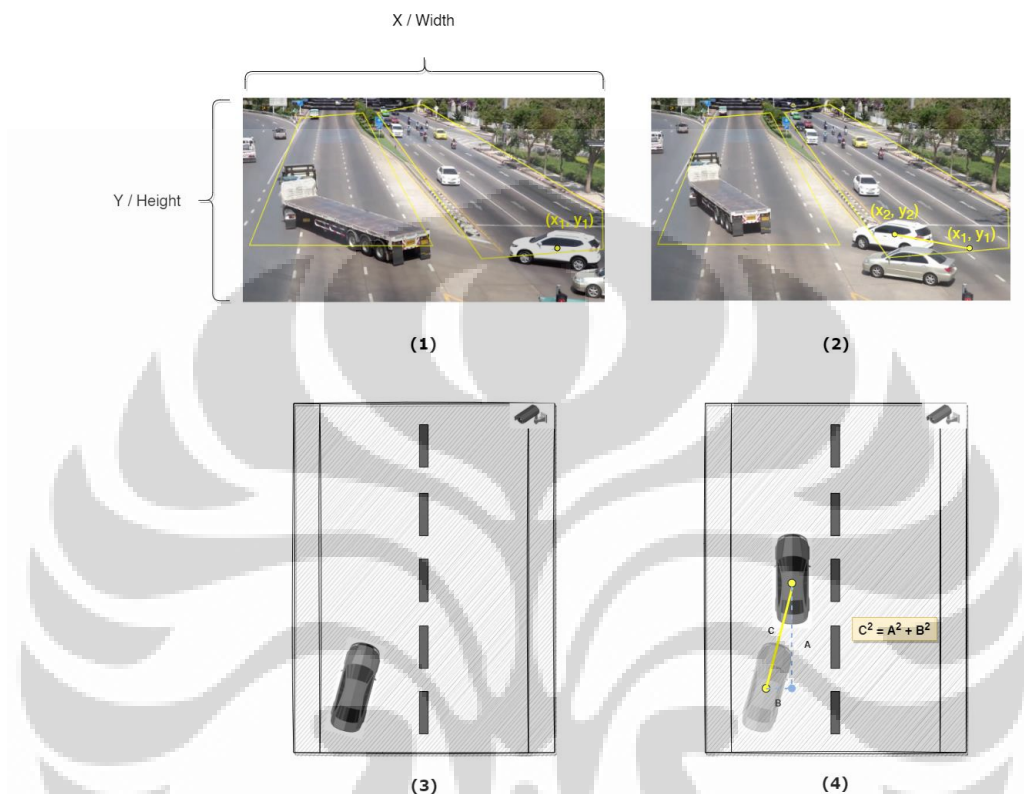
Formula 3-2 Perhitungan Titik *Anchor* dan Titik Pojok Kanan Bawah *Bounding Box*

Setelah *bounding box* terbentuk maka dapat dilakukan pengecekan apakah kendaraan berada di dalam *region* yang telah ditentukan atau tidak. Jika kendaraan berada di dalam maka *bounding box* divisualisasikan serta diikutkan kedalam perhitungan kepadatan kendaraan. Proses ini terus dilakukan selama sistem dapat mengakses data citra yang ditangkap oleh Kamera CCTV.

Salah satu hal yang perlu dipertimbangkan dari sistem deteksi objek kendaraan ini adalah kemungkinan model AI yang digunakan mendeteksi objek yang sama sebagai objek yang baru ketika terjadi transisi dari sistem pelacakan ke sistem deteksi objek. Upaya dalam mengurangi serta mencegah kelemahan sistem



yang telah disebutkan sebelumnya adalah dengan mengenali apakah suatu objek kendaraan merupakan objek baru atau bukan melalui perpindahan *bounding box* objek tersebut. Hal tersebut dilakukan dengan mengimplementasikan perhitungan *Euclidean Distance*.

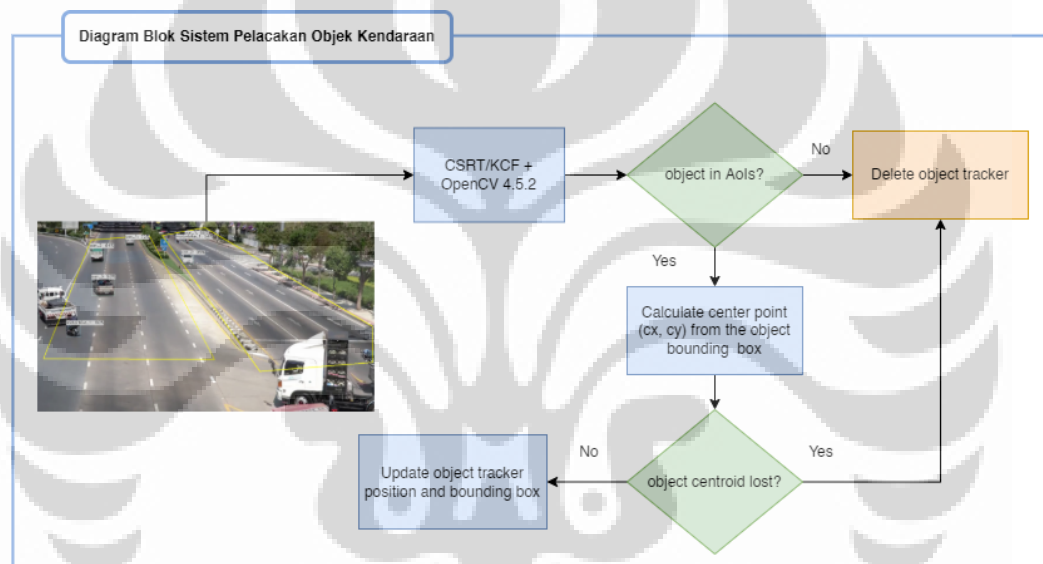


Gambar 3-8 Pencegahan Kelemahan Sistem Deteksi Objek dengan *Euclidean Distance*

Gambar 3-8 menjelaskan bagaimana *Euclidean Distance* diimplementasikan dalam sistem yaitu: (1) Sebuah citra yang diproses umumnya berbentuk bidang dua dimensi dengan panjang dan lebar tertentu yang secara tidak langsung merupakan bentuk dari diagram cartesius sehingga titik tengah dari suatu kendaraan dapat berada pada nilai  $(x, y)$  tertentu; (2) Ketika kendaraan berpindah dan masih berada dalam citra maka terdapat dua titik yaitu  $(x_1, y_1)$  yang merupakan posisi sebelumnya dan  $(x_2, y_2)$  sebagai posisi saat ini; (3) Gambaran situasi ruas jalan ketika dilihat dari sudut pandang bagian atas; (4) Contoh perpindahan kendaraan yang digambarkan dengan meninjau titik tengah sebagai acuan serta bagaimana *Euclidean Distance* dapat dikalkulasi dengan *pythagoras*.

Nilai perpindahan yang telah didapatkan dari proses perhitungan *Euclidean Distance* tersebut akan dibandingkan dengan nilai batasan yang didefinisikan sesuai dengan resolusi citra yang akan diolah. Pada penelitian ini batasan yang digunakan adalah sebesar 30 *pixel*. Sehingga, jika nilai perpindahan kurang dari nilai batasan maka dapat dikatakan bahwa objek kendaraan merupakan objek dengan id yang sama. Sedangkan, jika nilai perpindahannya berbeda dengan nilai sebelumnya maka dapat dikatakan bahwa objek tersebut merupakan objek yang baru dideteksi.

### 3.2.2. Sistem Pelacakan Objek Kendaraan



Gambar 3-9 Blok Diagram Sistem Pelacakan Objek Kendaraan

Proses pelacakan dilakukan dengan menggunakan CSRT/KCF *tracker* setelah mendapatkan data *bounding box* dari hasil deteksi objek. Data tersebut berisikan koordinat dari titik *anchor* yang berada di pojok kanan masing-masing *bounding box* serta panjang dan lebar dari *bounding box*. Informasi tersebut digunakan untuk menginisialisasi model pelacak CSRT/KCF sehingga dapat melakukan pelacakan terhadap objek kendaraan terkait. Inisialisasi hanya dilakukan jika objek kendaraan berada di region yang telah ditentukan dan juga titik *centroid* dari *bounding box* objek kendaraan terbaca. Jika kedua hal tersebut tidak terpenuhi maka objek pelacak tidak akan diinisialisasi atau dihapus.



Proses pelacakan berjalan setiap waktu kecuali saat sistem melakukan deteksi objek, karena pelacakan memiliki biaya komputasi yang lebih rendah dari pada menjalankan deteksi objek. Saat proses berlangsung, posisi dari objek pelacak selalu diperbaharui dengan fungsi CSRT/KCF *tracker* yang ada pada OpenCV. Ketika berhasil melakukan pembaharuan posisi pelacak, fungsi tersebut mengembalikan nilai titik tengah, panjang, dan lebar objek pada citra yang terbaru sehingga dapat digunakan untuk menggambar *bounding box* yang baru dengan Formula 3-2.

### 3.2.3. Metode Kalkulasi Kepadatan pada Ruas Jalan

Kalkulasi kepadatan pada ruas jalan tidak dapat hanya dilakukan dengan menghitung total jumlah objek yang terdeteksi saja. Hal tersebut dikarenakan setiap kendaraan memiliki kapasitas penumpang yang berbeda dan memiliki pengaruh terhadap luas area yang diambil oleh kendaraan tersebut. Untuk mengimbangi hal tersebut maka dibuat sistem pembobotan untuk masing-masing jenis kendaraan. Pada penelitian ini terdapat 5 jenis kendaraan yang dideteksi oleh sistem yaitu sepeda, motor, mobil, bus, dan truk. Berikut adalah pembobotan yang menjadi faktor kali dari masing-masing kendaraan yang telah disebutkan:

- Sepeda = 1
- Motor = 2
- Mobil = 5
- Bus / Truk = 10

Bobot yang digunakan sebagai faktor kali diatas ditentukan berdasarkan jumlah kemungkinan penumpang yang menaiki, mengendarai, atau berada di dalam kendaraan.

### 3.3. Rancangan Analisis dan Evaluasi Sistem

Pada bagian ini dijelaskan penggunaan dari masing-masing metrik dalam mengevaluasi performa dari sistem. Performa yang dievaluasi antara lain adalah

akurasi dan kecepatan model serta kemampuan sistem dalam mengkalkulasi kepadatan kendaraan. Akurasi dinilai menggunakan metrik mAP dan kecepatan dengan menggunakan FPS. Sedangkan, kepadatan kendaraan dengan menggunakan perhitungan yang telah dibahas pada Formula 2-9.

### 3.3.1. Evaluasi Akurasi Model dengan mAP

Pada evaluasi akurasi sistem deteksi dan pelacakan objek digunakan dua jenis arsitektur YOLO yaitu YOLOv4 dan YOLOv4-tiny. Masing-masing arsitektur diuji dengan mengukur performa dari beberapa parameter yaitu:

1. *Intersection over Union* (IoU) dari masing-masing objek.
2. *Average Precision* (AP) dari masing-masing kendaraan.
3. *Mean Average Precision* (mAP) dari keseluruhan model dalam mendeteksi kendaraan.

### 3.3.2. Evaluasi Sistem dalam Mengkalkulasi Kepadatan Kendaraan

Dalam mengevaluasi kemampuan sistem deteksi dan pelacakan dalam melakukan kuantifikasi objek kendaraan digunakan metrik *traffic density* dengan metode kalkulasi seperti pada Formula 2-9. Hasil perhitungan tersebut kemudian dibandingkan dengan nilai aktual yaitu perhitungan *traffic density* yang dilakukan secara manual. Selain itu, dalam proses evaluasi digunakan 4 kombinasi dari model deteksi dan pelacakan yang berbeda-beda yaitu: YOLOv4 dan CSRT, YOLOv4 dan KCF, YOLOv4-tiny dan CSRT, YOLOv4-tiny, dan KCF. Keempat skenario tersebut dianalisis dan dibandingkan hasilnya.

### 3.3.3. Evaluasi Kecepatan Sistem dalam Memproses Citra dengan FPS

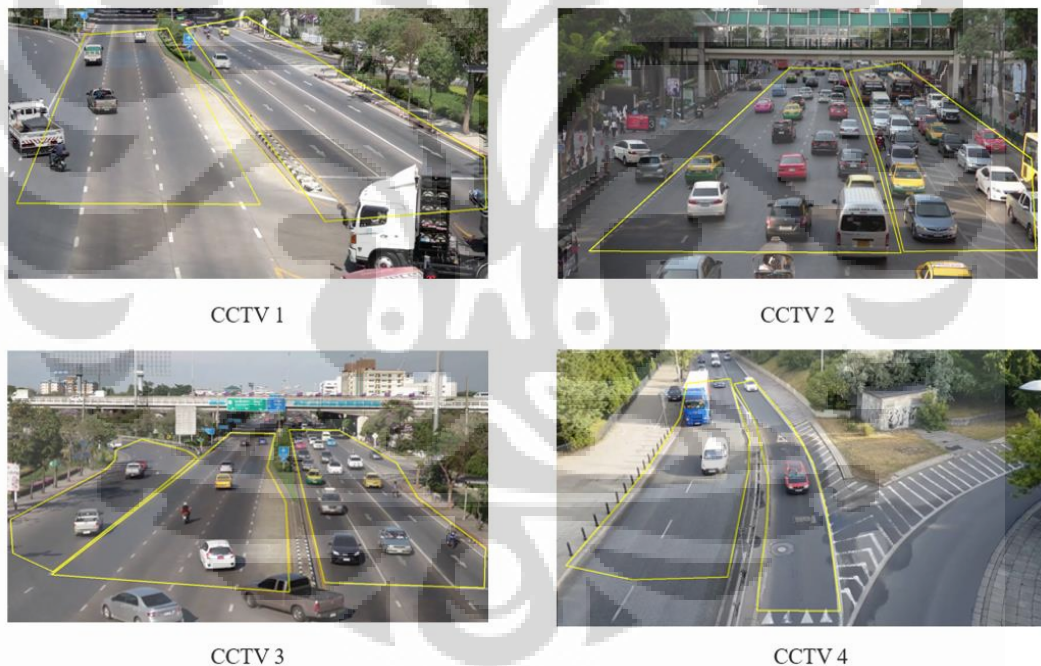
Pada evaluasi kecepatan sistem deteksi dan pelacakan objek kendaraan digunakan 4 kombinasi dari model deteksi dan pelacakan yang berbeda-beda yaitu: YOLOv4 dan CSRT, YOLOv4 dan KCF, YOLOv4-tiny dan CSRT, YOLOv4-tiny, dan KCF. Keempat skenario tersebut dianalisis dan dibandingkan hasilnya. Masing-masing skenario diuji dengan melihat rata-rata FPS yang dihasilkan ketika sistem mengolah citra dari video.

## BAB 4

### IMPLEMENTASI DAN ANALISIS PERFORMA SISTEM KUANTIFIKASI KEPADATAN LALU LINTAS DENGAN METODE DETEKSI DAN PELACAKAN OBJEK KENDARAAN DI RUAS JALAN

#### 4.1. Implementasi Sistem

Sistem pendeteksian dan pelacakan objek kendaraan pada ruas jalan ini dirancang untuk diolah melalui perangkat GPU NVIDIA yang dilengkapi dengan CUDA dan cuDNN sehingga dapat mengakselerasi pemrosesan citra. Perangkat GPU diintegrasikan dengan program python melalui *framework* OpenCV versi 4.5.2 Citra yang diolah diambil dari rekaman CCTV. Untuk proses pengujian digunakan 4 rekaman dari sumber yang CCTV yang berbeda.



Gambar 4-1 Citra dari 4 CCTV yang berbeda

Dengan menggunakan metode *transfer learning* model YOLOv4 yang berfungsi untuk mendeteksi kendaraan dapat melewati tahap pembelajaran karena sudah memiliki beberapa objek kendaraan yang perlu dideteksi (mobil, motor, sepeda, bus, dan truk). Model YOLOv4 sendiri termasuk golongan *start-of-the-art*

dalam melakukan deteksi objek. Selain itu, OpenCV juga menyediakan beberapa fungsi yang dapat digunakan untuk memuat *configuration file* dan *pre-trained weights* model dari COCO Dataset. OpenCV juga menyediakan beberapa model untuk pelacakan objek seperti CSRT, KCF, MOSSE, dll. Untuk penelitian ini model pelacakan yang digunakan adalah CSRT dan KCF.

Performa model dan sistem diuji berdasarkan keempat rekaman yang telah disediakan tersebut. Dalam kasus kuantifikasi kepadatan lalu lintas pada ruas jalan maka region telah didefinisikan terlebih dahulu pada masing-masing rekaman. Selanjutnya, sistem diuji untuk memprediksi kendaraan dan melakukan kalkulasi kepadatan kendaraan pada masing-masing region. Proses pengujian sistem dilakukan pada sistem lokal dengan spesifikasi seperti berikut.

Kategori	Keterangan
Jenis Perangkat	Laptop ASUS Dash F15
Prosesor	11th Gen Intel(R) Core (TM) i7-11370H @ 3.30GHz 3.30 GHz
Memori	16 GB
Kartu Grafis	1 x NVIDIA GeForce RTX 3050 Ti
Sistem Operasi	Windows 11
Penyimpanan	512 GB

Tabel 4-1 Spesifikasi Perangkat untuk Pengujian Sistem

## 4.2. Pengujian Akurasi Model Deteksi Objek Kendaraan

Dalam pengujian akurasi model deteksi objek kendaraan digunakan dua arsitektur YOLO yang berbeda yaitu: YOLOv4 dan YOLOv4-tiny. Perbedaan utama dari kedua model tersebut adalah jumlah layer yang digunakan sebagai *backbone* dan juga pemilihan *anchor boxes* dimana YOLOv4 memiliki layer yang lebih banyak dari pada YOLOv4-tiny. Masing-masing model diuji berdasarkan kemampuannya dalam mengenali objek kendaraan (bus, mobil, motor, sepeda, truk) serta melokalisasi objek tersebut pada citra dengan menggunakan *bounding box*.

Parameter yang digunakan dalam pengujian ini adalah *Average Precision* (AP) untuk masing-masing kendaraan, *Mean Average Precision* (mAP) yang merupakan presisi rata-rata model dalam mengenali seluruh objek, dan yang terakhir adalah *Intersection over Union* (IoU) untuk menilai kemampuan model dalam melokalisasi objek. Objek dikatakan sebagai *true positive* atau berhasil dideteksi dengan benar ketika prediksi jenis objek kendaraan/kelas sama dengan kelas yang ada pada *ground-truth* serta nilai dari IoU *bounding-boxes* antara hasil prediksi dan *ground-truth* lebih dari 50%. Nilai AP dan mAP nantinya didapatkan dari hasil perbandingan *true positive* dan *false positive*.

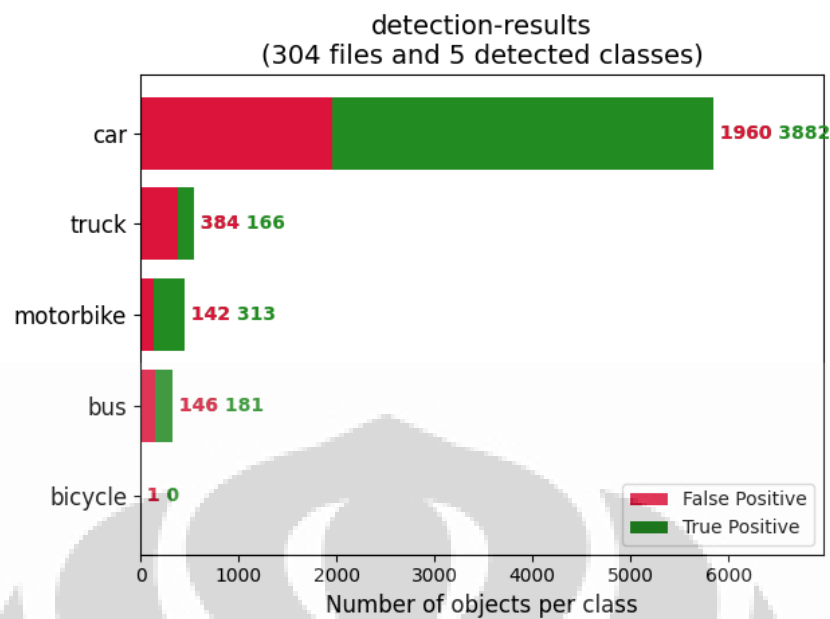
### 4.2.1. Model YOLOv4

Pengujian model deteksi YOLOv4 dilakukan pada citra dari rekaman CCTV yang telah dibahas sebelumnya. Pada masing-masing rekaman CCTV sudah terinisiasi region yang menjadi area fokus model dalam mendeteksi objek. Pada pengujian ini ditampilkan hasil dari pendeteksian objek oleh model YOLOv4 beserta AP dan mAP untuk masing-masing region yang telah didefinisikan. Berikut ini adalah hasil dan analisis dari percobaan yang dilakukan.

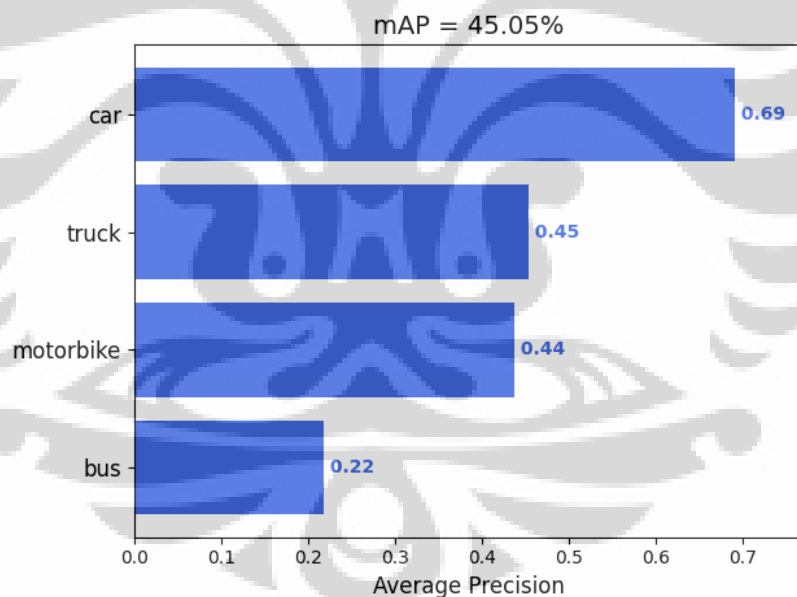
Footage	Region	AP (bus, mobil, motor, sepeda, truk) dalam %	mAP
CCTV 1	region 0	-, 88.15, 75.11, -, 3.7	55.65 %
	region 1	-, 64.87, 32.33, -, 70.26	55.82 %
CCTV 2	region 0	7.44, 94.0, 5.4, -, 17.98	31.21 %
	region 1	-, 96.91, -, -, 90.24	93.57 %
CCTV 3	region 0	3.45, 71.17, 13.33, -, 89.72	44.42 %
	region 1	-, 76.0, 37.2, -, 17.0	32.58 %
	region 2	-, 95.23, -, -, -	47.61 %
CCTV 4	region 0	15.73, 29.8, 19.8, -, -	21.78 %
	region 1	50.0, 79.43, 48.51, -, -	59.30 %

Tabel 4-2 Nilai AP dan mAP dari YOLOv4 untuk Setiap CCTV

Pada Tabel 4-2 diketahui bahwa letak kamera CCTV dan penentuan region juga menjadi faktor yang penting dalam proses kuantifikasi kepadatan kendaraan. Hal tersebut dibuktikan dengan adanya variasi dari nilai metrik mAP model YOLOv4 dengan nilai standar deviasi sebesar 31%. Area dengan nilai mAP tertinggi ada pada CCTV 2 – region 1 karena sudut pandang dari kamera lebih miring yang membuat setiap kendaraan tidak mudah tertumpuk. Sedangkan, nilai mAP terendah ada pada CCTV 4 – region 0 karena kebanyakan kendaraanya bertumpuk dan juga membelakangi kamera. Maka dari itu, dapat dikatakan penentuan posisi atau sudut pandang kamera juga dapat mempengaruhi akurasi dari model deteksi objek.



Gambar 4-2 Perbandingan Nilai FP dan TP Objek untuk Model YOLOv4



Gambar 4-3 Grafik Nilai AP dan mAP dari Hasil Deteksi YOLOv4

Pada percobaan deteksi objek yang pertama peneliti melakukan uji coba dengan menggunakan model YOLOv4. Hasil pengujian pada empat CCTV yang berbeda untuk masing-masing region tertentu dapat dilihat pada Tabel 4-2. Nilai mAP model deteksi secara keseluruhan adalah 45.05% dimana objek kendaraan

wdengan akurasi paling tinggi secara berurutan adalah mobil, truk, motor, dan sepeda. Mobil memiliki akurasi uji yang paling tinggi dikarenakan memiliki jumlah data yang paling banyak pada nilai *ground-truth*. Model YOLOv4 juga sering kali salah dalam memprediksi objek kendaraan truk dan bus karena memiliki bentuk yang sama dengan kendaraan mobil. Nilai mAP model YOLOv4 ini juga mengalami penurunan sebesar 20.65% dari nilai hasil uji coba YOLOv4 dengan *dataset* COCO [3]. Penurunan ini terjadi karena adanya peningkatan kompleksitas dari objek yang harus dideteksi pada citra. Citra yang diolah merupakan kendaraan yang ada pada area jalan yang luas sehingga untuk objek kendaraan yang jauh dari kamera akan menjadi lebih kecil dan lebih sulit untuk dideteksi.



Gambar 4-4 Hasil Deteksi YOLOv4 pada Setiap Citra CCTV

Gambar diatas merupakan perbandingan antara hasil lokalisasi serta klasifikasi objek kendaraan dari model YOLOv4 yang digambarkan dengan *bounding box* berwarna hijau dan juga label serta nilai *ground-truth* yang digambarkan dengan *bounding box* berwarna hitam.



#### 4.2.2. Model YOLOv4-tiny

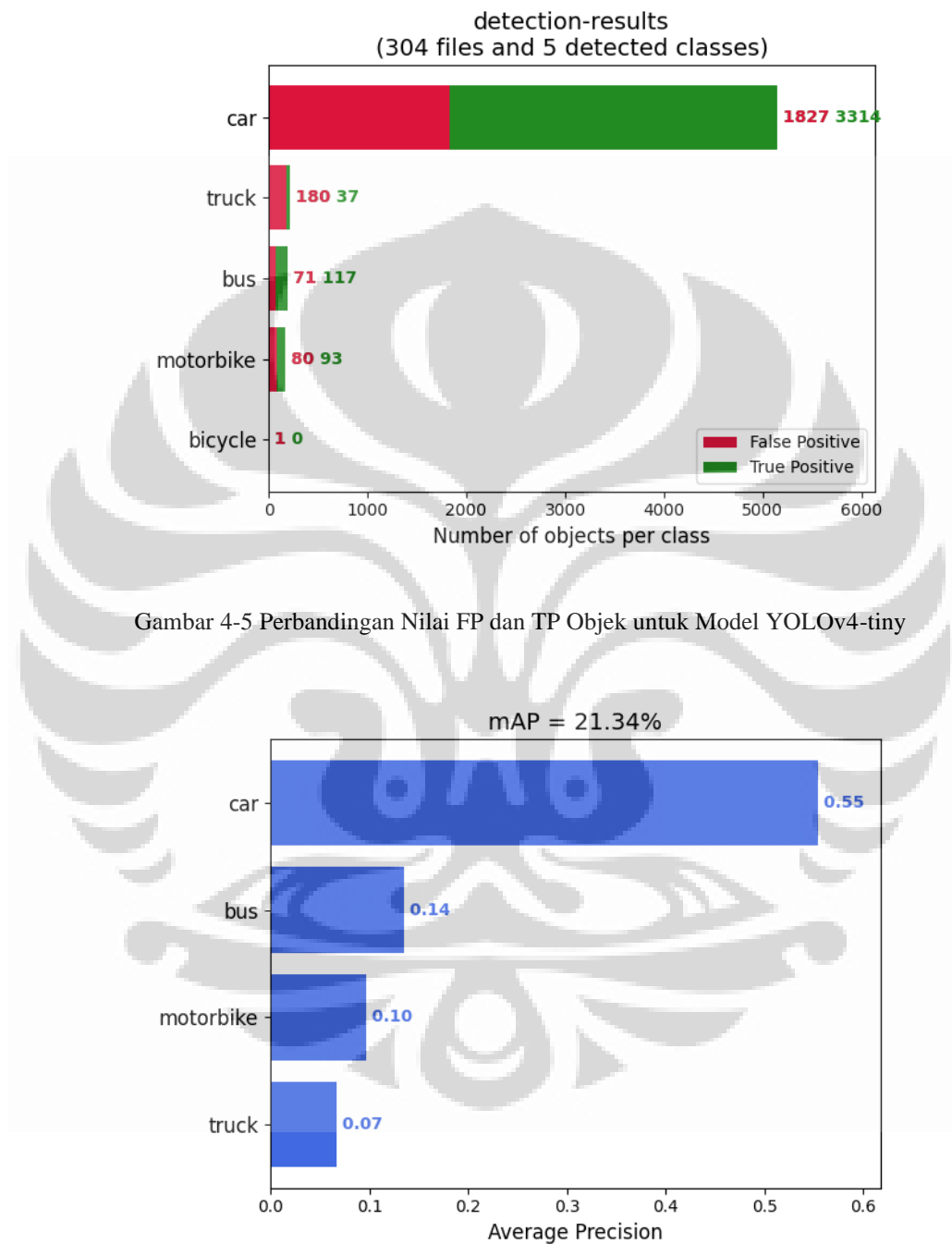
Pengujian model deteksi YOLOv4-tiny juga dilakukan pada citra dari rekaman CCTV yang telah dibahas sebelumnya. Pada masing-masing rekaman CCTV sudah terinisiasi region yang menjadi area fokus model dalam mendeteksi objek. Pada pengujian ini ditampilkan hasil dari pendeteksian objek oleh model YOLOv4-tiny beserta AP dan mAP untuk masing-masing region yang telah definisikan. Berikut ini adalah hasil dan analisis dari percobaan yang dilakukan.

Footage	Region	AP (bus, mobil, motor, sepeda, truk) dalam %	mAP
CCTV 1	region 0	-, 71.68, 23.55, -, -	31.74 %
	region 1	-, 49.82, 19.2, -, 27.99	32.34 %
CCTV 2	region 0	4.99, 71.14, -, -, -	19.03 %
	region 1	-, 74.96, -, -, 27.12	51.04 %
CCTV 3	region 0	4.31, 58.42, -, -, -	15.68 %
	region 1	1.15, 65.16, -, -, -	16.58 %
	region 2	-, 68.42, -, -, -	34.21 %
CCTV 4	region 0	1.55, 40.94, 13.58, -, -	18.69 %
	region 1	36.65, 79.72, 0.6, -, -	38.98 %

Tabel 4-3 Nilai AP dan mAP dari YOLOv4-tiny untuk Setiap CCTV

Pada tabel 4-3 dapat dilihat juga nilai mAP untuk masing-masing region pada CCTV yang berbeda. Nilai mAP juga cukup bervariasi, namun tidak seperti pada YOLOv4, nilai standar deviasinya lebih kecil dari sebelumnya yaitu sekitar 16.1%. Area dengan nilai mAP tertinggi dan terbesar juga sama seperti hasil pengujian YOLOv4 yaitu secara berurutan CCTV 2 – region 1 dan CCTV 4 region 0. Hal ini membuktikan bahwa penempatan kamera CCTV memang mempengaruhi proses model dalam melakukan deteksi objek kendaraan pada

ruas jalan. Secara keseluruhan model YOLOv4 lebih unggul dalam perihal akurasi deteksi objek kendaraan yang diukur berdasarkan metrik mAP dalam penelitian ini.



Gambar 4-6 Grafik Nilai AP dan mAP dari Hasil Deteksi YOLOv4-tiny

Hasil uji model dengan metrik mAP secara keseluruhan adalah 21.34%. Hasil ini turun sekitar 23.7% menjadi lebih rendah dari uji coba sebelumnya dengan menggunakan model YOLOv4. Hal ini dikarenakan YOLOv4-tiny memang memiliki kompleksitas model yang lebih rendah jika dibandingkan YOLOv4, namun karena itu juga model ini lebih ringan ketika diuji coba. Objek kendaraan dengan nilai AP tertinggi adalah mobil karena data yang digunakan pada pengujian ini juga masih sama dengan data sebelumnya, namun akurasi menurun sebanyak kurang lebih 14%. Model juga semakin sulit dalam melakukan deteksi objek yang kecil walaupun dekat dengan kamera seperti objek kendaraan motor atau kendaraan-kendaraan yang mirip dengan mobil seperti truk/bus.



Gambar 4-7 Hasil Deteksi YOLOv4-tiny pada Setiap Citra CCTV

Gambar diatas merupakan perbandingan antara hasil lokalisasi serta klasifikasi objek kendaraan dari model YOLOv4-tiny yang digambarkan dengan *bounding box* berwarna hijau dan juga label dengan nilai *ground-truth* yang digambarkan dengan *bounding box* berwarna hitam.

### 4.3. Pengujian Performa Sistem

Dalam pengujian performa sistem deteksi dan pelacakan kendaraan secara keseluruhan dilakukan dengan empat skenario berbeda untuk data yang sama. Skenario yang diuji sendiri terdiri dari kombinasi antara YOLOv4 dan CSRT, YOLOv4 dan KCF, YOLOv4-tiny dan CSRT, serta YOLOv4-tiny dan KCF. Pada bagian ini kedua model YOLOv4 dan YOLOv4-tiny akan tetap digunakan dalam pengujian karena mempertimbangkan kemampuan sistem yang tidak hanya akurat dalam mendeteksi, namun juga kecepatannya. Setiap skenario dibandingkan dengan nilai aktual dari hasil kuantifikasi kendaraan secara manual pada empat CCTV. Data tersebut dapat dilihat pada tabel di bawah.

Footage	Waktu (detik)	Region	Total Kendaraan (bus, mobil, motor, sepeda, truk)	Traffic Density
CCTV 1	25	region 0	1, 19, 6, 0, 0	58.5 %
		region 1	0, 11, 4, 0, 2	41.5 %
CCTV 2	126	region 0	5, 80, 4, 0, 8	76.0 %
		region 1	0, 32, 0, 0, 1	24.0 %
CCTV 3	78	region 0	3, 61, 10, 0, 3	48.3 %
		region 1	2, 35, 8, 0, 1	27.7 %
		region 2	0, 35, 3, 0, 1	24.0 %
CCTV 4	60	region 0	2, 41, 0, 5, 0	54.4 %
		region 1	6, 25, 4, 0, 0	45.6 %

Tabel 4-4 Nilai Aktual Kuantifikasi Kepadatan Lalu Lintas untuk Setiap CCTV

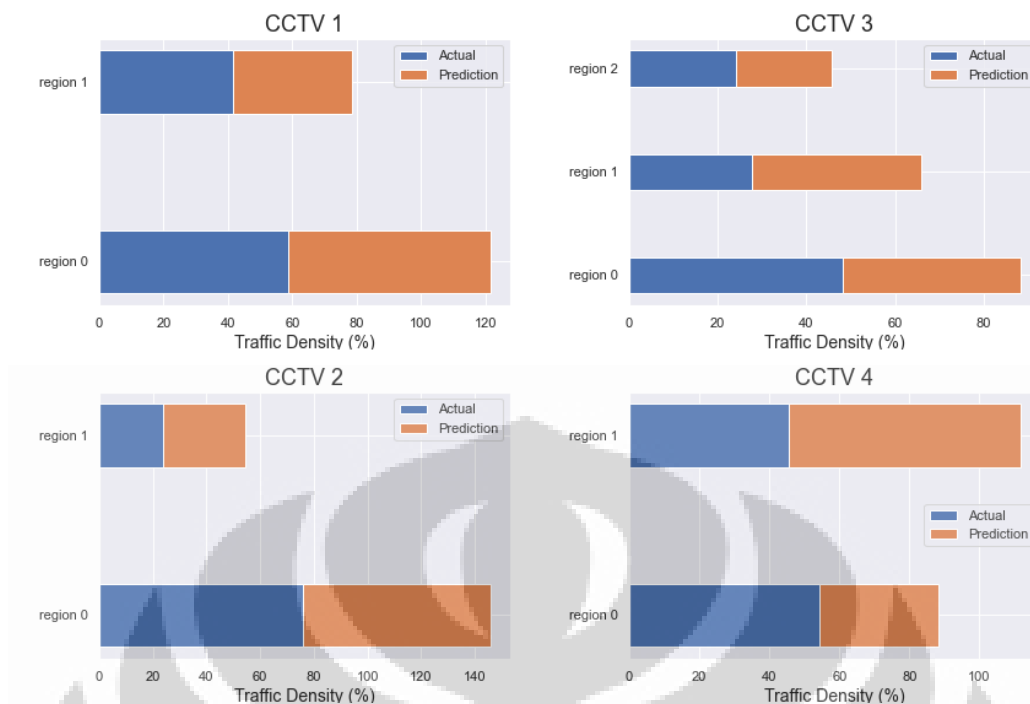
Total kendaraan untuk masing-masing *region* didapatkan dari hasil perhitungan selama waktu yang berbeda. Nilai aktual tersebut dibandingkan dengan hasil kuantifikasi kendaraan dari sistem deteksi dan juga pelacakan objek

kendaraan. Selain itu, kecepatan sistem dalam memproses data citra juga diuji dengan metrik *frame rate per-second* (FPS). Nilai FPS dicuplik setiap 5 detik bersamaan dengan jumlah kendaraan yang terbaca pada waktu tersebut. Semua pengujian yang dilakukan dari skenario yang telah disebutkan akan dianalisis dan disimpulkan menjadi hasil dari penelitian.

#### **4.3.1. Skenario 1: Pengujian Sistem Dengan YOLOv4 dan CSRT**

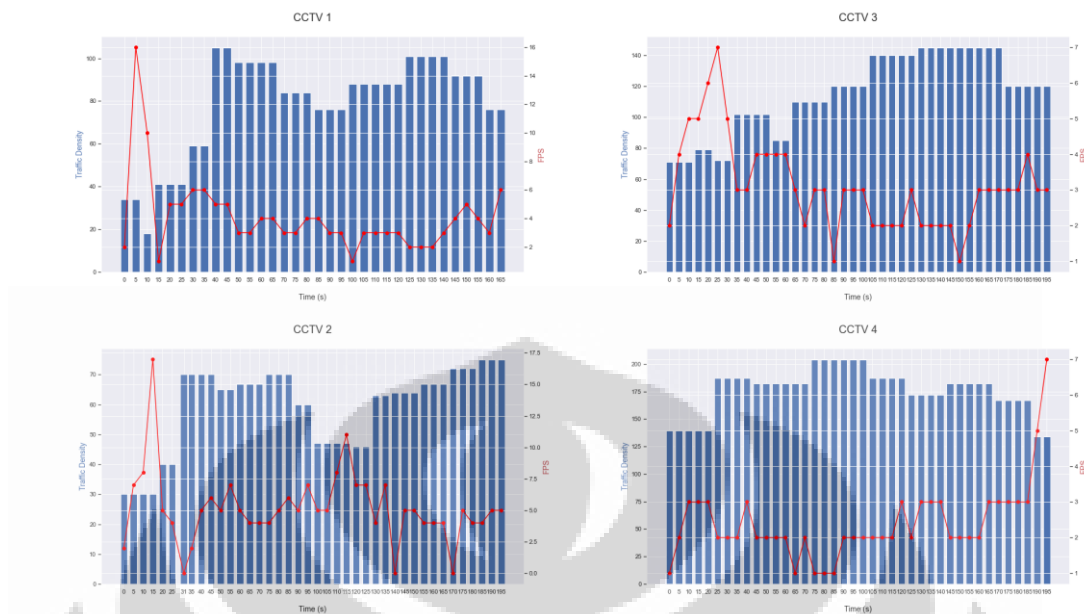
Pada skenario ini, variasi model deteksi dan pelacakan yang digunakan adalah YOLOv4 dan CSRT. Pada setiap pengujian ditampilkan grafik perbandingan antara nilai kuantifikasi dari sistem dengan nilai aktual dan grafik dari FPS yang dicuplik setiap 5 detik sekali beserta jumlah kendaraan pada saat itu. Kedua hal yang telah disebutkan menjadi parameter dalam pengukuran performa model.

Pada percobaan skenario pertama, kombinasi model deteksi dan pelacakan yang digunakan adalah YOLOv4 dan CSRT. Keduanya merupakan jenis model yang paling kompleks dalam penelitian ini. Untuk pengujian sistem dengan perbandingan antara nilai aktual kepadatan kendaraan dan nilai hasil kuantifikasi sistem dapat dilihat pada Gambar 4-7 untuk masing-masing CCTV dengan region yang berbeda. Sedangkan, untuk hasil pengujian kecepatan sistem dalam memproses data citra dapat dilihat pada Gambar 4-8.



Gambar 4-8 Grafik Perbandingan antara Nilai Kepadatan Kendaraan Hasil Prediksi Sistem dan Nilai Aktual untuk Skenario 1

Dari grafik pada Gambar 4-7 dapat dilihat bahwa 4 dari 9 region memiliki nilai hasil kuantifikasi sistem yang tidak jauh berbeda dari nilai aktual yaitu dengan rata-rata perbedaan nilai kepadatan kendaraan sekitar 5.1%. Sebagian besar dari hasil prediksi kepadatan kendaraan cenderung memiliki nilai yang lebih besar dari nilai aktualnya terutama pada CCTV 4. Hal tersebut dikarenakan banyaknya objek kendaraan dari citra CCTV 4 yang saling berhimpitan sehingga sulit untuk dideteksi dan dilacak. Selain itu, sistem kamera juga mengalami anomali dimana CCTV sering terguncang karena adanya angin kencang.



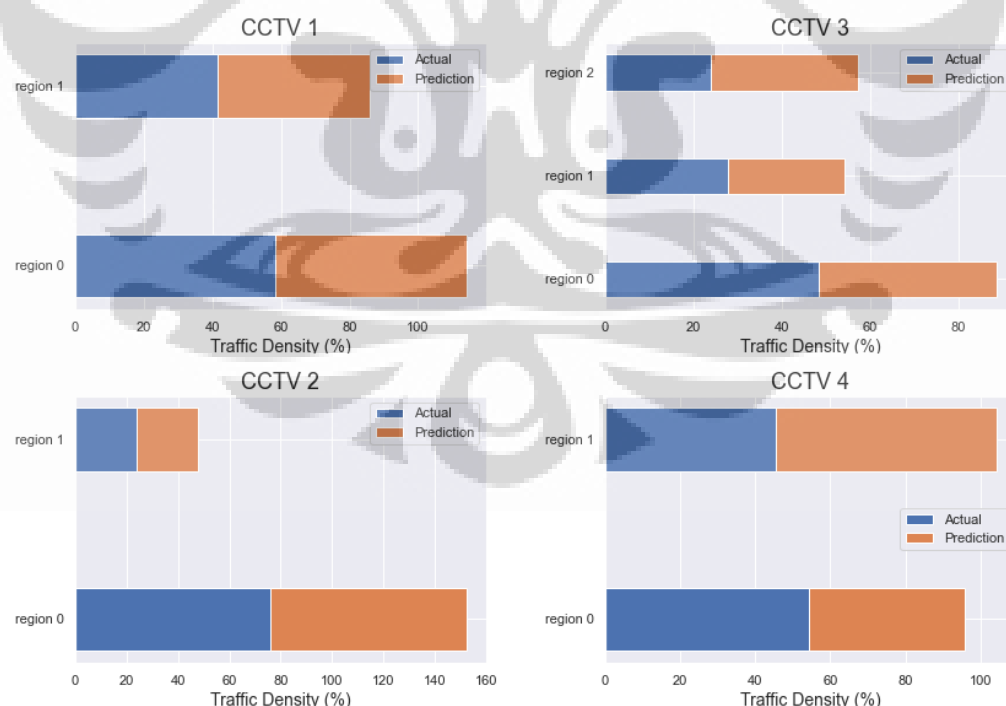
Gambar 4-9 Grafik Nilai FPS dan Kepadatan Kendaraan yang Dicuplik setiap 5 detik untuk Skenario 1

Untuk performa kecepatan sistem dapat dilihat pada Gambar 4-8 yang diukur dengan metrik FPS. Dapat dilihat bahwa untuk setiap CCTV pada 5-10 detik pertama memiliki nilai FPS yang rendah karena waktu tersebut digunakan untuk menginisialisasi model deteksi yang digunakan dalam hal ini adalah YOLOv4. Selain itu, waktu yang dibutuhkan untuk mengolah seluruh citra yang ada pada video CCTV jauh lebih lama dibandingkan dengan total waktu video aslinya. Hal tersebut terjadi karena adanya penurunan FPS ketika proses deteksi objek berlangsung terutama ketika kepadatan kendaraan meningkat. Sehingga dapat dikatakan bahwa jumlah objek kendaraan yang telah dibobotkan pada citra untuk diolah oleh sistem secara signifikan mempengaruhi FPS. Nilai rata-rata FPS untuk setiap CCTV sebesar 4 *frame rate per-second*. Dengan rata-rata waktu total yang diperlukan untuk mengolah seluruh video dari setiap CCTV adalah 192.5 detik.

### 4.3.2. Skenario 2: Pengujian Sistem Dengan YOLOv4 dan KCF

Pada skenario ini, variasi model deteksi dan pelacakan yang digunakan adalah YOLOv4 dan KCF. Pada setiap pengujian ditampilkan grafik perbandingan antara nilai kuantifikasi dari sistem dengan nilai aktual serta grafik dari FPS yang dicuplik setiap 5 detik sekali beserta jumlah kendaraan pada saat itu. Kedua hal yang telah disebutkan menjadi parameter dalam pengukuran performa model.

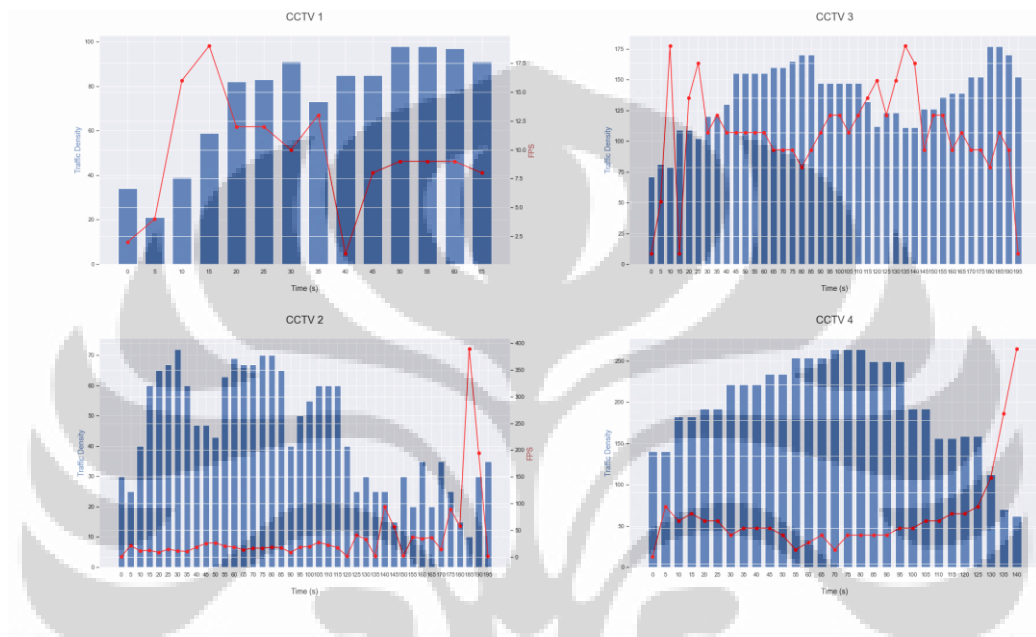
Pada percobaan skenario kedua kombinasi model deteksi yang digunakan adalah YOLOv4 dan KCF. Model deteksi yang digunakan sama seperti pada percobaan sebelumnya, namun untuk model pelacakan yang digunakan sekarang adalah KCF yang mana memiliki akurasi lebih rendah dari CSRT namun memiliki waktu proses yang lebih cepat [13]. Untuk pengujian sistem dalam kuantifikasi kepadatan kendaraan dapat dilihat pada Gambar 4-9 untuk masing-masing CCTV dengan region yang berbeda. Sedangkan, untuk hasil pengujian kecepatan sistem dalam memproses data citra dapat dilihat pada Gambar 4-10.



Gambar 4-10 Grafik Perbandingan antara Nilai Kepadatan Kendaraan Hasil Prediksi Sistem dan Nilai Aktual untuk Skenario 2



Dapat dilihat pada Gambar 4-9 bahwa 5 dari 9 region yang memiliki hasil kuantifikasi yang tidak terlalu jauh dari nilai aktual yaitu dengan rata-rata selisih 1.6%. Untuk beberapa region lainnya hasil kuantifikasi kepadatan kendaraan dari sistem dapat lebih/kurang dari nilai aktual. Selain itu, perbedaan yang paling besar juga terdapat pada CCTV 4 dikarenakan permasalahan yang telah disebutkan sebelumnya pada percobaan skenario 1.



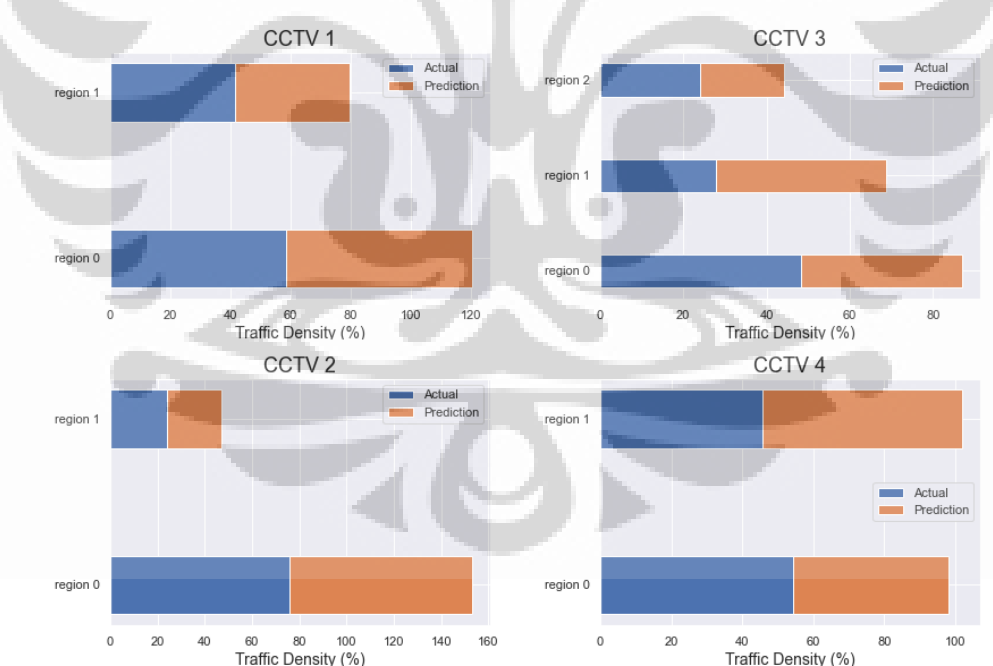
Gambar 4-11 Grafik Nilai FPS dan Kepadatan Kendaraan yang Dicuipk setiap 5 detik untuk Skenario 2

Rata-rata waktu yang dibutuhkan untuk mengolah citra berdasarkan grafik pada Gambar 4-10 untuk data dari setiap CCTV adalah sekitar 153.75 detik. Hal tersebut membuktikan adanya penurunan waktu yang dibutuhkan dalam mengolah citra karena adanya pengurangan kompleksitas model pelacakan dari CSRT menjadi KCF jika dibandingkan dengan kecepatan sistem pada skenario 1. Pada grafik dapat dilihat bahwa waktu penurun FPS diawal tetap terjadi karena adanya inisialisasi model deteksi dan nilai FPS juga berbanding terbalik dengan nilai kepadatan kendaraan yang telah dibobatkan berdasarkan hasil cuplikan data dari pengujian sistem. Rata-rata FPS untuk kombinasi dari model YOLOv4 dan KCF untuk mengolah citra adalah 17 *frame rate per-second* hampir 6 kali lebih cepat dari konfigurasi YOLOv4 dan CSRT dan cenderung lebih stabil terhadap perubahan kepadatan kendaraan.

### 4.3.3. Skenario 3: Pengujian Sistem Dengan YOLOv4-tiny dan CSRT

Pada skenario ini, variasi model deteksi dan pelacakan yang digunakan adalah YOLOv4-tiny dan CSRT. Pada setiap pengujian ditampilkan grafik perbandingan antara nilai kuantifikasi dari sistem dengan nilai aktual serta grafik dari FPS yang dicuplik setiap 5 detik sekali beserta jumlah kendaraan pada saat itu. Kedua hal yang telah disebutkan menjadi parameter dalam pengukuran performa model.

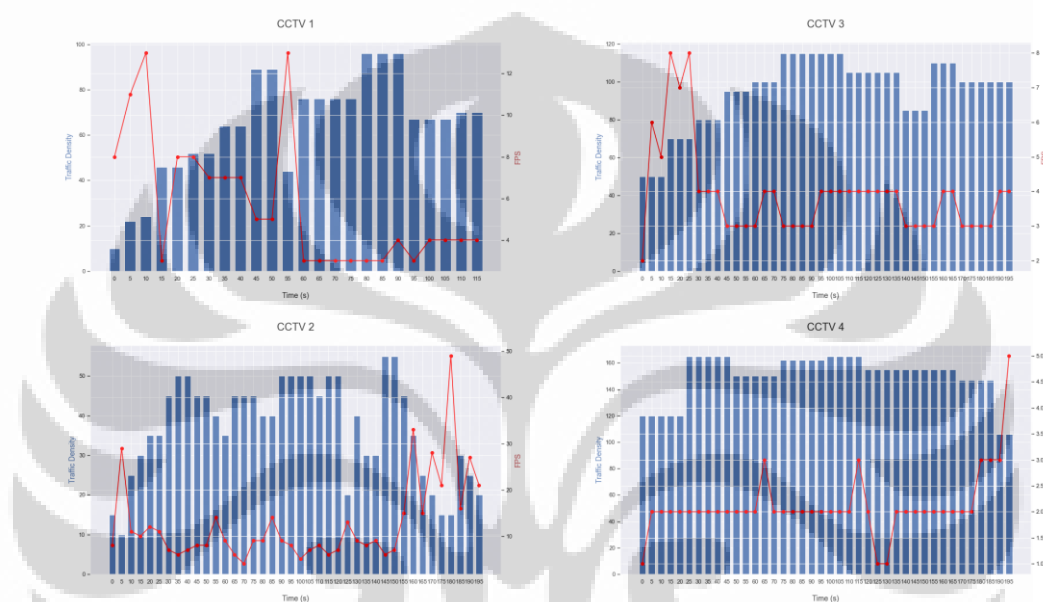
Pada percobaan skenario 3 kombinasi model yang digunakan untuk deteksi dan pelacakan objek kendaraan adalah YOLOv4-tiny dan CSRT. Terdapat degradasi kompleksitas sistem karena adanya perubahan model deteksi objek menjadi YOLOv4-tiny yang memiliki akurasi lebih rendah dari model YOLOv4. Untuk pengujian sistem dalam kuantifikasi kepadatan kendaraan dapat dilihat pada Gambar 4-11 untuk masing-masing CCTV dengan region yang berbeda. Sedangkan, untuk hasil pengujian kecepatan sistem dalam memproses data citra dapat dilihat pada Gambar 4-12.



Gambar 4-12 Grafik Perbandingan antara Nilai Kepadatan Kendaraan Hasil Prediksi Sistem dan Nilai Aktual untuk Skenario 3

Gambar 4-11 menunjukkan bahwa terdapat 6 dari 9 region dari hasil kuantifikasi kepadatan kendaraan oleh sistem yang memiliki nilai mendekati nilai

aktual yaitu dengan rata-rata perbedaan sebesar 2.3%. Beberapa region lainnya memiliki nilai kuantifikasi yang bervariasi dimana terdapat beberapa region dengan hasil kuantifikasi lebih tinggi, namun juga ada yang lebih rendah. Pada percobaan ini tidak hanya CCTV 4 saja yang memiliki nilai kuantifikasi jauh berbeda, namun permasalahan yang sama juga dialami pada CCTV 3. Hal tersebut kemungkinan dikarenakan adanya penurunan akurasi dari model deteksi yang digunakan menjadi YOLOv4-tiny.



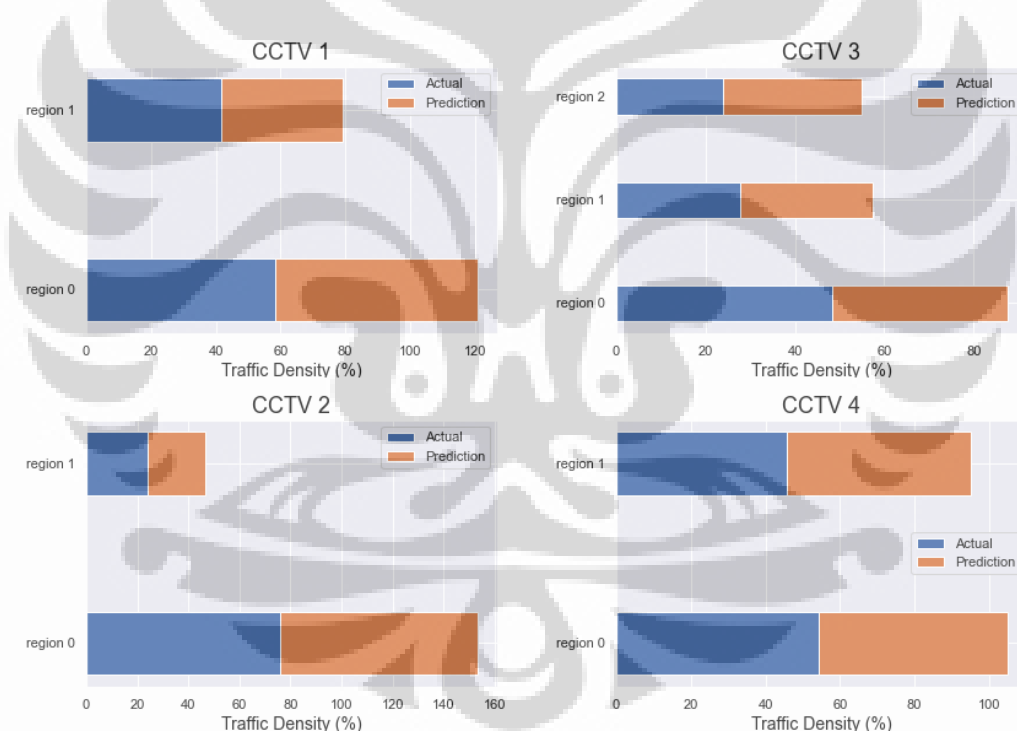
Gambar 4-13 Grafik Nilai FPS dan Kepadatan Kendaraan yang Dicuplik setiap 5 detik untuk Skenario 3

Pada Gambar 4-12 dapat dilihat bahwa rata-rata waktu yang dibutuhkan untuk mengolah seluruh citra yang dihasilkan dari masing-masing CCTV adalah 180 detik dengan rata-rata FPS sebesar 6 *frame rate per-second*. Dari angka-angka yang dihasilkan menunjukkan bahwa kombinasi model deteksi YOLOv4-tiny dan model pelacakan CSRT ini tidak terlalu berdampak pada performa kecepatan sistem, tidak seperti pada percobaan skenario 2. Walaupun model deteksi yang digunakan lebih ringan (ditunjukkan oleh waktu inisialisasi yang lebih cepat), namun model yang paling sering digunakan adalah model pelacakan. Maka dari itu, tidak adanya degradasi kompleksitas dari model pelacakan membuat peningkatan FPS sistem tidak signifikan.

#### 4.3.4. Skenario 4: Pengujian Sistem Dengan YOLOv4-tiny dan KCF

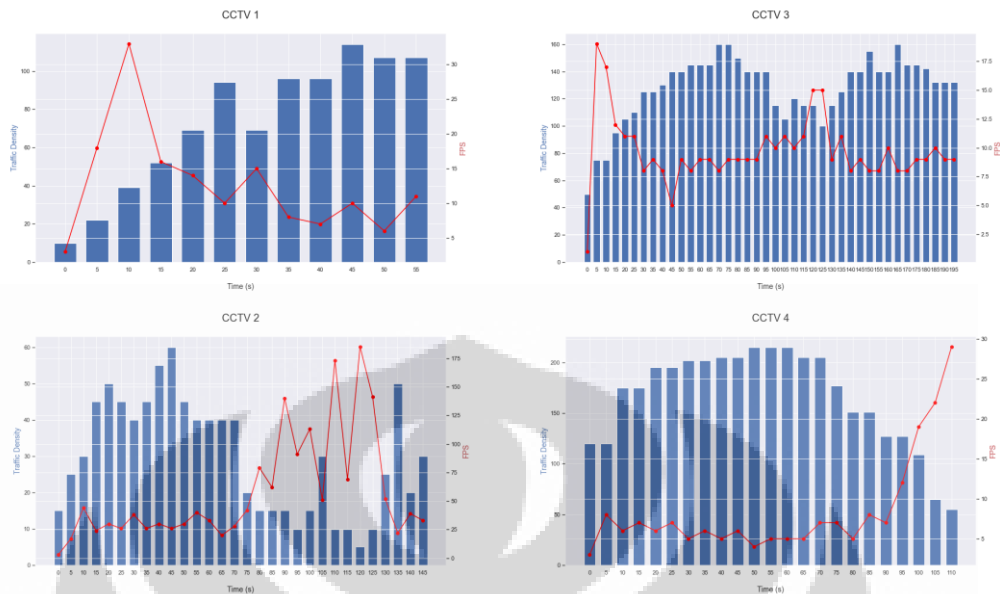
Pada skenario ini, variasi model deteksi dan pelacakan yang digunakan adalah YOLOv4-tiny dan KCF. Pada setiap pengujian ditampilkan grafik perbandingan antara nilai kuantifikasi dari sistem dengan nilai aktual serta grafik dari FPS yang dicuplik setiap 5 detik sekali beserta jumlah kendaraan pada saat itu. Kedua hal yang telah disebutkan menjadi parameter dalam pengukuran performa model.

Kedua model deteksi dan pelacakan merupakan model dengan kompleksitas paling rendah pada penelitian ini. Untuk pengujian sistem dalam kuantifikasi kepadatan kendaraan dapat dilihat pada Gambar 4-13 untuk masing-masing CCTV dengan region yang berbeda. Sedangkan, untuk hasil pengujian kecepatan sistem dalam memproses data citra dapat dilihat pada Gambar 4-14.



Gambar 4-14 Grafik Perbandingan antara Nilai Kepadatan Kendaraan Hasil Prediksi Sistem dan Nilai Aktual untuk Skenario 4

Pada Gambar 4-13 ditunjukkan bahwa terdapat 6 dari 9 region dengan hasil kuantifikasi kepadatan kendaraan sistem yang memiliki nilai rata-rata selisih nilai kuantifikasi dengan nilai aktual yaitu sekitar 2.6%. Untuk region lainnya hasil kuantifikasi dapat dikatakan cenderung lebih rendah dari nilai aktual.



Gambar 4-15 Grafik Nilai FPS dan Kepadatan Kendaraan yang Dicuipkan setiap 5 detik untuk Skenario 4

Dapat dilihat juga pada grafik dari Gambar 4-14 bahwa secara rata-rata waktu yang dibutuhkan untuk mengolah seluruh citra dari masing-masing CCTV adalah 131.25 detik dengan rata-rata pengujian metrik FPS sebesar 23 *frame rate per-second*. Artinya, kombinasi model deteksi YOLOv4-tiny dan model pelacakan KCF pada penelitian ini memiliki performa kecepatan yang paling baik dibandingkan skenario yang lainnya. Perbandingan antara FPS dan juga kepadatan kendaraan dari hasil cuplikan data setiap 5 detik juga cukup stabil jika dilihat pada grafik.

## BAB 5

### KESIMPULAN DAN SARAN

Dari penelitian sistem kuantifikasi kepadatan lalu lintas dengan metode deteksi dan pelacakan objek kendaraan di ruas jalan dapat disimpulkan bahwa:

1. Sistem kuantifikasi kepadatan lalu lintas dengan kombinasi model deteksi YOLOv4 serta model pelacakan CSRT/KCF berhasil direalisasikan.
2. Akurasi dari model deteksi YOLOv4 lebih tinggi dari versi YOLOv4 tiny yaitu dengan selisih metrik mAP sekitar 23.7%. Namun, pada tahap inisialisasi dan pemrosesan, model YOLOv4-tiny lebih cepat karena arsitektur model yang tidak terlalu kompleks.
3. Model pelacakan memiliki pengaruh yang paling besar dalam performa sistem terutama dalam perihal kecepatan pemrosesan.
4. Model KCF dapat melakukan kuantifikasi pada tahap *real-time* yaitu dengan hasil pengujian metrik FPS pada jangkauan 17-23 *frame rate per-second*. Sedangkan model CSRT hanya pada tahap interaktif yaitu 4-6 *frame rate per-second*.
5. Dalam keseluruhan pengujian sistem dapat dikatakan bahwa kombinasi yang paling tepat untuk kuantifikasi kepadatan kendaraan adalah model deteksi YOLOv4 dan model pelacakan KCF.

Untuk penelitian selanjutnya agar sistem yang telah dibuat dapat lebih baik lagi, peneliti memiliki beberapa saran:

1. Walaupun sudah menggunakan metode *transfer learning* untuk inisialisasi bobot model YOLOv4, namun model masih mengalami kesulitan untuk mendeteksi objek kendaraan yang berukuran kecil. Maka dari itu, disarankan untuk melakukan *training* model kembali dengan menggunakan citra dari CCTV yang diolah.
2. Saat ini pemilihan *area of interests* (AoIs) / region untuk diolah pada ruas jalan masih dilakukan secara manual, baiknya untuk meningkatkan efisiensi sistem pemilihan region dapat dilakukan secara otomatis

contohnya dengan melakukan segmentasi ruas jalan terlebih dahulu dengan pendekatan *image processing*.

3. Data yang dihasilkan dari sistem bersifat *time-series* sehingga dapat disimpan dan diolah dengan banyak pendekatan seperti memahami pola dari masyarakat yang berkendara, melakukan prediksi kepadatan kendaraan, serta mengotomatisasi sistem lampu merah.



## DAFTAR PUSTAKA

- [1] S. F. Laucereno, "Jokowi Ingin Ibu Kota Baru di Kaltim Jadi Smart City Rujukan Dunia," 17 4 2021. [Online]. Available: <https://finance.detik.com/properti/d-5536386/jokowi-ingin-ibu-kota-baru-di-kaltim-jadi-smart-city-rujukan-dunia>. [Accessed 6 12 2021].
- [2] S. A. Nugroho, "Ada 2 Jenis CCTV untuk Pengawasan Lalu Lintas," 9 10 2019. [Online]. Available: <https://otomotif.kompas.com/read/2017/10/09/112200515/ada-2-jenis-cctv-untuk-pengawasan-lalu-lintas>. [Accessed 5 12 2021].
- [3] C.-Y. W. H.-Y. M. L. Alexey Bochkovskiy, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *bochkovskiy2020yolov*, 2020.
- [4] N. D. Asha C. S., "Vehicle Counting for Traffic Management System using YOLO and Correlation Filter," in *Vehicle Counting for Traffic Management System using YOLO and Correlation Filter*, 2018.
- [5] J. New, "Machine Learning From Scratch: Classification, Regression, Clustering and Gradient Descent," 27 Juni 2020. [Online]. Available: <https://medium.com/swlh/machine-learning-101-classification-regression-gradient-descent-and-clustering-b3449f270dbe>. [Accessed 17 12 2021].
- [6] M. I. P. Z. Zhong-Qiu Zhao, "Object Detection with Deep Learning: A Review," in *Object Detection with Deep Learning: A Review*, Anhui, 2019.
- [7] "What Is a Machine Learning Pipeline?," 22 Agustus 2019. [Online]. Available: <https://valohai.com/machine-learning-pipeline/>. [Accessed 17 12 2021].
- [8] I. R. Team, "Computer Vision - IBM," 14 Juni 2021. [Online]. Available: <https://www.ibm.com/id-en/topics/computer-vision>. [Accessed 17 Desember 2021].



2021].

- [9] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way," 16 Desember 2018. [Online]. Available: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>. [Accessed 18 Desember 2021].
- [10] J. Redmon, "YOLO: Real-Time Object Detection," 2018. [Online]. Available: <https://pjreddie.com/darknet/yolo/>. [Accessed 5 12 2021].
- [11] S. D. R. G. A. F. Joseph Redmon, "You Only Look Once: Unified, Real-Time Object Detection," Washington, 2016.
- [12] T. V. L. C. Z. J. M. M. K. Alan Lukezic, "Discriminative Correlation Filter Tracker with Channel and Spatial Reliability," Czech Republic, 2019.
- [13] C. R. M. P. B. J. Henriques J. F., " High-Speed Tracking with Kernelized Correlation Filters," *IEEE Trans on PAMI*, pp. 583-596, 2015.
- [14] python.org, "What is Python? Executive Summary," 22 Februari 2014. [Online]. Available: <https://www.python.org/doc/essays/blurb/>. [Accessed 18 Desember 2021].
- [15] "Introduction to OpenCV-Python Tutorials," 1 Agustus 2013. [Online]. Available: [https://docs.opencv.org/4.x/d0/de3/tutorial\\_py\\_intro.html](https://docs.opencv.org/4.x/d0/de3/tutorial_py_intro.html). [Accessed 18 Desember 2021].
- [16] N. Corp, "NVIDIA CUDA-X GPU-Accelerated Libraries," 2021. [Online]. Available: <https://developer.nvidia.com/gpu-accelerated-libraries>. [Accessed 18 Desember 2021].
- [17] S. Yohanandan, "mAP (mean Average Precision) might confuse you!," 9 Juni 2020. [Online]. Available: <https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2>. [Accessed 19 Desember 2021].

- [18] E. a. Z. Y. Zhang, "Eleven Point Precision-recall Curve," *Encyclopedia of Database Systems*, pp. 981-982, 2013.
- [19] R. Khandelwal, "Evaluating performance of an object detection model," 6 Januari 2020. [Online]. Available: <https://towardsdatascience.com/evaluating-performance-of-an-object-detection-model-137a349c517b>. [Accessed 19 Desember 2021].
- [20] "Docker Overview," 9 April 2017. [Online]. Available: <https://docs.docker.com/get-started/overview/>. [Accessed 18 Desember 2021].
- [21] "What is Kubernetes?," 23 Juni 2021. [Online]. Available: <https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/>. [Accessed 19 Desember 2021].