

**DETEKSI KENDARAAN MENGGUNAKAN ALGORITMA  
YOU ONLY LOOK ONCE (YOLO) V3**



Disusun Oleh:

N a m a : Muhammad Sauqi Khatami  
NIM : 17523142

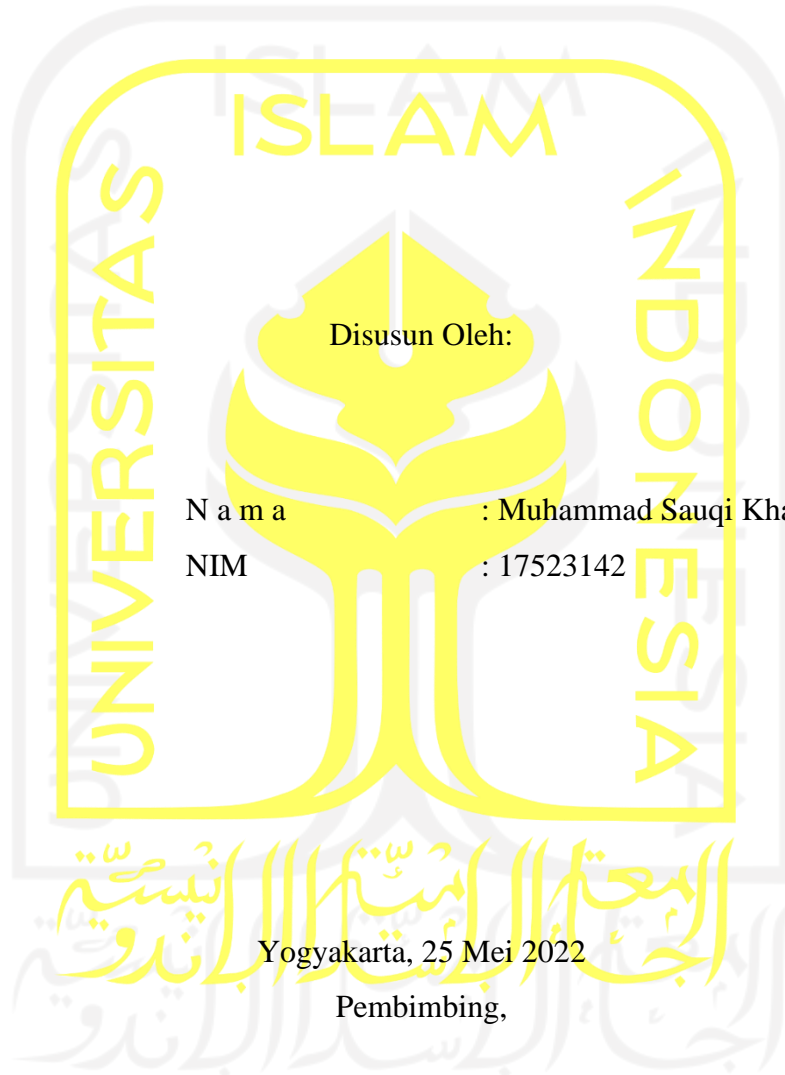
**PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA  
FAKULTAS TEKNOLOGI INDUSTRI  
UNIVERSITAS ISLAM INDONESIA**

**2022**

HALAMAN PENGESAHAN DOSEN PEMBIMBING

**DETEKSI KENDARAAN MENGGUNAKAN ALGORITMA  
YOU ONLY LOOK ONCE (YOLO) V3**

**TUGAS AKHIR**



Yogyakarta, 25 Mei 2022

Pembimbing,

( Arrie Kurniawardhani, S.Si, M.Kom )

**HALAMAN PENGESAHAN DOSEN PENGUJI**

**DETEKSI KENDARAAN MENGGUNAKAN ALGORITMA  
YOU ONLY LOOK ONCE (YOLO) V3**

**TUGAS AKHIR**

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 25 Mei 2022

Tim Penguji

Arrie Kurniawardhani, S.Si., M.Kom.



**Anggota 1**

Dhomas Hatta Fudholi, S.T.,  
M.Eng., Ph.D.



**Anggota 2**

Rian Adam Rajagede, S.Kom., M.Cs.



Mengetahui,

Ketua Program Studi Informatika – Program Sarjana  
Fakultas Teknologi Industri  
Universitas Islam Indonesia



( Dr. Raden Teduh Dirgahayu, S.T., M.Sc. )

## HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Muhammad Sauqi Khatami  
NIM : 17523142

Tugas akhir dengan judul:

### **DETEKSI KENDARAAN MENGGUNAKAN ALGORITMA *YOU ONLY LOOK ONCE (YOLO) V3***

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

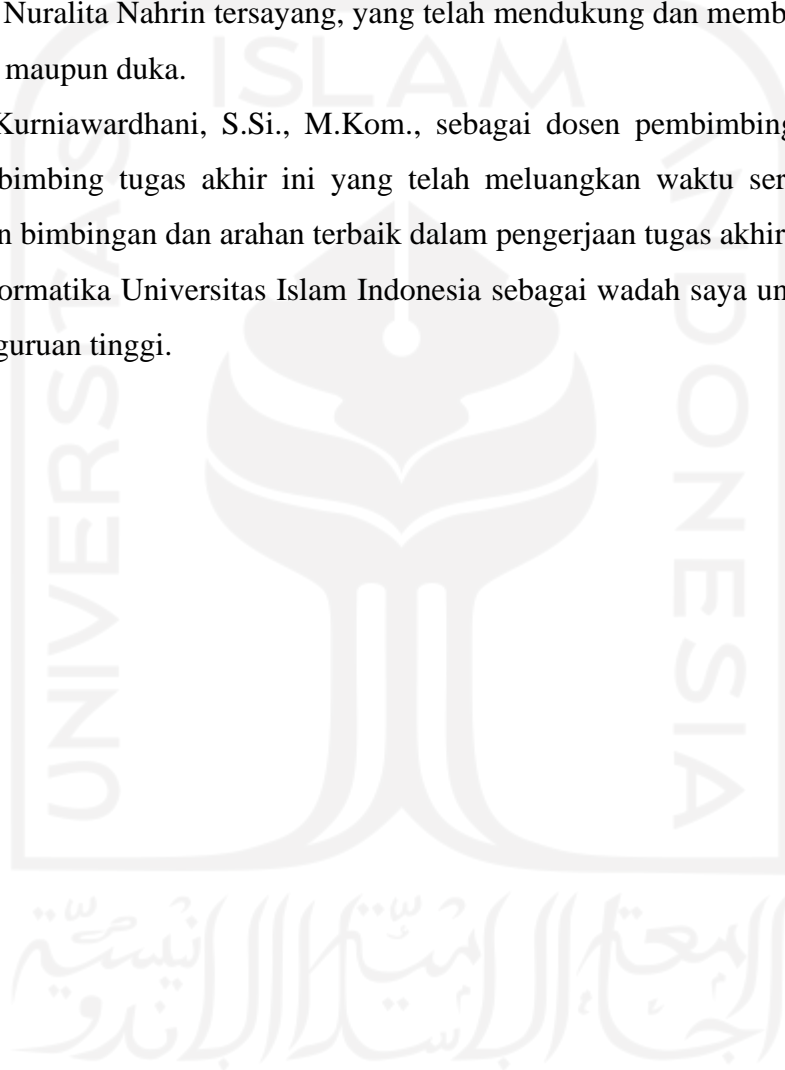
Yogyakarta, 21 April 2022

  
( Muhammad Sauqi Khatami )

## HALAMAN PERSEMBAHAN

*Alhamdulillah Robbil' Alamin* puji syukur atas segala nikmat Allah SWT yang telah diberikan kepada saya, keluarga dan sahabat serta kerabat. Tugas akhir ini saya persembahkan kepada:

- a. Bapak Syahrani dan Ibu Sri Fatimah yang saya sayangi, yang telah memberikan segalanya sampai saya bisa berada pada tahap ini.
- b. Kakak Sari Nuralita Nahrin tersayang, yang telah mendukung dan memberikan semangat dalam suka maupun duka.
- c. Ibu Arrie Kurniawardhani, S.Si., M.Kom., sebagai dosen pembimbing akademik dan dosen pembimbing tugas akhir ini yang telah meluangkan waktu serta tenaga untuk memberikan bimbingan dan arahan terbaik dalam pengerjaan tugas akhir ini.
- d. Jurusan Informatika Universitas Islam Indonesia sebagai wadah saya untuk mengemban ilmu di perguruan tinggi.



## HALAMAN MOTO

*“Allah tidak akan membebani seorang manusia melainkan sesuai dengan kemampuan manusia itu sendiri”*

(Al-Baqarah:286)



## KATA PENGANTAR

*Bismillahirrahmanirrahim.*

*Assalamualaikum Warahmatullah Wabarakatuh.*

*Alhamdulillahrobbil'alamin*, segala puji dan syukur penulis panjatkan kepada Allah SWT atas segala rahmat dan hidayah-Nya, sehingga penulis mampu menyelesaikan tugas akhir dengan judul “Deteksi Kendaraan Menggunakan Algoritma *You Only Look Once (YOLO) v3*”.

Tugas akhir ini merupakan salah satu syarat agar dapat memperoleh gelar Sarjana (S1) di Jurusan Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia. Penyelesaian tugas akhir ini tidak lepas dari segala bantuan dan bimbingan berbagai pihak. Oleh karena itu, penulis ingin mengucapkan terima kasih kepada:

1. Allah SWT atas segala nikmat dan karunia yang telah diberikan selama ini.
2. Kedua orang tua tersayang yang telah memberikan segalanya sampai penulis mampu berada di tahap ini.
3. Kakak tersayang yang telah memberikan dukungan dan arahan dalam hidup.
4. Bapak Prof. Fathul Wahid, S.T., M.Sc., Ph.D., sebagai Rektor Universitas Islam Indonesia.
5. Bapak Hendrik, S.T., M.Eng., sebagai Ketua Jurusan Informatika Universitas Islam Indonesia.
6. Bapak Dr. Raden Teguh Dirgahayu, S.T., M.Sc., sebagai Ketua Program Studi Informatika Program Sarjana Universitas Islam Indonesia.
7. Ibu Arrie Kurniawardhani, S.Si, M.Kom., sebagai dosen pembimbing akademik dan dosen pembimbing tugas akhir yang selalu memberikan arahan dan dukungan dalam pengerjaan tugas akhir ini.
8. Bapak dan Ibu dosen Jurusan Informatika yang telah memberikan ilmu dan pengetahuan yang bermanfaat.
9. Sahabat penulis, Dimas Ariyoga dan Dimas Setywan Ramadhansyah yang telah bersama-sama melalui banyak hal.
10. Teman-teman Jurusan Informatika UII angkatan 2017 PIXEL.
11. Semua pihak yang tidak bisa disebutkan satu persatu.

Seperti semua hal yang ada di dunia, tugas akhir ini tidak lepas dari ketidak sempurnaan, oleh karena itu, penulis mengharapkan kritik dan saran yang membangun agar penulis mampu menjadi lebih baik. Penulis juga mengharapkan tugas akhir ini dapat menjadi referensi atau

sumber pengetahuan bagi yang membutuhkannya di masa mendatang. Akhir kata, semoga tugas akhir ini dapat membawa manfaat kepada dunia.

*Wasalamu'alaikum Warahmatullahi Wabarakatuh.*

Yogyakarta, 21 April 2022



( Muhammad Sauqi Khatami )





## SARI

Kepadatan lalu lintas adalah satuan yang menyatakan jumlah kendaraan per satuan panjang jalan tertentu (Julianto, 2010). Dalam mendapatkan parameter jumlah kendaraan, pihak berwajib biasanya menggunakan proses perhitungan manual yang rentan akan kesalahan, selain itu perhitungan manual juga memakan banyak waktu dan tenaga (Leriansyah, 2020). Berbagai macam alat deteksi kendaraan telah dibangun, salah satu contohnya adalah alat deteksi kendaraan menggunakan jaringan deteksi dengan algoritma *Deep Learning* dan sejenisnya. Dari kebanyakan pekerjaan yang menggunakan metode ini, kerap ditemukan masalah mengenai kurang bagusnya performa deteksi kendaraan pada kondisi lalu lintas yang ramai atau macet. Hal ini disebabkan karena berdempetannya kendaraan pada kondisi lalu lintas yang ramai. Berangkat dari permasalahan tersebut, penulis berusaha untuk membangun sebuah alat pendeteksi kendaraan otomatis. Penulis akan melatih dua jaringan YOLO v3 yang mampu mendeteksi kendaraan dengan menggunakan dua skenario pelabelan *dataset* berdasarkan sudut pengambilan citra yaitu skenario satu yang melakukan pelabelan citra tanpa mempedulikan sudut pengambilan citra dan skenario dua yang melakukan pelabelan citra dengan mempedulikan sudut pengambilan citra (dari depan, kanan, kiri, dan separuh). Dengan menggunakan dua skenario pelabelan ini, diharapkan akan terjadi peningkatan performa deteksi kendaraan yang dilakukan oleh jaringan YOLO v3 yang dilatih.

Dari pengujian kedua jaringan yang telah dilatih pada *dataset* rekaman lalu lintas dari website ATCS (*Area Traffic Control System*) Kota Bandung (<http://atcs-dishub.bandung.go.id/index.php#>) dan Banyumas (<http://atcs.banyumaskab.go.id/#>), kedua bobot jaringan mendapatkan rata-rata nilai performa jaringan di atas 70%, hal ini menunjukkan bahwa kedua jaringan ini mampu mendeteksi kendaraan pada *dataset* dari penulis dengan baik dan terdapat peningkatan yang tidak terlalu signifikan pada jaringan dua.

Kata kunci: Deteksi Kendaraan, Pelabelan Jaringan, *You Only Look Once* v3.

## GLOSARIUM

<i>ATCS</i>	<i>Area Traffic Control System.</i>
<i>Dataset</i>	Kumpulan data.
<i>Frame</i>	Satu citra per satuan detik pada video.
<i>Labelling</i>	Proses pelabelan citra agar citra dapat dijadikan data latih jaringan.



## DAFTAR ISI

HALAMAN JUDUL .....	i
HALAMAN PENGESAHAN DOSEN PEMBIMBING.....	ii
HALAMAN PENGESAHAN DOSEN PENGUJI .....	<b>Error! Bookmark not defined.</b>
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR .....	iv
HALAMAN PERSEMBAHAN .....	v
HALAMAN MOTO .....	vi
KATA PENGANTAR .....	vii
SARI.....	ix
GLOSARIUM.....	x
DAFTAR ISI .....	xi
DAFTAR TABEL .....	xii
DAFTAR GAMBAR.....	xiii
BAB I PENDAHULUAN .....	1
1.1 Latar Belakang .....	1
<b>1.2 Rumusan Masalah</b> .....	1
1.3 Batasan Masalah .....	2
1.4 Tujuan Penelitian .....	2
1.5 Manfaat Penelitian .....	2
1.6 Metodologi Penelitian .....	2
1.7 Rancangan Sistematika Penulisan.....	3
BAB II LANDASAN TEORI .....	4
2.1 Landasan Teori.....	4
2.1.1 Kendaraan.....	4
2.1.2 Deteksi Objek .....	5
2.2 Kajian Pustaka Penelitian Terdahulu .....	23
BAB III METODOLOGI PENELITIAN .....	28
3.1 Tahapan Penelitian .....	28
3.2 Uraian Penelitian.....	28
3.2.1 Perumusan Masalah.....	28
3.2.2 Landasan Teori .....	28
3.2.3 Pengumpulan Data .....	29
3.2.4 Pelatihan Jaringan.....	34
3.2.5 Pengujian dan Penarikan Kesimpulan .....	36
BAB IV HASIL DAN PEMBAHASAN .....	39
4.1 Implementasi .....	39
4.1.1 Pengumpulan Data .....	39
4.1.2 Pelabelan Objek.....	41
4.1.3 <i>Data Augmentation</i> .....	43
4.1.4 Pelatihan Jaringan.....	48
4.1.5 Proses Deteksi Jaringan YOLO v3.....	53
<b>4.2 Pengujian Performa Jaringan</b> .....	57
4.2.1 Pengujian Performa Jaringan pada <i>Dataset</i> Penulis.....	57
4.2.2 Rangkuman Pengujian Performa Jaringan .....	67
BAB V KESIMPULAN DAN SARAN.....	69
5.1 Kesimpulan .....	69
5.2 Saran.....	69

DAFTAR PUSTAKA .....	71
----------------------	----

### DAFTAR TABEL

Tabel 2.1 Tabel rangkuman kajian pustaka dari penelitian terdahulu .....	23
Tabel 3.1 Tabel perbandingan citra asli dan citra hasil teknik <i>augmentation</i> .....	32
Tabel 3.2 Tabel perbandingan total kendaraan aktual dan prediksi.....	37
Tabel 3.3 Tabel <i>Confusion Matrix</i> .....	38
Tabel 4.1 Tabel perbandingan total kendaraan aktual dan total kendaraan prediksi ketiga jaringan pada <i>dataset</i> penulis .....	57
Tabel 4.2 Tabel <i>Confusion Matrix</i> hasil deteksi kendaraan jaringan skenario satu pada <i>dataset</i> penulis .....	64
Tabel 4.3 Tabel <i>Confusion Matrix</i> hasil deteksi kendaraan jaringan skenario dua pada <i>dataset</i> penulis .....	66
Tabel 4.4 Tabel rangkuman perbandingan total deteksi kendaraan pada masing-masing bobot jaringan.....	67
Tabel 4.5 Tabel rangkuman perbandingan performa jaringan pada masing-masing bobot jaringan.....	68

## DAFTAR GAMBAR

Gambar 2.1 Deteksi objek pada kendaraan <i>autonomous</i> .....	6
Gambar 2.2 Plot skema (a) deteksi objek “ <i>One-stage Detector</i> ” dan (b) deteksi objek “ <i>Two-stage Detector</i> ” .....	7
Gambar 2.3 <i>Road map</i> perkembangan deteksi objek selama dua dekade terakhir .....	8
Gambar 2.4 Komponen <i>bounding box</i> .....	9
Gambar 2.5 Contoh <i>Intersection Over Union</i> .....	10
Gambar 2.6 Perhitungan IOU .....	10
Gambar 2.7 Alur kerja algoritma YOLO dalam pendeteksian objek pada citra .....	11
Gambar 2.8 Prediksi <i>Bounding Box</i> yang beragam .....	12
Gambar 2.9 Pembuangan <i>bounding box</i> yang nilainya kurang dari <i>threshold</i> yang ditentukan .....	12
Gambar 2.10 Arsitektur YOLO .....	13
Gambar 2.11 Contoh operasi konvolusi pada lapisan <i>Convolutional</i> .....	14
Gambar 2.12 Contoh operasi <i>Max Pooling</i> .....	15
Gambar 2.13 Arsitektur Darknet-53 .....	17
Gambar 2.14 Arsitektur jaringan YOLO v3 .....	18
Gambar 2.15 Atribut <i>Bounding Box</i> .....	19
Gambar 2.16 <i>Anchors</i> dan <i>bounding box</i> objek deteksi .....	20
Gambar 2.17 <i>Confusion Matrix</i> .....	22
Gambar 3.1 Tahapan penelitian .....	28
Gambar 3.2 Situs ATCS Kota Bandung .....	29
Gambar 3.3 Situs ATCS Banyumas.....	30
Gambar 3.4 Contoh satu frame citra dalam format “.jpg” .....	30
Gambar 3.5 Alur pengumpulan data.....	34
Gambar 3.6 Alur pelatihan jaringan.....	35
Gambar 3.7 Arsitektur jaringan deteksi kendaraan YOLO v3 pada penelitian ini.....	36
Gambar 4.1 Kode program untuk konversi video rekaman menjadi frame citra.....	39
Gambar 4.2 <i>Frames</i> citra yang telah dikonversi .....	40
Gambar 4.3 <i>Frames</i> citra yang kurang memiliki informasi signifikan.....	41
Gambar 4.4 <i>Frames</i> citra yang memiliki informasi yang hampir sama .....	41
Gambar 4.5 Proses pelabelan citra.....	42
Gambar 4.6 Isi dari <i>File</i> “.txt” .....	42

Gambar 4.7 Isi dari file “.txt” pelabelan skenario dua.....	43
Gambar 4.8 Kode program mount Google Drive .....	44
Gambar 4.9 Kode program <i>unzip file dataset</i> .....	44
Gambar 4.10 Dataset citra yang telah di- <i>unzip</i> .....	44
Gambar 4.11 Kode program menghitung total <i>file</i> citra dan anotasi.....	45
Gambar 4.12 Total file citra dan anotasi.....	45
Gambar 4.13 Kode program <i>import libraries</i> .....	45
Gambar 4.14 Kode program untuk membangun objek “Augmentor” .....	46
Gambar 4.15 Kode program untuk menambahkan teknik augmentasi ke objek “Augmentor” .....	46
Gambar 4.16 Citra hasil augmentation .....	47
Gambar 4.17 Kode program <i>clone, configure, dan make</i> Darknet .....	48
Gambar 4.18 Kode program untuk konfigurasi <i>hyperparameter</i> jaringan YOLO v3 untuk <i>dataset</i> pelabelan skenario satu.....	49
Gambar 4.19 Kode program untuk konfigurasi <i>hyperparameter</i> jaringan YOLO v3 untuk <i>dataset</i> pelabelan skenario dua .....	49
Gambar 4.20 Kode program untuk membuat <i>file</i> “obj.names” dan “obj.data” skenario satu .....	50
Gambar 4.21 Kode program untuk membuat <i>file</i> “obj.names” dan “obj.data” skenario dua .....	50
Gambar 4.22 Kode program untuk menyimpan file “yolov3_testing.cfg” dan “classes.txt” ke dalam Google Drive penulis .....	51
Gambar 4.23 Kode program untuk membuat <i>folder</i> “obj” dan <i>unzipping dataset</i> .....	51
Gambar 4.24 Kode program untuk membuat <i>file</i> “train.txt” dan menulis seluruh nama <i>file</i> <i>dataset</i> ke dalam file “train.txt” .....	51
Gambar 4.25 Kode program untuk mengunduh bobot <i>pretrained</i> “darknet53.conv.74” .....	51
Gambar 4.26 Kode program untuk pelatihan jaringan deteksi kendaraan YOLO v3.....	52
Gambar 4.27 Kode program untuk pelanjutan pelatihan jaringan deteksi kendaraan YOLO v3 yang terganggu.....	52
Gambar 4.28 <i>File</i> “yolov3_training_last.weights” dan “yolov3_training_final.weight” hasil pelatihan jaringan.....	53
Gambar 4.29 Kode program <i>import libraries</i> .....	53
Gambar 4.30 Kode program load jaringan deteksi kendaraan YOLO v3 skenario satu.....	54
Gambar 4.31 Kode program <i>load</i> nama kelas keluaran ke dalam list “classes” .....	54

Gambar 4.32 Kode program pembacaan citra yang ingin diuji .....	54
Gambar 4.33 Kode program <i>define</i> “blob” dan menjadikan “blob” sebagai masukan jaringan .....	54
Gambar 4.34 Kode program <i>define</i> “blob” dan menjadikan “blob” sebagai masukan jaringan .....	55
Gambar 4.35 Kode program pembuatan <i>bounding box</i> .....	55
Gambar 4.36 Kode program penulisan <i>bounding box</i> ke citra .....	56
Gambar 4.37 Hasil citra deteksi .....	57
Gambar 4.38 Total kendaraan deteksi .....	57



## **BAB I**

### **PENDAHULUAN**

#### **1.1 Latar Belakang**

Kepadatan lalu lintas adalah satuan yang menyatakan jumlah kendaraan per satuan panjang jalan tertentu (Julianto, 2010). Satuan ini dapat digunakan sebagai berbagai macam parameter penting dalam beragam pekerjaan, misalnya untuk melakukan perencanaan jalan seperti penetapan geometri serta penentuan tebal perkerasan, evaluasi suatu taksiran ekonomis di bidang jalan, dan sebagai informasi bagi instansi atau masyarakat umum (Departemen Pekerjaan Umum, 2005). Dalam mendapatkan parameter jumlah kendaraan, pihak berwajib biasanya menggunakan proses perhitungan manual yang rentan akan kesalahan, selain itu perhitungan manual juga memakan banyak waktu dan tenaga (Leriansyah, 2020). Seiring dengan berkembangnya teknologi, peneliti dan pihak lainnya berusaha untuk membangun sebuah alat pendeteksi kendaraan otomatis yang mampu memudahkan proses perhitungan jumlah kendaraan. Berbagai macam alat deteksi kendaraan telah dibangun, salah satu contohnya adalah alat deteksi kendaraan menggunakan jaringan deteksi dengan algoritma *Deep Learning* dan sejenisnya (*Convolutional Neural Network, Region Based Convolutional Neural Network, You Only Look Once*) (Pratama & Rasywir, 2021).

Dari kebanyakan pekerjaan yang menggunakan metode ini, kerap ditemukan masalah mengenai kurang bagusya performa deteksi kendaraan pada kondisi lalu lintas yang ramai atau macet. Hal ini disebabkan karena berdempetannya kendaraan pada kondisi lalu lintas yang ramai. Berdempetannya kendaraan ini akan menyebabkan tergabungnya banyak fitur dari objek kendaraan yang tentunya hal ini akan mempersulit jaringan untuk memisahkan batas-batas antar objek kendaraan (Liu, 2020). Berbagai solusi telah diterapkan untuk menyelesaikan masalah ini, salah satunya adalah dengan melakukan pelabelan secara *multiview class* seperti yang dilakukan pada penelitian Tang et al. (2018). Pelabelan *multiview class* akan melabeli satu objek ke dalam berbagai macam kelas berdasarkan sudut pandangnya. Pelabelan *multiview class* dalam penelitian Tang et al. (2018) terbukti mampu meningkatkan nilai performa jaringan deteksi. Berangkat dari permasalahan ini, peneliti ingin membandingkan jaringan deteksi dengan pelabelan *multiview class* dan jaringan deteksi dengan pelabelan pada umumnya.

#### **1.2 Rumusan Masalah**

Dalam penelitian ini, ditentukan beberapa rumusan masalah yaitu:



- a. Bagaimana caranya membangun model yang mampu mendeteksi kendaraan secara efektif?
- b. Seberapa besar tingkat performa deteksi jaringan dengan pelabelan skenario satu dan skenario dua?

### 1.3 Batasan Masalah

Dalam penelitian ini, ditentukan beberapa batasan masalah agar masalah yang diteliti tidak terlalu luas dan menyimpang, batasan-batasan tersebut ialah:

- a. Terdapat dua skenario pelabelan data pelatihan sesuai dengan sudut pengambilan citra.
- b. Pengambilan citra untuk *dataset* pembangunan jaringan diambil dari situs ATCS Bandung dan ATCS Banyumas.
- c. Kendaraan yang dijadikan sebagai objek deteksi adalah sepeda motor, mobil, dan kendaraan besar (truk dan bus).
- d. Waktu pengambilan citra *dataset* adalah di siang hari.
- e. Kondisi cuaca pengambilan citra *dataset* adalah cerah.

### 1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah membangun jaringan deteksi kendaraan menggunakan algoritma *You Only Look Once* (YOLO) dengan dua skenario pelabelan data pelatihan serta mengetahui performa dari kedua jaringan yang telah dilatih.

### 1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah:

- a. Memudahkan proses peningkatan kinerja lalu lintas.
- b. Dapat digunakan untuk pengembangan alat atau sistem lainnya yang berhubungan dengan pengaturan lalu lintas (seperti deteksi kecelakaan atau sistem lampu lalu lintas adaptif).
- c. Dapat digunakan sebagai referensi untuk pekerjaan-pekerjaan lainnya yang berkaitan dengan deteksi objek kendaraan.

### 1.6 Metodologi Penelitian

- a. Perumusan masalah, merupakan tahap untuk mengidentifikasi masalah-masalah yang ingin diangkat menjadi tema penelitian. Pada tahap ini juga akan ditentukan tujuan dan manfaat penelitian.

- b. Landasan teori, merupakan tahap untuk mempelajari segala teori dan pekerjaan terdahulu yang berkaitan mengenai tema penelitian (lalu lintas dan deteksi objek).
- c. Pengumpulan data, merupakan tahap untuk mengumpulkan dan menyiapkan (melabeli dan melakukan *augmentation*) *dataset* yang dibutuhkan untuk membangun jaringan deteksi kendaraan.
- d. Pelatihan jaringan, merupakan tahap untuk melakukan pembangunan jaringan deteksi kendaraan menggunakan algoritma YOLO v3.
- e. Pengujian dan penarikan kesimpulan, merupakan tahap untuk melakukan pengujian jaringan deteksi kendaraan YOLO v3 yang telah dilatih dan dari hasil pengujian tersebut, akan ditarik kesimpulan.

## 1.7 Rancangan Sistematika Penulisan

Penulisan laporan penelitian ini dibagi menjadi beberapa bagian agar mempermudah pencarian informasi yang dibutuhkan serta sebagai bukti penyelesaian pekerjaan yang sistematis, berikut pembagian sistematika penulisan laporan ini:

**Bab I Pendahuluan**, berisi pemaparan mengenai latar belakang permasalahan mengapa dibutuhkannya alat yang mampu mendeteksi kendaraan secara otomatis, serta hal lainnya yang mendasari penelitian ini seperti rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, serta rancangan sistematika penulisan laporan.

**Bab II Landasan Teori**, berisi pemaparan mengenai teori-teori yang berhubungan dengan penelitian ini (lalu lintas dan deteksi objek) serta kajian dari penelitian dan pekerjaan terdahulu yang berkaitan dengan penelitian ini.

**Bab III Metodologi Penelitian**, berisi pemaparan rancangan model (penentuan parameter pelatihan model seperti *batch size*, *learning rate*, dan *steps*), alur proses pelatihan model deteksi kendaraan YOLO serta alur proses pengujian model deteksi kendaraan.

**Bab IV Hasil dan Pembahasan**, berisi pemaparan mengenai hasil serta pembahasan pembangunan dan pengujian model deteksi kendaraan dengan model deteksi kendaraan lainnya.

**Bab V Kesimpulan dan Saran**, kesimpulan dari penelitian serta saran untuk meningkatkan kualitas pekerjaan pada penelitian ini kepada pihak-pihak yang akan menggunakan penelitian ini sebagai referensinya.

## **BAB II**

### **LANDASAN TEORI**

#### **2.1 Landasan Teori**

##### **2.1.1 Kendaraan**

Menurut Pasal 1 Angka 7 UU Nomor 22 Tahun 2009 Lalu Lintas dan Angkutan Jalan, kendaraan adalah suatu sarana angkut di jalan yang terdiri atas kendaraan bermotor dan kendaraan tidak bermotor. Kendaraan bermotor adalah setiap kendaraan yang digerakkan oleh peralatan mekanik berupa mesin selain kendaraan yang berjalan di atas rel, sedangkan kendaraan tidak bermotor setiap kendaraan yang digerakkan oleh tenaga manusia dan/atau hewan (Amwin, 2021). Selanjutnya, Pasal 1 Angka 7 UU Nomor 22 Tahun 2009 menjelaskan bahwa sepeda motor, mobil penumpang, mobil bus, mobil barang, dan kendaraan khusus termasuk ke dalam jenis kendaraan bermotor, sedangkan kendaraan yang digerakkan oleh tenaga manusia (seperti becak dan sepeda) dan kendaraan yang digerakkan oleh tenaga hewan (seperti delman) termasuk ke dalam jenis kendaraan tidak bermotor.

Setiap jenis kendaraan mempunyai karakteristiknya masing-masing, baik itu karakteristik bentuk, ukuran atau kemampuan kendaraan. Karakteristik-karakteristik ini mempunyai pengaruh terhadap desain pembuatan jalan, seperti desain geometrik jalan, lebar jalan, tingkat kekerasan jalan agar mampu menopang kendaraan, penempatan papan tanda rambu lalu lintas yang bisa dilihat oleh pengemudi, dan sebagainya. Andika (2018) memaparkan bahwa karakteristik tersebut dibagi menjadi tiga yaitu:

- a. Karakteristik statis yang meliputi dimensi, berat, dan kemampuan manuver kendaraan.
- b. Karakteristik kinematik yang meliputi kemampuan kendaraan untuk melakukan percepatan dan perlambatan.
- c. Karakteristik dinamis yang meliputi kemampuan kendaraan selama bergerak, diantaranya tahanan terhadap udara, tahanan dalam menghadapi tanjakan, tenaga dan pengereman.

Karena pada penelitian ini akan digunakan data rekaman lalu lintas yang memperlihatkan visual kendaraan sebagai objek deteksi, maka pada subbab ini hanya akan dijelaskan karakteristik yang berhubungan dengan visual kendaraan yaitu karakteristik statis dimensi kendaraan.

Karakteristik statis dimensi kendaraan adalah karakteristik sebuah kendaraan yang didapatkan saat kendaraan tersebut berada dalam kondisi diam atau statis (Parmar, 2021).

Karakteristik ini berhubungan dengan geometri, desain serta sistem kendali kendaraan yang diantaranya adalah dimensi, berat, dan kemampuan manuver kendaraan.

Karakteristik statis dimensi kendaraan adalah karakteristik sebuah kendaraan yang berhubungan dengan panjang, lebar, dan tinggi kendaraan tersebut. Karakteristik-karakteristik ini akan mempengaruhi desain dan pembangunan jalan, seperti lebar jalur, pola lintasan kendaraan, dan lebar median saat kendaraan melakukan putaran balik arah (Andika, 2018). Karakteristik inilah yang juga dapat diekstrak atau diambil sebagai ciri atau fitur dari sebuah kendaraan pada penelitian ini.

### 2.1.2 Deteksi Objek

Deteksi objek merupakan salah satu teknik dari bidang *Computer Vision* (salah satu bidang kecerdasan buatan yang membahas bagaimana mesin dapat melihat layaknya manusia) (Aningtiyas et al., 2020). Mengutip dari Zou et al. (2019), deteksi objek adalah “*an important computer vision task that deals with detecting instances of visual objects of a certain class (such as humans, animals, or cars) in digital images*” atau bisa diartikan bahwa deteksi objek merupakan proses deteksi atau penemuan *instances* (contoh) dari suatu objek visual kelas tertentu (seperti manusia, hewan, dan kendaraan) dalam sebuah citra digital. Deteksi objek yang merupakan masalah fundamental dari bidang *Computer Vision* memiliki tujuan untuk mengembangkan model dan teknik komputasi yang mampu menyediakan informasi paling dasar yang dibutuhkan oleh aplikasi *Computer Vision* yaitu: *What objects are where?* (Zou et al., 2019). Selain itu, deteksi objek juga meliputi berbagai tugas *Computer Vision* seperti segmentasi *instance*, *Image Captioning* (penamaan citra), serta *Object Tracking* (pelacakan objek).

Deteksi objek banyak digunakan di berbagai bidang pekerjaan, mulai dari bidang kesehatan sampai ke bidang keamanan. Untuk bidang transportasi sendiri, deteksi objek biasanya digunakan untuk mendeteksi objek-objek yang dapat mengganggu kenyamanan berkendara seperti yang dilakukan oleh Gao et al. (2018) pada penelitiannya mengenai klasifikasi objek pada lingkungan kendaraan *autonomous* (kendaraan yang mampu mengendalikan dirinya sendiri, tanpa bantuan manusia) seperti pada Gambar 2.1.



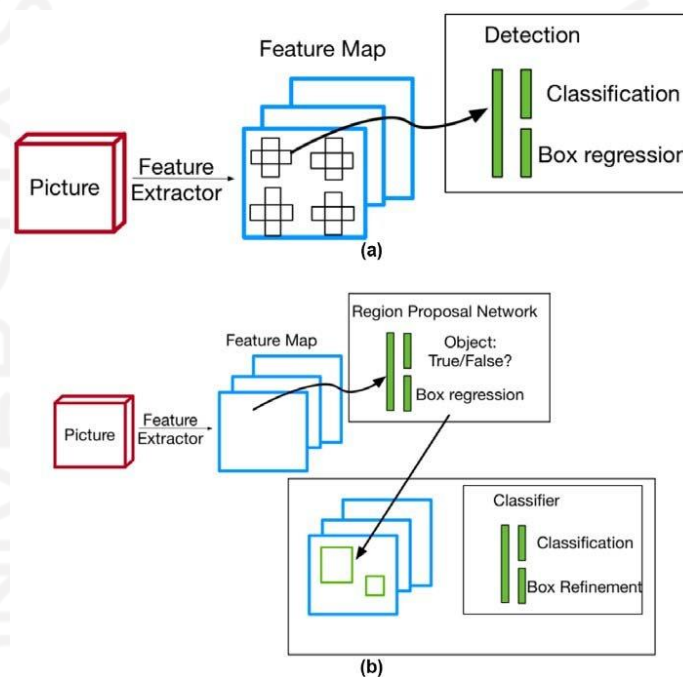
Gambar 2.1 Deteksi objek pada kendaraan *autonomous*

Sumber: Bin Issa (2021)

Dalam dua dekade terakhir, deteksi objek telah melalui dua periode yaitu “periode deteksi objek tradisional (sebelum 2014)” dan “periode deteksi objek berbasis *Deep Learning* (sesudah 2014)” (Zou et al., 2019). Berdasarkan Zou et al. (2019), “periode deteksi objek tradisional” adalah periode dimana deteksi objek dilakukan dengan menggunakan metode-metode yang terkesan tradisional karena pada saat itu belum ditemukan representasi citra yang efektif sehingga mengharuskan seseorang untuk mendesain representasi fitur yang rumit dan berbagai kemampuan yang mampu mempercepat proses komputasi untuk mengurangi penggunaan sumber daya yang pada saat itu tentunya relatif terbatas, sedangkan “periode deteksi objek berbasis *Deep Learning*” adalah periode dimana deteksi objek dilakukan dengan menggunakan metode-metode yang lebih efektif dan efisien karena model deteksi objek berbasis *Deep Learning* dapat mempelajari representasi fitur yang rumit dan bertingkat tinggi. *Deep Learning* adalah salah satu bidang pada *Machine Learning* (ilmu yang memungkinkan komputer mampu mengembangkan perilaku berdasarkan data empiris, seperti data sensor atau basis data) yang menggunakan banyak lapisan pada pengolahan informasi nonlinier dalam melakukan ekstraksi fitur, pengenalan pola, dan klasifikasi (Amwin, 2021).

Pada era deteksi objek berbasis *Deep Learning*, deteksi objek terbagi menjadi dua kategori yaitu “*Two-stage Detection*” dan “*One-stage Detection*” (Zou et al., 2019). Deteksi objek “*Two-stage Detection*” mempunyai dua *stages* (tahapan) dalam melakukan proses deteksi objek, tahap pertama yaitu memprediksi kandidat *object proposal* atau *bounding box* (kotak pembatas untuk mengidentifikasi posisi dan tipe objek) pada citra dan selanjutnya pada tahap kedua dilakukan proses klasifikasi dan regresi terhadap kandidat *bounding box* yang nantinya akan menghasilkan *bounding box* yang menunjukkan letak serta tipe objek,

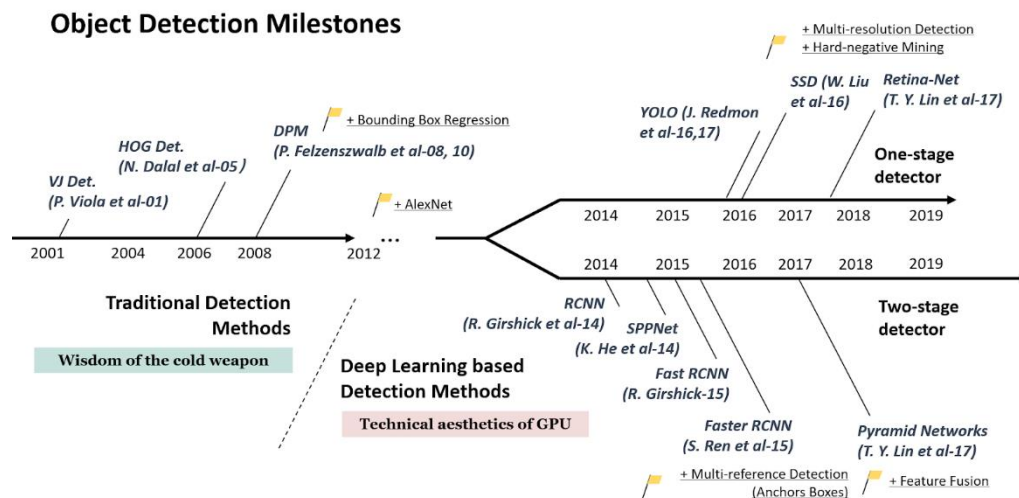
sebaliknya deteksi objek “*One-stage Detection*” hanya membutuhkan satu *stage* (tahapan) untuk melakukan proses deteksi objek (Lohia et al., 2021). Banyaknya tahapan yang dibutuhkan deteksi objek “*Two-stage Detection*” meningkatkan akurasi lokalisasi (penentuan posisi) dan rekognisi (pengenalan tipe) objek namun juga meningkatkan waktu yang dibutuhkan untuk melakukan proses deteksi, sebaliknya deteksi objek “*One-stage Detection*” mengurangi waktu yang dibutuhkan untuk melakukan proses deteksi namun juga mengurangi akurasi lokalisasi dan rekognisi objek (Zou et al., 2019). Tahapan umum pada deteksi objek “*One-stage Detection*” dan “*Two-stage Detection*” dapat dilihat pada Gambar 2.2.



Gambar 2.2 Plot skema (a) deteksi objek “*One-stage Detector*” dan (b) deteksi objek “*Two-stage Detector*”

Sumber: Kemajou et al. (2019)

Viola Jones *Detector*, HOG (*Histogram of Oriented Gradients*) *Detector*, dan DPM (*Deformable Part-based Model*) merupakan metode deteksi objek yang tergolong tradisional, sedangkan RCNN (*Region-based Convolutional Neural Networks*), *Fast RCNN*, *Faster RCNN*, SPPNet, merupakan metode deteksi objek “*Two-stage Detector*” berbasis *Deep Learning* serta YOLO (*You Only Look Once*), SSD (*Single Shot Multibox Detector*), dan RetinaNet merupakan metode deteksi objek “*One-stage Detector*” berbasis *Deep Learning*. Berikut Gambar 2.3 yang memaparkan *road map* perkembangan deteksi objek selama dua dekade terakhir.



Gambar 2.3 Road map perkembangan deteksi objek selama dua dekade terakhir

Sumber: Zou et al. (2019)

### ***You Only Look Once (YOLO)***

*You Only Look Once (YOLO)* adalah salah satu algoritma deteksi objek berbasis *Deep Learning* yang dikembangkan pertama kali oleh Redmon et al. pada tahun 2015. YOLO merupakan algoritma deteksi objek yang berasal dari pengembangan metode *Convolutional Neural Network (CNN)* (Leriansyah, 2020). Algoritma ini merupakan algoritma deteksi objek “*One-stage Detector*” pertama yang berbasis *Deep Learning*.

Sesuai dengan namanya, algoritma *You Only Look Once (YOLO)*, berbeda dengan algoritma-algoritma deteksi objek sebelumnya yang menganut paradigma “deteksi proposal (kandidat *Bounding Boxes*) + verifikasi (klasifikasi dan lokalisasi objek)”, YOLO hanya menggunakan satu lapisan jaringan syaraf (*Neural Network*) pada citra. Jaringan ini akan membagi citra menjadi beberapa *regions* (daerah) dan memprediksi *bounding boxes* dan probabilitas setiap *regions* secara bersamaan (Zou et al., 2019). Walaupun memiliki kecepatan deteksi yang lebih cepat, YOLO membuat lebih banyak kesalahan lokalisasi objek, hal ini disebabkan karena YOLO tidak melakukan tahap deteksi proposal terlebih dahulu. Selain itu, YOLO juga kesusahan dalam mendeteksi objek berukuran kecil dan berdempetan.

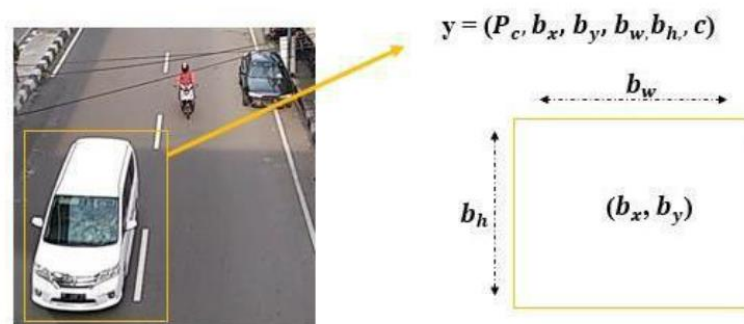
Algoritma YOLO membagi citra masukan menjadi beberapa sel *grid* (kotak kecil) berukuran  $s \times s$ . Jika titik pusat dari sebuah objek dalam citra masuk ke salah satu sel *grid* pada citra, maka sel *grid* tersebut bertanggung jawab untuk mendeteksi objek tersebut.

Setiap sel *grid* bertanggung jawab untuk memprediksi sebanyak B *bounding boxes* dan *confidence scores* setiap *bounding box*. *Confidence scores* berisi nilai seberapa yakin (*confident*) model terhadap kemungkinan keberadaan sebuah objek dalam *bounding box* serta akurasi dari perkiraan *bounding box* itu sendiri. Nilai *confident* sebuah *bounding box* dapat dihitung dengan Persamaan 2.1.

$$\text{Confident} = P_r(\text{Object}) \times IOU_{pred}^{\text{truth}} \quad (2.1)$$

Nilai *confident* dapat juga dihitung dengan nilai *Intersection Over Union* (IOU) antara *Bounding Box* yang diprediksi dengan *ground truth box* (kotak pembatas yang dibuat manual oleh manusia saat proses *labelling* dan berisikan tipe serta lokasi objek yang akurat). Jika tidak ada objek di dalam sel *grid*, maka besar nilai *confidence* adalah nol. Setiap sel *Grid* juga memprediksi probabilitas C kelas kondisional.

Setiap *bounding box* berisikan lima nilai prediksi, yaitu x, y, w, h, dan *confidence*, dimana x dan y adalah koordinat yang merepresentasikan titik pusat *bounding box*, w adalah lebar (*width*) *bounding box*, h adalah tinggi (*height*) *bounding box*, dan *confidence* adalah representasi IOU antara *bounding box* yang diprediksi dengan *ground truth box*. Berikut Gambar 2.4 yang berisi ilustrasi *bounding box* pada sebuah citra.

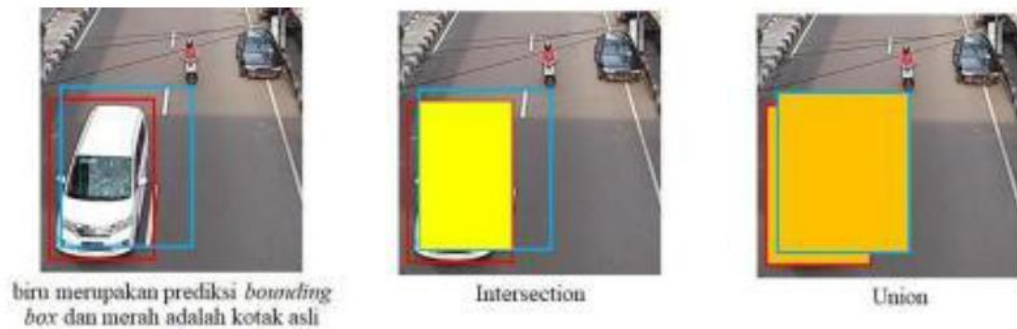


Gambar 2.4 Komponen *bounding box*

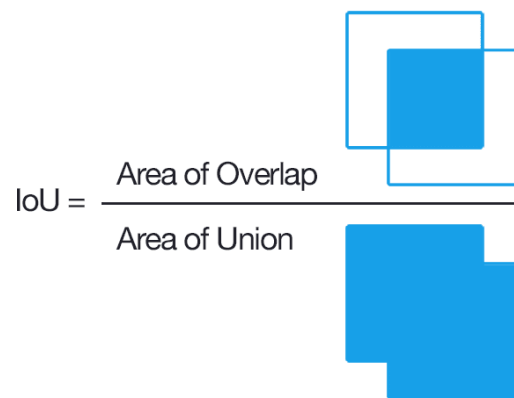
Sumber: Leriensyah (2020)

Nilai IOU adalah nilai perbandingan luas area yang tumpang tindih antara area prediksi *bounding box* dan area *ground truth box* dengan luas area gabungan prediksi *bounding box* dan area *ground truth box* seperti yang diilustrasikan pada Gambar 2.5 dan Gambar 2.6.



Gambar 2.5 Contoh *Intersection Over Union*

Sumber: Leriensyah (2020)



Gambar 2.6 Perhitungan IOU

Sumber: Rosebrock (2016)

Pada Gambar 2.5, objek yang dideteksi adalah mobil, dan terdapat dua kotak pada citra tersebut. Kotak berwarna merah adalah *ground truth box* yang didapatkan melalui proses *labelling* secara manual, yang tentunya akurasi lokalisasinya sangat hampir akurat, sedangkan kotak berwarna biru adalah prediksi *bounding box* yang didapatkan melalui prediksi jaringan YOLO, yang akurasi lokalisasinya tidak sebagus *ground truth box*. Untuk mendapatkan nilai IOU objek mobil, dilakukan perbandingan antara area tumpang tindih prediksi *bounding box* objek mobil dengan *ground truth box* objek mobil dan area gabungan prediksi *bounding box* dengan *ground truth box* objek mobil. Melalui perbandingan ini akan didapatkan nilai IOU yang berkisar antara 0-1, jika nilai IOU semakin mendekati satu maka itu artinya *bounding box* yang diprediksi mempunyai akurasi lokalisasi yang tinggi.

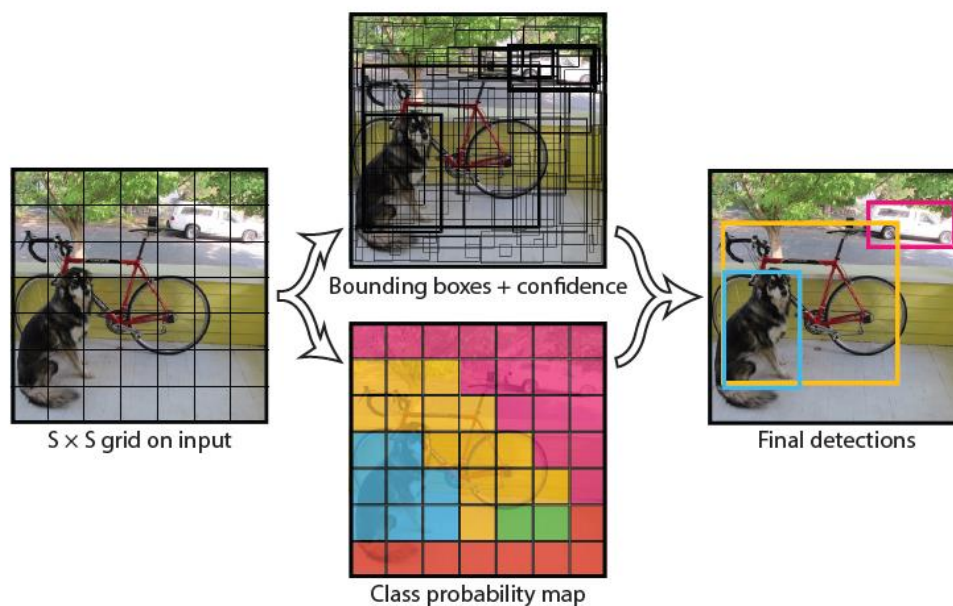
Untuk mendapatkan probabilitas sebuah objek muncul dalam sebuah *bounding box* dan seberapa cocok *bounding box* membatasi sebuah objek, YOLO mengalikan probabilitas kelas kondisional dan prediksi *confidence* sebuah *bounding box* seperti pada Persamaan 2.2.

$$P_r(Class_i|Object) \times P_r(Object) \times IOU_{pred}^{truth} = P_r(Class_i) \times IOU_{pred}^{truth} \quad (2.2)$$

Hasil dari prediksi deteksi objek algoritma YOLO dituliskan sebagai *tensor* dengan ukuran seperti pada Persamaan 2.3.

$$\text{Ukuran Tensor} = S \times S \times (B \times 5 + C) \quad (2.3)$$

Pada Persamaan 2.3, S adalah banyak *grid*, B adalah nilai-nilai komponen *bounding box* yang berjumlah lima buah, dan C adalah nilai *confidence*. Berikut Gambar 2.7 yang memaparkan alur kerja algoritma YOLO.

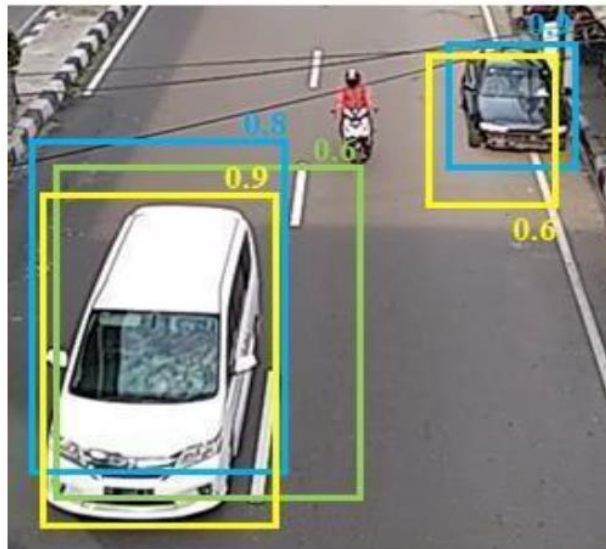


Gambar 2.7 Alur kerja algoritma YOLO dalam pendeteksian objek pada citra

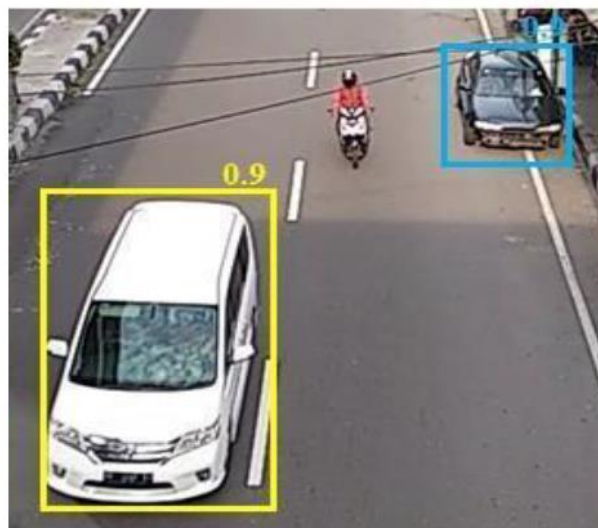
Sumber: Redmon et al. (2016)

Dalam proses deteksi objek, algoritma YOLO akan mendeteksi beberapa objek besar atau objek yang berdekatan dengan batas sel-sel *grid* dalam banyak *bounding box* yang akurasi

lokalisasinya beragam (seperti pada Gambar 2.8), dari yang rendah (nilai IOU mendekati 0) dan tinggi (nilai IOU mendekati 1). Hal ini tentunya tidak diinginkan dan untuk mengatasi itu, dapat dilakukan operasi *Non-maximal Suppression*. *Non-maximal Suppression* (NMS) adalah operasi pembuangan *bounding box* yang memiliki nilai IOU kurang dari *threshold* (nilai ambang) yang ditentukan seperti pada Gambar 2.8 dan Gambar 2.9.



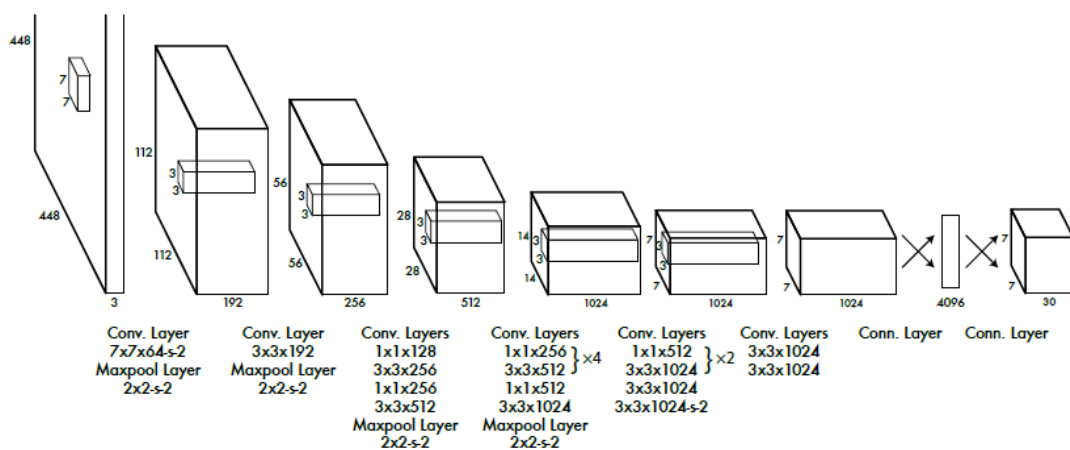
Gambar 2.8 Prediksi Bounding Box yang beragam  
(Leriansyah, 2020)



Gambar 2.9 Pembuangan *bounding box* yang nilainya kurang dari *threshold* yang ditentukan  
(Leriansyah, 2020)

Pada Gambar 2.8, dapat dilihat pada citra tersebut, terdapat beberapa *bounding box* pada objek mobil yang mempunyai nilai IOU yang beragam. Dengan NMS, *bounding box* yang berlebihan ini dapat dibuang. Pada Gambar 2.9, ditentukan *Threshold* sebesar 0,9, dengan begitu *Bounding Box* yang memiliki nilai IOU kurang dari 0,9 akan dibuang.

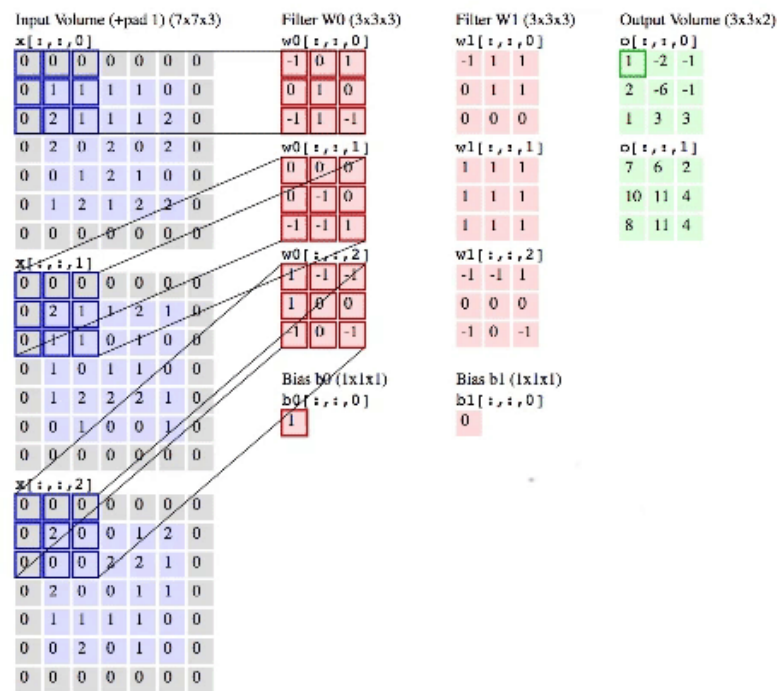
Algoritma YOLO diimplementasikan sebagai jaringan syaraf *Convolutional* atau *Convolutional Neural Network* (CNN) dan dievaluasi menggunakan PASCAL VOC *Detection Dataset*. Lapisan *Convolutional* digunakan untuk mengekstrak fitur dari citra dan lapisan *Fully Connected* digunakan untuk memprediksi probabilitas dan koordinat keluaran. YOLO terdiri dari 24 lapisan *Convolutional* yang diikuti oleh dua lapisan *Fully Connected*. Berikut Gambar 2.10 yang memaparkan arsitektur jaringan YOLO.



Gambar 2.10 Arsitektur YOLO

(Redmond et al., 2016)

Pada lapisan *Convolutional* terjadi operasi konvolusi terhadap setiap masukan jaringan (piksel citra). Konvolusi didefinisikan sebagai proses untuk memperoleh suatu piksel didasarkan pada nilai piksel itu sendiri dan tetangganya dengan melibatkan suatu matriks yang disebut *Kernel* yang merepresentasikan pembobotan (Ilahiyah and Nilogiri, 2018). Berikut Gambar 2.11 yang memvisualisasikan contoh perhitungan konvolusi sebuah citra RGB (matriks tiga dimensi) pada jaringan *Convolutional*.



Gambar 2.11 Contoh operasi konvolusi pada lapisan *Convolutional*

(Lina, 2019)

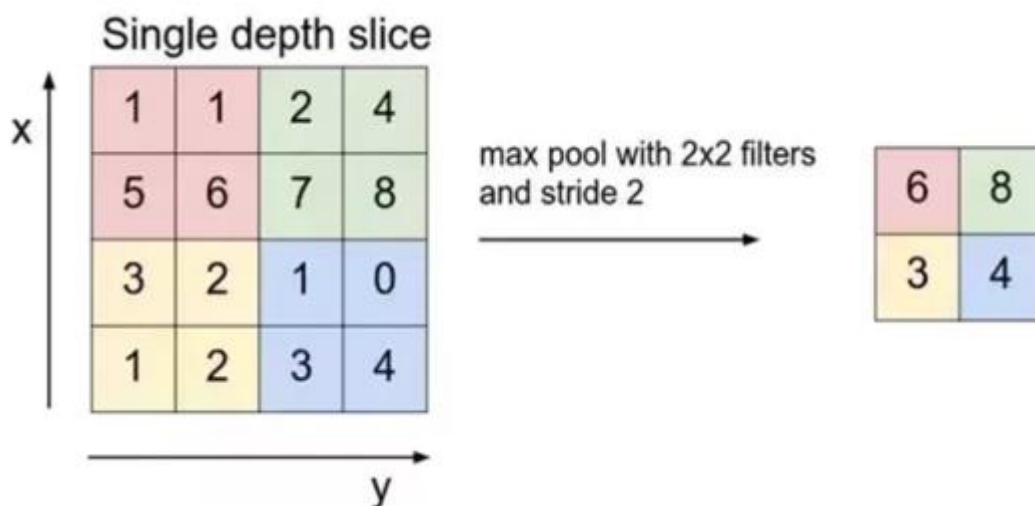
Operasi konvolusi dilakukan dengan melakukan perkalian dot terhadap ketiga dimensi matriks citra masukan tersebut, dimulai dari bagian sudut atas kiri sebesar 3x3. Bagian 3x3 matriks citra ini akan dikalikan dengan filter *kernel* sebesar 3x3x3 juga. Pada operasi konvolusi pertama, dilakukan operasi perkalian dot matriks 3x3 pertama dengan filter *kernel* 3x3 sesuai dengan dimensinya (misalnya matriks citra dimensi pertama dengan filter *kernel* dimensi pertama). Hasil dari operasi perkalian dot matriks citra dengan filter *kernel* akan menghasilkan *feature map* sebesar 3x3. Proses pelatihan ditujukan untuk menemukan bobot terbaik yang merupakan filter *kernel* yang mampu mendeteksi atau mengekstraksi fitur dari citra masukan sebaik mungkin.

*Feature map* yang didapatkan melalui operasi konvolusi akan dimasukkan ke dalam fungsi aktivasi *Leaky ReLu*. Fungsi aktivasi merupakan fungsi yang digunakan pada jaringan untuk mengaktifkan atau tidak mengaktifkan neuron atau *node* (Fandisyah, 2021). Fungsi aktivasi akan membantu jaringan untuk mempelajari pola yang lebih kompleks dari *dataset*. Hasil dari penerapan fungsi aktivasi akan dilanjutkan menuju lapisan selanjutnya. Pada YOLO, digunakan fungsi aktivasi *Leaky ReLu* (*Rectified Linear Unit*). Fungsi aktivasi *Leaky ReLu* adalah pengembangan dari fungsi aktivasi *ReLu* yang dihitung dengan Persamaan 2.4.

$$f(x) = \max(0, x) \quad (2.4)$$

Pada fungsi aktivasi *Leaky ReLu*, jika masukan bernilai di bawah 0 (negatif) maka keluarannya akan bernilai mendekati 0, dan jika masukan bernilai lebih dari 0 maka keluarannya akan bernilai sebesar masukan tersebut. Fungsi aktivasi *Leaky ReLu* digunakan untuk menutupi kekurangan fungsi aktivasi ReLu. Kekurangan dari fungsi aktivasi ReLu disebut *Dying ReLu*. *Dying ReLu* adalah kondisi dimana kebanyakan neuron tidak aktif (karena mempunyai nilai masukan negatif yang menghasilkan keluaran bernilai 0 atau tidak aktif). *Leaky ReLu* mengatasi masalah ini dengan memberikan keluaran yang hampir mendekati 0, namun tidak pernah 0, jika masukannya bernilai negatif.

Terdapat juga lapisan *Pooling* yang berguna untuk memperbesar (*Upsampling*) atau memperkecil (*Downsampling*) dimensi *feature map*. YOLO menggunakan beberapa lapisan *Downsampling* untuk mengambil *feature* yang penting serta memperkecil ukuran *feature map*. Dengan pengurangan jumlah *feature*, risiko terjadinya *Overfitting* (jaringan mempunyai akurasi yang bagus pada data *training*, namun akurasi yang jelek pada data *testing*). Berikut Gambar 2.12 yang mengilustrasikan contoh operasi *Max Pooling* pada matriks berdimensi 4x4 menjadi matriks berdimensi 2x2.



Gambar 2.12 Contoh operasi *Max Pooling*  
(Alderliesten, 2020)

Setelah semua *Hidden Layer* (lapisan konvolusi dan lapisan *Pooling*) telah dilalui oleh *feature map*, *feature map* akan di-*flattened* (diratakan atau digabungkan jadi satu) menjadi satu vektor panjang dan dimasukkan ke dalam lapisan *Fully Connected*. Pada lapisan *Fully Connected*, vektor yang berisikan seluruh *feature map* dari lapisan-lapisan sebelumnya akan digunakan sebagai masukan fungsi aktivasi Linear yang akan menghasilkan prediksi objek deteksi beserta *bounding box*-nya.

Proses pelatihan jaringan YOLO dilakukan menggunakan *framework* Darknet dengan waktu 1 minggu dan didapatkan akurasi sebesar 88% pada ImageNet 2012 *Validation Set*. Dilakukan proses *pretraining* dengan menggunakan 20 lapisan *Convolutional* pertama. Proses *pretraining* ini dilakukan dengan ImageNet 1000-*class Competition Dataset*.

Jiang et al. (2022) memaparkan bahwa selama dua dekade terakhir, YOLO mengalami beberapa perkembangan, di antaranya adalah:

- a. YOLO v2 pada tahun 2016 yang menambahkan *anchor* dengan K-means, *Two-stage Training*, serta *Full Convolutional Network*.
- b. YOLO v3 pada tahun 2018 yang menambahkan deteksi *Multi-scale* dengan menggunakan FPN.
- c. YOLO v4 pada tahun 2019 yang menambahkan SPP, fungsi aktivasi MISH, *Data Enhancement Mosaic/Mixup*, GIOU (*Generalized Intersection over Union*) *Loss Function*.
- d. YOLO v5 pada tahun 2020 yang menambahkan kontrol yang lebih fleksibel terhadap ukuran model, pengaplikasian fungsi aktivasi Hardswish, dan *Data Enhancement*.

Selain beberapa versi YOLO di atas terdapat juga beberapa versi YOLO lain yang memiliki kelebihan serta kekurangannya masing-masing seperti YOLO 9000, Tiny YOLO, Fast YOLO, dan YOLACT (*You Only Look at Coefficients*).

Terhitung dari YOLO v4, Joseph Redmond, penggagas pertama algoritma YOLO, tidak lagi mempunyai kontribusi langsung terhadap perkembangan YOLO v4 dan algoritma-algoritma YOLO selanjutnya dikarenakan Redmond memilih untuk hengkang dari komunitas AI.

### ***You Only Look Once (YOLO) v3***

YOLO v3 merupakan salah satu pengembangan algoritma YOLO yang menggunakan Darknet-53 sebagai *feature extractor*-nya. Darknet-53 memiliki 53 lapisan konvolusi, berbeda dengan Darknet-19 yang digunakan oleh YOLO v2, yang hanya memiliki 19 lapisan konvolusi. Berikut Gambar 2.13 yang memaparkan arsitektur Darknet-53.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1x	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2x	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8x	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8x	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4x	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Gambar 2.13 Arsitektur Darknet-53

(Redmond et al., 2016)

Selain 53 lapisan Darknet-53 yang digunakan untuk mengekstraksi fitur dari citra masukan, terdapat 53 lapisan lain untuk melakukan proses deteksi, yang menjadikan total lapisan pada YOLO v3 menjadi 106 lapisan.

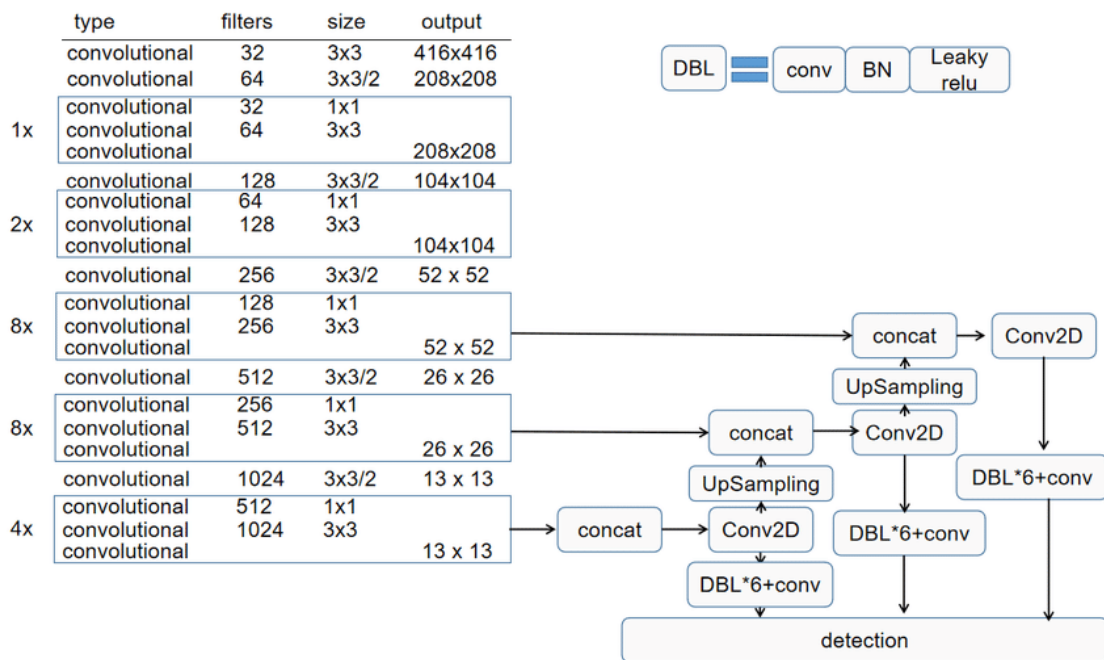
Pada YOLO v3 proses deteksi objek dilakukan pada tiga skala ukuran dan lapisan yang berbeda. Proses deteksi objek dilakukan pada lapisan ke-82, 94, dan 106, serta skala ukuran *stride* 32, 16, dan 8. *Stride* adalah jarak pergeseran filter terhadap matriks citra masukan ketika melakukan proses konvolusi (Hamman, 2020). *Stride* di sini digunakan untuk melakukan *Downsamples* (proses untuk mengurangi tinggi dan lebar citra, hal ini ditujukan untuk meringankan proses deteksi) pada citra masukan, misalnya citra masukan berukuran  $416 \times 416$ , maka ukuran dari citra masukan akan menjadi  $13 \times 13$ ,  $26 \times 26$ , dan  $52 \times 52$  (didapatkan dari proses pembagian antara ukuran citra masukan dengan *stride*), selanjutnya proses deteksi akan dilakukan pada masing-masing skala ukuran dan lapisan seperti berikut:

- a. Proses deteksi untuk objek ukuran besar dilakukan pada citra keluaran  $13 \times 13$  pada lapisan ke-82.



- b. Proses deteksi untuk objek ukuran sedang dilakukan pada citra keluaran  $26 \times 26$  pada lapisan ke-94.
- c. Proses deteksi untuk objek ukuran kecil dilakukan pada citra keluaran  $52 \times 52$  pada lapisan ke-106.

Untuk lebih jelasnya, dapat dilihat arsitektur dari jaringan YOLO v3 pada Gambar 2.14.



Gambar 2.14 Arsitektur jaringan YOLO v3

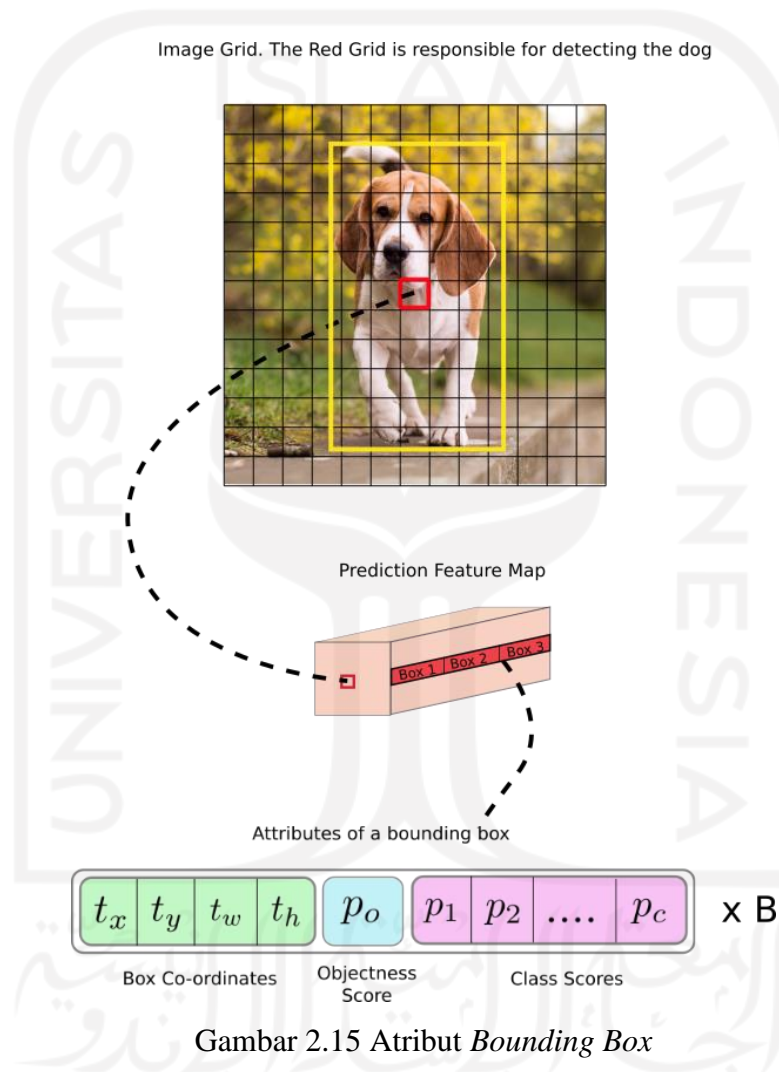
(Ding et al., 2019)

Masukan dari jaringan YOLO v3 adalah sebuah *Batch* citra, yang setiap citranya harus memiliki ukuran resolusi yang dapat dibagi oleh 32, karena YOLO v3 melakukan proses deteksi pada *stride* 32, 16, dan 8.

Untuk melakukan deteksi YOLO v3 menerapkan filter *kernel*  $1 \times 1$  pada citra yang telah di-*downsampled* oleh *stride* 32, 16, dan 8 di tiga lapisan yang telah disebutkan (lapisan ke-82, 94, dan 106). Filter *kernel*  $1 \times 1$  ini mempunyai besar *depth* sebesar  $b \times (5 + c)$ ,  $b$  adalah total *bounding box* yang diprediksi oleh setiap sel *grid*,  $(5 + c)$  adalah total atribut yang dimiliki oleh setiap *bounding box*, atribut-atribut itu adalah titik pusat koordinat  $x$  dan  $y$ , tinggi, lebar serta *objectness score* (nilai yang menyatakan seberapa yakin sebuah *bounding box* mengandung sebuah objek) *bounding box* (lima atribut) serta banyak kelas objek (atribut  $c$ ). Karena YOLO v3 memprediksi tiga *bounding box* pada setiap sel *grid* citra, maka atribut  $b$  akan selalu bernilai tiga. Misalnya, terdapat 80 kelas objek deteksi, maka besar *depth* filter

*kernel* deteksi adalah  $3 \times (5 + 80) = 255$ , sehingga didapatkan filter *kernel* berukuran  $1 \times 1 \times 255$  yang apabila diaplikasikan ke citra masukan berukuran  $416 \times 416$  pada tiga skala ukuran yang berbeda (*stride* 32,16, dan 8) akan menghasilkan tiga *feature maps* yang masing-masing berukuran  $13 \times 13 \times 255$ ,  $26 \times 26 \times 255$ , dan  $52 \times 52 \times 255$ .

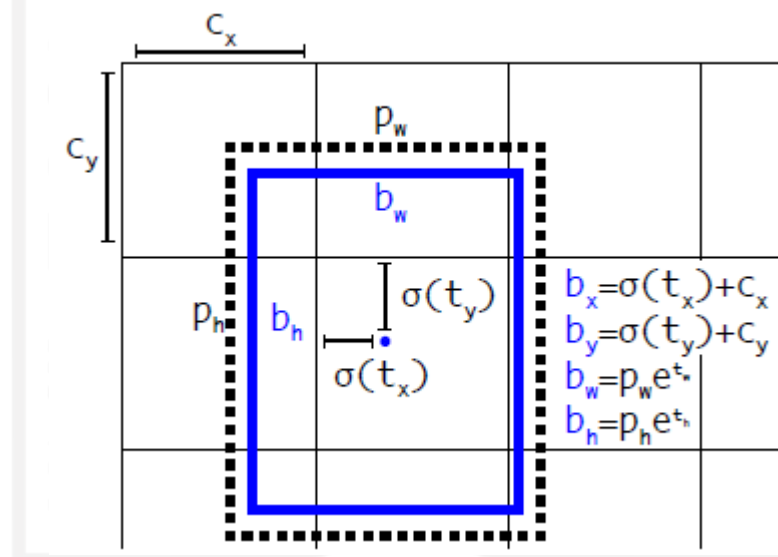
Selanjutnya, untuk memahami atribut *bounding box* pada jaringan YOLO v3, dapat diperhatikan Gambar 2.15.



Pada Gambar 2.15, sel *grid* yang berwarna merah adalah sel *grid* yang bertanggung jawab untuk melakukan deteksi kelas objek pada *ground truth box*. Sel *grid* tersebut mengandung tiga *bounding box* yang masing-masing mempunyai atribut koordinat titik pusat  $x$  dan  $y$ , tinggi, lebar, *objectness score*, serta *confidences score* (nilai yang menyatakan seberapa yakin sebuah *bounding box* mengandung sebuah objek kelas tertentu) kelas objek. Sel

*grid* lainnya yang berada di dua masukan citra dengan *stride* berbeda juga mengandung atribut-atribut tersebut.

Untuk proses deteksi *Bounding Box* objek deteksi, YOLO v3 menggunakan *anchors* atau *priors* (*Pre-defined default bounding boxes*). *anchors* akan digunakan untuk menghitung lebar serta tinggi *bounding box* objek deteksi. Pada YOLO v3, terdapat sembilan *anchor boxes* yang akan dibagi masing-masing tiga *anchor boxes* pada tiap skala deteksi. Untuk menghitung kesembilan *anchor boxes* ini, digunakan *K-Means Clustering*. Pada YOLO v3, *Default Anchor Boxes* didapatkan melalui *K-Means Clustering* pada *Dataset VOC*. Transformasi *Log-space* digunakan untuk menghitung lebar dan tinggi sebuah *bounding box* objek deteksi, dan fungsi Sigmoid akan digunakan untuk memprediksi titik pusat *bounding box* objek deteksi seperti pada Gambar 2.16.



Gambar 2.16 *Anchors* dan *bounding box* objek deteksi  
(Redmon & Farhadi, 2018)

Komponen  $b_x$  dan  $b_y$  adalah titik pusat *bounding box* objek deteksi,  $b_w$  dan  $b_h$  adalah lebar serta tinggi *bounding box* objek deteksi,  $t_x$ ,  $t_y$ ,  $t_w$ , dan  $t_h$  adalah keluaran jaringan setelah proses pelatihan,  $c_x$  dan  $c_y$  adalah koordinat ujung kiri atas sebuah sel *grid*, serta  $p_w$  dan  $p_h$  adalah lebar dan tinggi *anchor box*.

### Training (Pelatihan) Jaringan

*Training* atau pelatihan jaringan adalah proses pembelajaran jaringan agar jaringan mampu melakukan tugas yang diperintahkan. Pada proses pelatihan, jaringan akan berusaha

untuk menemukan *weight* (bobot) terbaik, yang mampu memberikan performa pekerjaan yang memuaskan. Proses pencarian *weight* terbaik akan dilakukan melalui perhitungan-perhitungan yang menggunakan masukan *dataset* dan beberapa parameter lain yang biasa disebut *hyperparameter*. Waktu yang dibutuhkan jaringan untuk melakukan proses pelatihan tergantung dari seberapa besar dan kompleks *dataset* dan *hyperparameter* yang digunakan. Proses pelatihan jaringan mempunyai dua proses penting yaitu *Pass Forward*, yaitu proses perhitungan nilai keluaran jaringan berdasarkan bobot dan *dataset* yang ada, serta *Backpropagation*, yaitu proses perbaikan nilai bobot berdasarkan nilai keluaran jaringan yang didapat dari proses *Pass Forward*.

Pada proses pelatihan, *dataset* akan dibagi menjadi masukan jaringan (*input / predictor / x*) dan keluaran jaringan (*output / prediction / y*). Proses *Pass Forward* akan menggunakan data masukan untuk menghitung keluaran yang nantinya akan dibandingkan dengan data keluaran dari *dataset*. Perbandingan antara keluaran yang dihasilkan proses *Pass Forward* dan data keluaran dari *dataset* ini akan dijadikan patokan kesalahan jaringan dalam melakukan tugasnya. Selanjutnya pada proses *backpropagation*, jaringan akan memperbaiki bobotnya sesuai dengan kesalahan yang dilakukan jaringan.

Pada YOLO v3 terdapat beberapa *hyperparameter* yang harus diperhatikan, antara lain:

- a. *Image size* (ukuran citra masukan dalam bentuk *lebar × tinggi*).
- b. *Batch size* adalah total citra yang digunakan untuk memperbarui bobot pada satu iterasi pelatihan.
- c. *Subdivisions* digunakan untuk memproses sebagian kecil *batch* citra menggunakan GPU.
- d. *Max batches* adalah jumlah maksimal iterasi pelatihan, *max batches* bernilai *jumlah kelas × 2000*.
- e. *Steps* digunakan untuk menyesuaikan tingkat rata-rata *loss*, *steps* diatur pada 80% dan 90% jumlah maksimal iterasi *max batches*.
- f. *Learning rate* digunakan untuk menentukan seberapa cepat jaringan dapat memperbarui bobotnya. Nilai *learning rate* yang kecil akan membuat proses pelatihan menjadi lebih lambat namun meningkatkan kemungkinan mendapatkan nilai bobot yang lebih baik, sebaliknya nilai *learning rate* yang besar akan membuat proses pelatihan lebih cepat namun mengurangi kemungkinan mendapatkan nilai bobot yang lebih baik.
- g. *Epoch* adalah total terjadinya rangkaian proses *Forward Pass* hingga *Backpropagation*.

Untuk mengatasi masalah lamanya waktu yang dibutuhkan untuk melakukan proses pelatihan, terdapat jenis proses pelatihan yang lebih cepat yang menggunakan bobot yang

sudah dilatih oleh jaringan lain. Proses ini disebut sebagai proses *Transfer Learning*. *Transfer Learning* adalah proses menyalin pengetahuan dari jaringan yang sudah ada (bobot pra latih atau *pretrained weight*) ke jaringan yang baru untuk menyelesaikan masalah serupa (Fandisyah, 2021). Rajagede (2021), menjelaskan bahwa terdapat dua cara untuk menggunakan proses *Transfer Learning* yaitu:

- a. *Fixed Feature Extractor* yang melatih jaringan menggunakan *pretrained weight* tanpa melatih ulang *weight* tersebut. Cara ini akan membekukan (*freeze*) sebuah lapisan dalam jaringan, yang berarti jika digunakan jaringan YOLO v3 maka jaringan yang mungkin untuk dibekukan antara lain adalah jaringan konvolusi (sebagai *feature extractor* yang akan menghasilkan fitur atau ciri) atau klasifikasi.
- b. *Fine Tuning* yang melatih jaringan menggunakan *pretrained weight* dengan melatih ulang *weight* tersebut. Cara ini tidak akan membekukan lapisan dalam jaringan.

### Pengujian Performa Jaringan

Terdapat beberapa pengujian performa yang dapat menguji keberhasilan sebuah jaringan di antaranya adalah *Confusion Matrix*. *Confusion Matrix* adalah ringkasan hasil prediksi pada permasalahan klasifikasi (Amwin, 2021). *Confusion Matrix* membandingkan total klasifikasi benar sebuah kejadian terjadi dengan kejadian yang benar terjadi (*True Positive*) dan yang tidak terjadi (*True Negative*) serta total klasifikasi salah sebuah kejadian terjadi dengan kejadian yang benar terjadi (*False Positive*) dan yang tidak terjadi (*False Negative*) seperti pada Gambar 2.17.

		ACTUAL	
		Class 1 (Positive)	Class 2 (Negative)
PREDICTION	Class 1 (Positive)	TP	FP
	Class 2 (Negative)	FN	TN

Gambar 2.17 *Confusion Matrix*

Sumber: Koech (2020)

Nilai *True Negative* tidak dapat dicari karena nilai ini didapatkan dari total klasifikasi benar kejadian yang tidak terjadi sedangkan terdapat banyak kemungkinan klasifikasi kejadian

yang seharusnya tidak terdeteksi dalam sebuah gambar (Koech, 2020). Dari perhitungan *Confusion Matrix* dapat dihitung nilai *Precision*, *Recall*, dan *F1 Score* dengan menggunakan Persamaan 2.5 sampai 2.7.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{Semua\ Prediksi} \quad (2.5)$$

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{Semua\ Ground\ Truth} \quad (2.6)$$

$$F1\ Score = 2 \times \left( \frac{(Precision \times Recall)}{(Precision + Recall)} \right) \quad (2.7)$$

*Precision* merepresentasikan kemampuan model untuk mengidentifikasi objek terkait sebagai nilai persentase prediksi yang benar, dan *Recall* merepresentasikan kemampuan model untuk menemukan semua objek relevan sebagai nilai persentase positif benar yang dapat terdeteksi di semua *Ground Truth* (Fandisyah, 2021). Sedangkan *F1 Score* adalah nilai rata-rata *Harmonic* dari *Precision* dan *Recall*.

## 2.2 Kajian Pustaka Penelitian Terdahulu

Selama beberapa tahun terakhir, penelitian mengenai deteksi kendaraan telah beberapa kali dilakukan. Beberapa penelitian tersebut menggunakan beragam metode dan alat untuk melakukan proses deteksi kendaraan.

Metode pertama adalah penggunaan sensor untuk mendeteksi kendaraan. Sensor yang digunakan dapat berupa sensor *infra red* (IR) dan sensor *Radio Frequency Identification* (RFID). Penggunaan sensor *infra red* dinilai mampu mendeteksi kendaraan dengan baik, seperti pada penelitian Oudat, Mousa, & Claudel (2015) yang mengembangkan sebuah perangkat deteksi kendaraan dengan akurasi 99%. Walaupun memiliki akurasi deteksi 99%, penggunaan sensor masih memiliki kekurangan, di antaranya adalah dibutuhkannya perawatan dan pemasangan yang memakan banyak sumber daya (Assidhiqi, 2021).

Metode kedua yang digunakan pada tema penelitian ini adalah metode *Image Processing*. *Image Processing* adalah salah satu teknologi pengolahan citra secara digital. Pada metode ini, citra akan diolah sedemikian rupa agar mampu memudahkan proses ekstraksi fitur. Penelitian Jain, Verma, & Jain (2018) yang berjudul “*Automatic Traffic Light Control System by Using*

*Digital Image Processing*” menggunakan *Edge Detection* untuk mendeteksi kendaraan. *Edge Detection* digunakan untuk mengidentifikasi titik pada citra digital yang sudah diubah kecerahannya secara tajam, titik-titik ini akan membentuk suatu segmen garis yang disebut tepi atau *Edges* (Jain, Verma, & Jain, 2018). Penelitian ini juga menggunakan juga menggunakan metode *Image Processing* lainnya yaitu *RGB to Gray Conversion* (konversi warna citra RGB menjadi abu-abu, ditujukan untuk menyederhanakan fitur dari citra), *Image Enhancement* (mengatur agar citra yang digunakan lebih cocok untuk dianalisis), serta *Image Matching* (Pencocokan citra yang sudah dideteksi kendaraannya dengan citra yang mengatur lama lampu lalu lintas).

Metode ketiga adalah *Machine Learning*. *Machine Learning* merupakan sebuah proses yang dilakukan oleh komputer atau mesin untuk mempelajari sesuatu yang dapat dilakukan manusia dengan semirip mungkin (Assidhiqi, 2021). Penelitian Velazquez et al. (2018) menggunakan algoritma OC-SVM (*One Class Support Vector Machine*) untuk membangun sebuah sistem deteksi kendaraan menggunakan satu kamera statik. Sistem akan mensegmentasi objek bergerak dari latar belakang menggunakan *Gaussian Mixture Model* adaptif, selanjutnya fitur-fitur geometris akan diekstraksi seperti luas, tinggi, lebar, titik pusat, serta *Bounding Box*. Fitur geometris luas, tinggi, dan lebar akan digunakan untuk melakukan klasifikasi kendaraan ke dalam tiga kelas keluaran yaitu kecil, menengah, dan besar. Sistem ini mempunyai nilai *Recall*, *Precision*, serta *F-measure* yang mencapai 98.190%.

Metode keempat adalah *Deep Learning*. *Deep Learning* merupakan cabang ilmu dari *Machine Learning* berbasis jaringan saraf tiruan untuk implementasi suatu permasalahan tertentu (Assidhiqi, 2021). Penelitian Irfan, Ardi, & Candradewi (2017) yang berjudul "Sistem Klasifikasi Kendaraan Berbasis Pengolahan Citra Digital dengan Metode *Multilayer Perceptron*” menggunakan *Haar Cascade Classifier* untuk mendeteksi apakah suatu objek benar merupakan kendaraan atau tidak dan *Multilayer Perceptron* untuk mengklasifikasi kendaraan ke dalam tiga kelas keluaran yaitu mobil, bus, dan truk. Sistem ini mampu mencapai nilai akurasi deteksi sebesar 92,67% serta nilai akurasi klasifikasi rata-rata sebesar 87,60%.

Terdapat juga penelitian deteksi kendaraan yang menggunakan algoritma *Faster R-CNN* (*Region-based Convolutional Neural Network*) seperti pada penelitian Ma'ali (2019). *Faster R-CNN* merupakan pengembangan dari algoritma R-CNN yang menggunakan RPN (*Region Proposal Network*) (Ma'ali, 2019). Jaringan *Faster R-CNN* dilatih menggunakan *dataset* yang berisikan 153 citra kendaraan mainan dengan kelas keluaran yaitu mobil, motor, bus, truk, sepeda, dan manusia. Jaringan yang dilatih memiliki akurasi klasifikasi sebesar 97,027%.

Selanjutnya terdapat algoritma YOLO (*You Only Look Once*) yang menerapkan satu jaringan utuh ke dalam citra secara langsung. Pada penelitian Leriensyah (2020), dilatih jaringan YOLO v3 untuk mendeteksi serta mengklasifikasi kendaraan. Jaringan dilatih menggunakan teknik *Transfer Learning* dengan *Dataset* yang berisi citra lalu lintas pada jalan Laksda Adisucipto, Yogyakarta serta bobot *pretrained* “darknet53.conv.74”. Terdapat empat kelas keluaran pada jaringan ini yaitu motor, mobil, bus, dan truk. Didapatkan akurasi klasifikasi jaringan sebesar 97,6% untuk kondisi lalu lintas sepi, 96,5% untuk kondisi lalu lintas normal, dan 91,6% untuk kondisi lalu lintas padat. Penurunan nilai akurasi klasifikasi pada setiap kondisi lalu lintas disebabkan oleh semakin banyak kendaraan yang berdempetan atau tumpang tindih pada kondisi lalu lintas yang lebih padat.

Pada penelitian Amwin (2021), dilatih jaringan YOLO v3 menggunakan teknik *Transfer Learning* dengan *dataset* yang berisikan 531 citra lalu lintas dan lima kelas keluaran yaitu mobil, sepeda motor, truk, bus, dan becak. Jaringan yang dilatih mampu mengenali kendaraan pada video *closed circuit television* (CCTV) yang dipasang di Simpang Air Mancur-Immanuel Kota Medan dengan nilai *Mean Average Precision* (mAP) sebesar 99,35%.

Pada penelitian Kim (2020), digunakan jaringan YOLO v2 yang telah dilatih menggunakan 190 citra yang didapatkan dari Seoul’s *Intelligent Transportation System* (TOPIS) dengan tiga kelas keluaran yaitu bagian depan mobil, bagian belakang mobil, dan bagian samping mobil. Jaringan diuji melalui lingkungan simulasi perempatan mobil mainan dan didapatkan nilai mAP sebesar 90%.

Berdasarkan kajian tersebut, peneliti tertarik untuk melakukan penelitian mengenai klasifikasi kendaraan menggunakan algoritma YOLO v3 dengan dua skenario kelas keluaran berdasarkan sudut pengambilan citra. Berikut Tabel 2.1 yang merangkum semua hasil kajian pustaka dari penelitian terdahulu.

Tabel 2.1 Tabel rangkuman kajian pustaka dari penelitian terdahulu

Judul Penelitian, Peneliti	Metode	<i>Dataset</i>	Kelas Keluaran	Hasil
<i>Vehicle Detection and Classification Using Passive Infrared Sensing</i> , (Oudat,	Sensor <i>Infra Red</i>	-	-	Didapatkan nilai akurasi deteksi sebesar 99%.



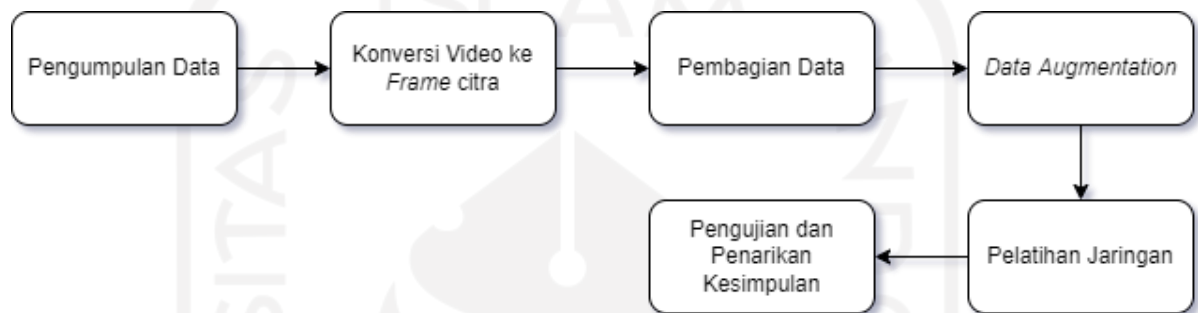
Mousa, & Claudel, 2015)				
<i>Automatic Traffic Light Control System By Using Digital Image Processing</i> , (Jain, Verma, & Jain, 2018)	<i>Edge Detection</i>	-	-	-
<i>Vehicle Detection with Occlusion Handling, Tracking, and OC-SVM Classification: A High Performance Vision-Based System</i> , (Velazquez et al., 2018)	<i>One Class - Support Vector Machine</i>	Berisikan nilai fitur luas, tinggi, dan lebar kendaraan.	Kendaraan kecil, sedang, dan besar.	Didapatkan nilai <i>Recall</i> , <i>Precision</i> , serta <i>F-measure</i> yang mencapai 98.190%.
Sistem Klasifikasi Kendaraan Berbasis Pengolahan Citra Digital dengan Metode <i>Multilayer Perceptron</i> , (Irfan, Ardi, & Candradewi, 2017)	<i>Multilayer Perceptron</i>	-	Mobil, bus, dan truk.	Didapatkan nilai akurasi deteksi sebesar 92,67% serta nilai akurasi klasifikasi rata-rata sebesar 87,60%
Rancang Bangun Sistem Pengendali Lampu Lalu Lintas Berdasarkan Pengenalan Citra Digital Kendaraan Menggunakan Metode	<i>Faster R-CNN</i>	Berisi 153 citra kendaraan mainan	Mobil, motor, bus, truk, sepeda, dan manusia.	Didapatkan nilai akurasi klasifikasi sebesar 97,027%.

<i>Faster R-CNN</i> , (Ma'ali, 2019)				
Deteksi dan Perhitungan Kendaraan untuk Mengetahui Arus Kepadatan Lalu Lintas Secara Otomatis Menggunakan YOLO v3, (Leriansyah, 2020)	YOLO v3	Citra lalu lintas pada jalan Laksda Adisucipto, Yogyakarta	Motor, mobil, bus, dan truk.	Didapatkan nilai akurasi klasifikasi jaringan sebesar 97,6% untuk kondisi lalu lintas sepi, 96,5% untuk kondisi lalu lintas normal, dan 91,6% untuk kondisi lalu lintas padat.
Deteksi dan Klasifikasi Kendaraan Berbasis Algoritma You Only Look Once (YOLO), (Amwin, 2021)	YOLO v3	Berisikan 531 citra lalu lintas.	Mobil, sepeda motor, truk, bus, dan becak.	Didapatkan nilai mAP sebesar 99,35%.
<i>Situation-cognitive traffic light control based on object detection using YOLO algorithm</i> , (Kim, 2020)	YOLO v2	190 citra yang didapatkan dari Seoul's <i>Intelligent Transportation System (TOPIS)</i>	Bagian depan mobil, bagian belakang mobil, dan bagian samping mobil.	Didapatkan nilai mAP sebesar 90%.

## BAB III METODOLOGI PENELITIAN

### 3.1 Tahapan Penelitian

Penelitian akan dilakukan melalui beberapa tahap yaitu perumusan masalah, penentuan tujuan, landasan teori, kajian pustaka, pengumpulan data, desain model, pembangunan model, pengujian model, dan penarikan kesimpulan seperti pada Gambar 3.1.



Gambar 3.1 Tahapan penelitian

### 3.2 Uraian Penelitian

#### 3.2.1 Perumusan Masalah

Pada tahap perumusan masalah, masalah-masalah yang ingin diangkat menjadi tema penelitian akan diidentifikasi terlebih dahulu. Dalam penelitian ini, tema yang diangkat adalah pendeteksian kendaraan, lantas masalah-masalah yang dapat diangkat adalah bagaimana caranya deteksi kendaraan secara efektif, bagaimana caranya membangun model yang mampu mendeteksi kendaraan secara efektif, dan seberapa besar tingkat akurasi deteksi jaringan. Pada tahap ini juga akan ditentukan tujuan dari penelitian yaitu pembangunan model deteksi kendaraan serta mengetahui akurasi dari jaringan deteksi yang dibangun.

#### 3.2.2 Landasan Teori

Pada tahap landasan teori, segala teori yang berkaitan mengenai tema penelitian akan dipelajari dan digunakan untuk membangun jaringan deteksi kendaraan. Selanjutnya akan dilakukan juga kajian pustaka penelitian-penelitian terdahulu yang berkaitan dengan penelitian yang sedang dilakukan untuk membangun jaringan deteksi kendaraan.

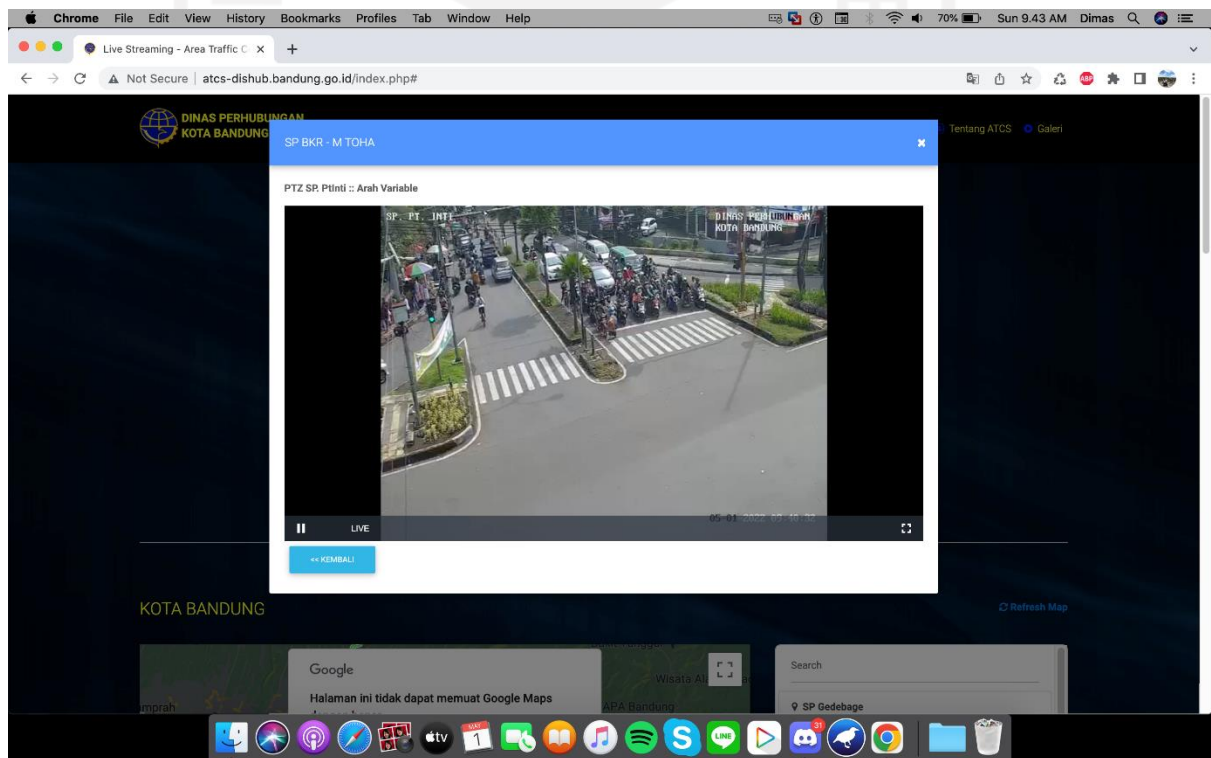
Untuk mempelajari serta mengkaji teori yang berkaitan dengan tema penelitian, penulis mengumpulkan berbagai pustaka dari berbagai macam sumber yaitu tulisan di berbagai situs

yang berhubungan dengan pembelajaran teori yang berkaitan (seperti medium.com dan sejenisnya), video Youtube yang membahas mengenai teori yang berkaitan, jurnal, dan laporan skripsi. Jurnal dan laporan skripsi didapatkan melalui situs Google Scholar (scholar.google.com), DSpace UII (dspace.uui.ac.id), dan Research Gate (researchgate.net).

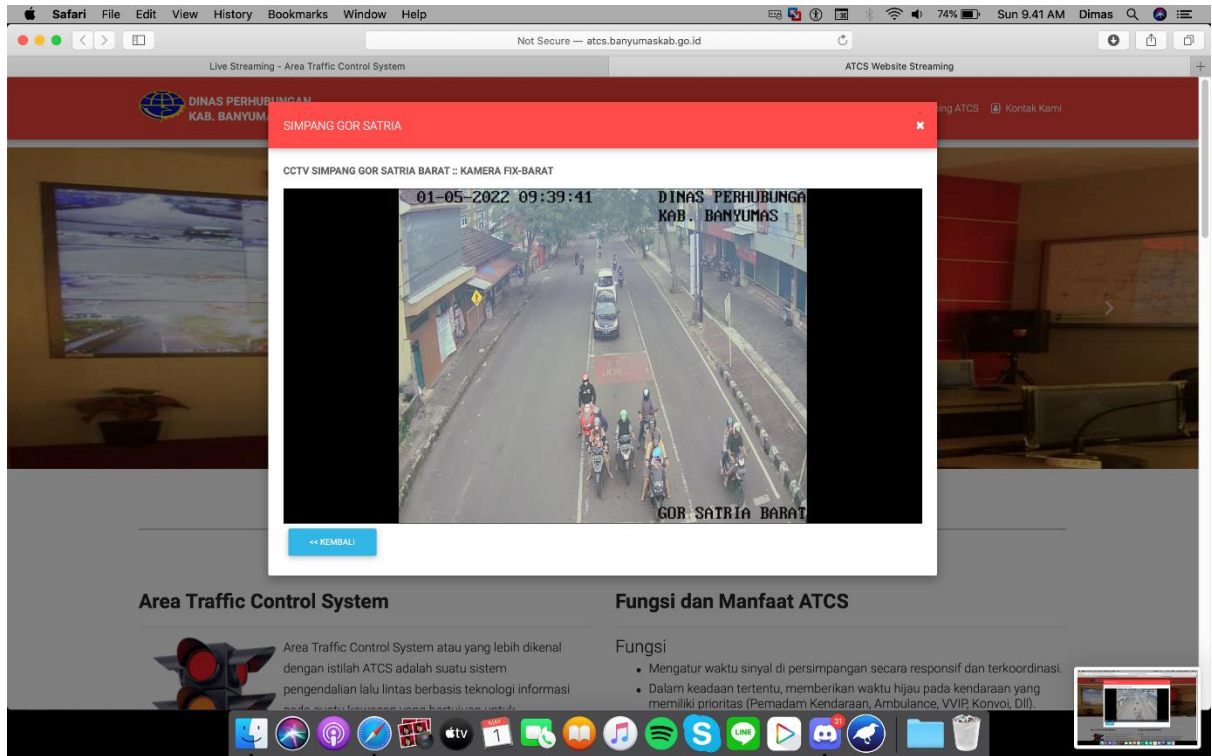
### 3.2.3 Pengumpulan Data

Pada tahap pengumpulan data, data yang dibutuhkan untuk membangun (melatih) jaringan akan dikumpulkan serta disiapkan untuk proses pelatihan jaringan (pelabelan objek kendaraan dan membuang data yang kurang jelas).

Data didapatkan dari rekaman *closed circuit television* (CCTV) lalu lintas pada beberapa jalan yang terekam di situs *Area Traffic Control System* (ATCS) Dinas Perhubungan Kota Bandung (<http://atcs-dishub.bandung.go.id/index.php#>) dan Banyumas (<http://atcs.banyumaskab.go.id/#>) seperti pada Gambar 3.2 dan Gambar 3.3. Digunakan aplikasi Xbox Game Bar untuk merekam rekaman dari situs ATCS Bandung dan Banyumas. Selanjutnya data rekaman lalu lintas akan disimpan ke dalam penyimpanan lokal dalam format .mp4.



Gambar 3.2 Situs ATCS Kota Bandung



Gambar 3.3 Situs ATCS Banyumas

Data rekaman lalu lintas tersebut akan dikonversikan menjadi format “.jpg” untuk dijadikan sebagai data pelatihan jaringan YOLO v3. Proses konversi ini akan dilakukan melalui kode program yang ditulis menggunakan bahasa pemrograman Python 3.7.7 dan IDE (*Integrated Development Environment*) Visual Studio Code. Gambar 3.4 merupakan salah satu contoh satu *frame* rekaman lalu lintas dalam format “.jpg”.



Gambar 3.4 Contoh satu *frame* citra dalam format “.jpg”

Setelah dikonversi ke dalam format “.jpg”, data akan dibagi menjadi *train set* dan *test set* dengan rasio pembagian 70% *train set* dan 30% *test set*. Setelah dilakukan proses pembagian data, *train set* dan *test set* akan diberi label menggunakan aplikasi labelimg (Tzutalin, 2015). Pada proses pelabelan, setiap objek kendaraan pada *frame* citra akan diberi label secara manual berdasarkan kelas keluaran yang telah ditentukan dengan tipe pelabelan untuk jaringan YOLO. Proses pelabelan melalui aplikasi labelimg akan menghasilkan dua keluaran yaitu *file .txt* yang berisikan semua kelas keluaran (“classes.txt”) dan *file .txt* yang berisikan nilai ID objek kelas keluaran, koordinat x dan y objek kelas keluaran, serta lebar dan tinggi objek kelas keluaran dari semua objek kendaraan yang diberi label. Pada penelitian ini, akan dilakukan dua skenario pelabelan yaitu:

- a. Skenario satu, yang tidak mempedulikan sudut pengambilan citra dan terdiri dari tiga kelas yaitu mobil, sepeda motor, dan kendaraan besar (seperti bis dan truk).
- b. Skenario dua, yang mempedulikan sudut pengambilan citra dan terdiri dari sebelas kelas yaitu mobil dari kanan, mobil dari kiri, mobil dari depan, mobil terlihat separuh, sepeda motor dari kanan, sepeda motor dari depan, sepeda motor terlihat separuh, kendaraan besar dari kanan, kendaraan besar dari kiri, kendaraan besar dari depan, dan bagian kendaraan besar terlihat separuh.

Setelah dilakukan proses pelabelan, akan diterapkan beberapa teknik *augmentation* pada *train set* yang telah dilabeli. Hal ini ditujukan untuk menambah variasi data yang akan digunakan untuk melatih jaringan deteksi kendaraan YOLO v3. Diharapkan dengan bertambahnya variasi data yang digunakan, akan meningkatkan performa deteksi kendaraan jaringan yang akan dilatih. Proses penerapan *augmentation* ini akan dilakukan melalui *platform* Google Collab dan dengan bantuan *library* “CLoDSA” (Joheras, 2019). *Library* ini berisikan berbagai macam fungsi untuk menerapkan teknik *augmentation* pada citra. Adapun teknik *augmentation* yang digunakan antara lain:

- a. *Vertical flip*, teknik *augmentation* yang membalik citra secara vertikal.
- b. *Horizontal flip*, teknik *augmentation* yang membalik citra secara horizontal.
- c. *Horizontal-vertical flip*, teknik *augmentation* yang membalik citra secara horizontal dan vertikal.
- d. *Rotation*, teknik *augmentation* yang memutar citra.
- e. *Average blurring*, teknik *augmentation* yang mengaburkan citra.

f. *Raise hue*, teknik *augmentation* untuk meningkatkan nilai hue.

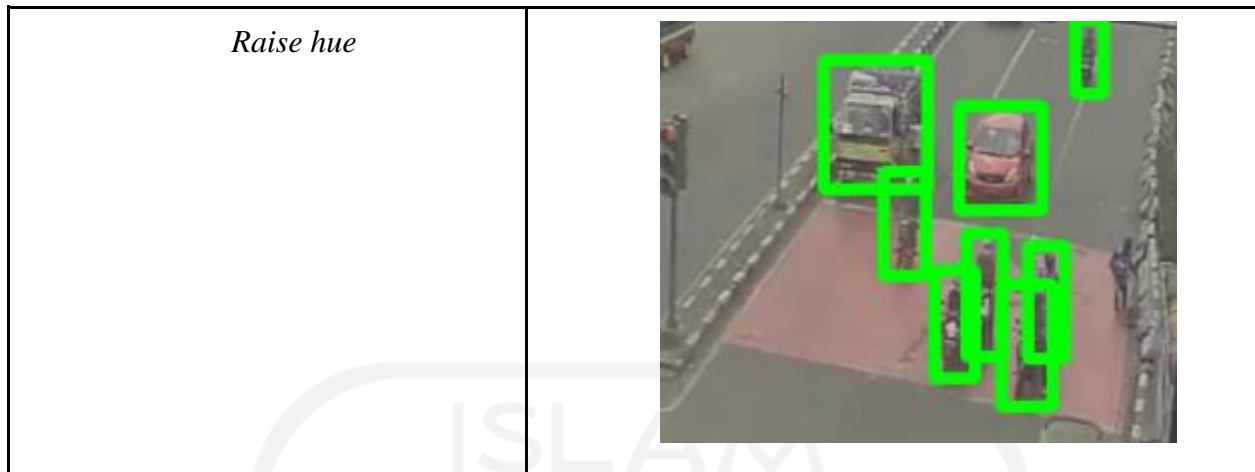
Berikut Tabel 3.1 yang membandingkan citra asli dan citra yang telah diterapkan teknik *augmentation*.

Tabel 3.1 Tabel perbandingan citra asli dan citra hasil teknik *augmentation*

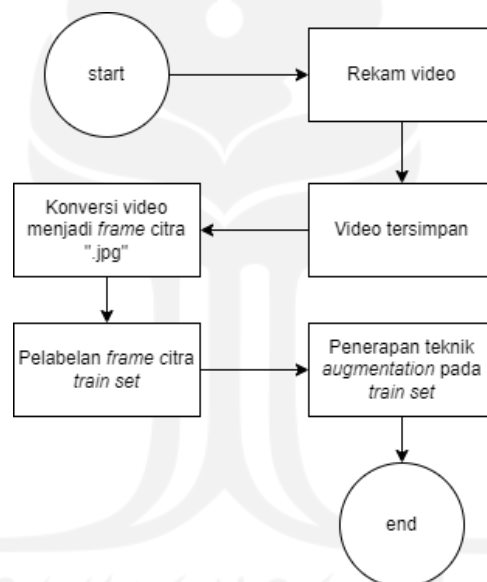
Teknik <i>augmentation</i>	Citra hasil teknik <i>augmentation</i>
<i>Original (none)</i>	
<i>Vertical flip</i>	
<i>Horizontal flip</i>	

<p><i>Horizontal and vertical flip</i></p>	
<p><i>Rotation</i></p>	
<p><i>Average blurring</i></p>	





Untuk lebih ringkas, berikut Gambar 3.5 yang menyajikan alur pengumpulan data pada penelitian ini.

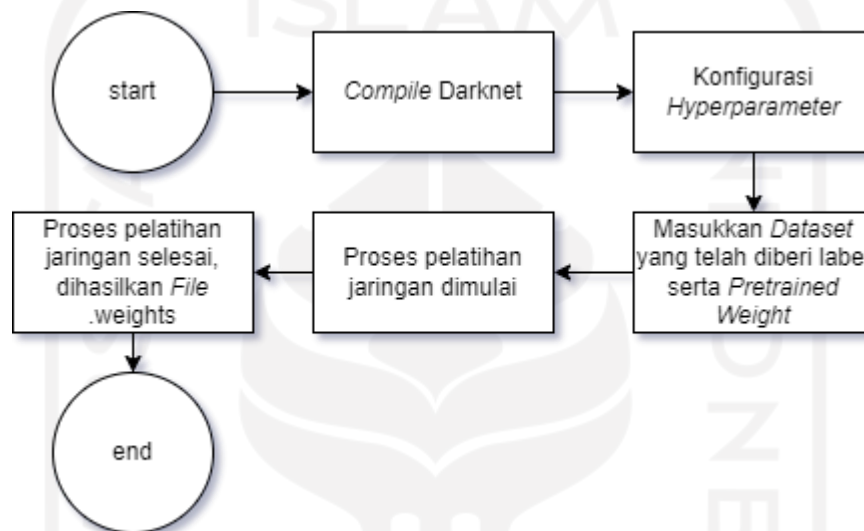


Gambar 3.5 Alur pengumpulan data

### 3.2.4 Pelatihan Jaringan

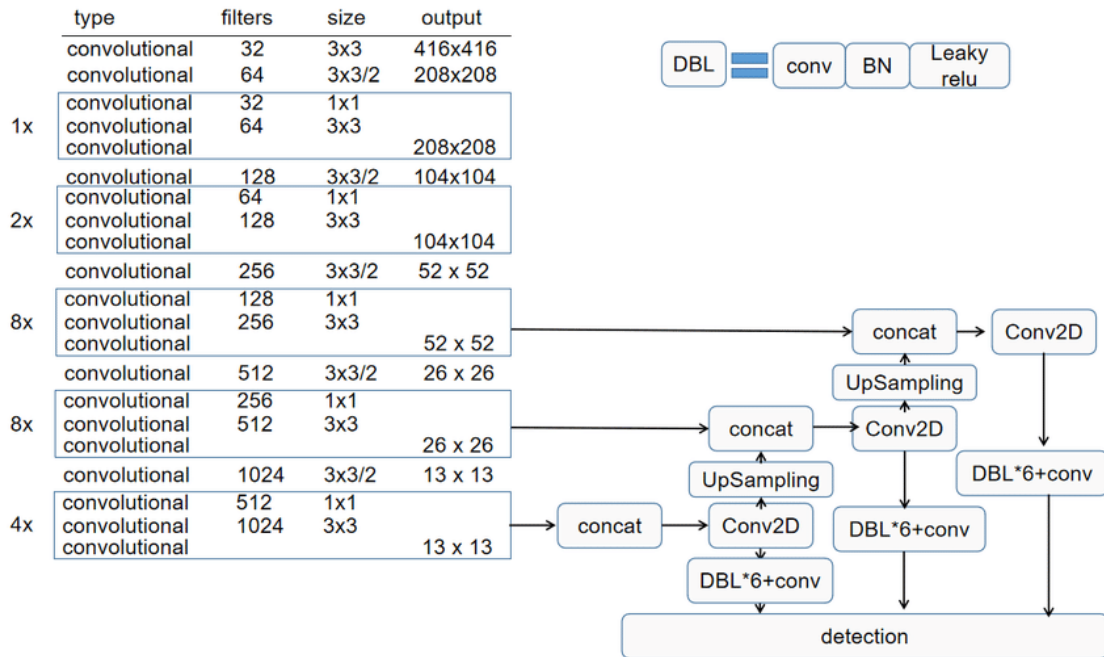
Pada tahap pelatihan jaringan, akan dilatih jaringan YOLO v3 menggunakan proses pelatihan *Transfer Learning* dengan *dataset* yang telah diberi label serta *pretrained weight* “darknet53.conv.74” yang telah dilatih sebelumnya menggunakan *dataset* ImageNet. Proses pelatihan *Transfer Learning* pada jaringan ini akan dilakukan dengan cara *Fine Tuning* yang melakukan pelatihan menggunakan bobot inisiasi dari *pretrained weight* “darknet53.conv.74” tanpa melakukan pembekuan pada lapisan jaringan YOLO v3 yang akan dilatih.

Sebelum proses pelatihan, akan di-*compile* sebuah *framework* pelatihan yang bernama “Darknet” dan ditentukan beberapa nilai *hyperparameter* seperti “classes”, “max\_batches”, dan “filters”. Proses pelatihan akan dilakukan melalui situs Google Collab ([colab.research.google.com](https://colab.research.google.com)) yang menyediakan GPU yang memadai. Proses pelatihan akan memakan banyak waktu dan menghasilkan keluaran yang berupa *file* dengan format *.weights* yang berisikan nilai *weight* atau bobot yang paling bagus berdasarkan *dataset* dan *pretrained weight* yang dimasukkan pada proses pelatihan. Berikut alur yang disajikan pada Gambar 3.6.



Gambar 3.6 Alur pelatihan jaringan

Gambar 3.7 menampilkan arsitektur jaringan deteksi kendaraan YOLO v3 yang akan dibangun pada penelitian ini.



Gambar 3.7 Arsitektur jaringan deteksi kendaraan YOLO v3 pada penelitian ini  
(Ding et al., 2019)

Gambar 3.7 menampilkan arsitektur jaringan deteksi kendaraan YOLO v3 yang akan dibangun pada penelitian ini. Pada Gambar 3.7 dapat dilihat bahwa terdapat tiga jenis lapisan yaitu:

- Lapisan konvolusi yang melakukan proses ekstraksi ciri dari citra masukan.
- Lapisan YOLO atau lapisan deteksi yang melakukan proses deteksi objek citra masukan. Dapat dilihat bahwa terdapat tiga lapisan deteksi yaitu lapisan ke-82, ke-94, dan ke-106, ketiga lapisan ini akan mendeteksi objek dalam citra masukan dengan ukuran yang berbeda-beda sesuai dengan “scale” dan “stride”.
- Lapisan *Upsampling* yang melakukan proses *upsample*, proses ini akan memperbesar ukuran masukan jaringan.

### 3.2.5 Pengujian dan Penarikan Kesimpulan

Pada proses ini, akan dilakukan pengujian terhadap jaringan yang telah dilatih. Pengujian akan dilakukan dengan menghitung *Confusion Matrix*, *Precision*, *Recall*, dan *F1 Score* bobot skenario satu dan bobot skenario dua pada pengujian sepuluh *frame* dari *dataset* penulis. Untuk menghitung *Confusion Matrix* akan terlebih dahulu dibuat tabel perbandingan total kendaraan aktual dengan total kendaraan deteksi jaringan pada masing-masing *frame*. Berikut Tabel 3.2 yang berisi perbandingan tersebut.

Tabel 3.2 Tabel perbandingan total kendaraan aktual dan prediksi

Aktual	Jaringan Skenario Satu	Jaringan Skenario Dua
<i>Frame n</i>		
Sepeda Motor: $TP_1$ Mobil: $TP_2$ Kendaraan Besar: $TP_3$	<i>Frame</i> citra hasil deteksi jaringan skenario satu	<i>Frame</i> citra hasil deteksi jaringan skenario dua
	Sepeda Motor: $TP_1$ Mobil: $TP_2$ Kendaraan Besar: $TP_3$	Sepeda Motor: $TP_1$ Mobil: $TP_2$ Kendaraan Besar: $TP_3$
	Analisis kesalahan deteksi	Analisis kesalahan deteksi
Total		
Sepeda Motor: $\sum_{i=1}^n TP_1$ Mobil: $\sum_{i=1}^n TP_2$ Kendaraan Besar: $\sum_{i=1}^n TP_3$	Sepeda Motor: $\sum_{i=1}^n TP_1$ Mobil: $\sum_{i=1}^n TP_2$ Kendaraan Besar: $\sum_{i=1}^n TP_3$	Sepeda Motor: $\sum_{i=1}^n TP_1$ Mobil: $\sum_{i=1}^n TP_2$ Kendaraan Besar: $\sum_{i=1}^n TP_3$

Kolom “Aktual” berisi total kendaraan sebenarnya yang ada dalam satu *frame* citra, dan kolom “Jaringan Skenario Satu” dan “Jaringan Skenario Dua” berisi *frame* citra hasil deteksi, total kendaraan prediksi, dan analisis kesalahan deteksi dari masing-masing jaringan. Nilai  $TP_1$ ,  $TP_2$ , dan  $TP_3$  adalah nilai total deteksi kendaraan yang benar dideteksi sesuai dengan kelasnya (*True Postives*) pada masing-masing kelas dan  $\sum_{i=1}^n TP_1$ ,  $\sum_{i=1}^n TP_2$ , dan  $\sum_{i=1}^n TP_3$  adalah nilai total akumulatif dari seluruh deteksi kendaraan yang benar dideteksi sesuai dengan kelasnya.

Setelah mendapatkan total kendaraan yang berhasil dideteksi pada masing-masing *frame*, akan dihitung *Confusion Matrix* beserta nilai *Precision*, *Recall*, dan *F1 Score* semua bobot dan

*dataset*. Pada tabel ini, akan dihitung total keseluruhan objek aktual dan deteksi pada masing-masing sepuluh *frame* dataset penulis. Akan dibuat tabel *Confusion Matrix* seperti Tabel 3.3.

Tabel 3.3 Tabel *Confusion Matrix*

<i>Confusion Matrix</i>		Aktual		
		Motor	Mobil	Kendaraan Besar
Prediksi	Motor	$TP_1$	$F_{21}$	$F_{31}$
	Mobil	$F_{12}$	$TP_2$	$F_{32}$
	Kendaraan Besar	$F_{13}$	$F_{23}$	$TP_3$
	Tidak Terdeteksi	$F_{14}$	$F_{24}$	$F_{34}$

Nilai  $TP_1$ ,  $TP_2$ , dan  $TP_3$  adalah total deteksi kendaraan yang benar dideteksi sesuai dengan kelasnya (*True Positives*) pada masing-masing kelas, sedangkan  $F_{21}$ ,  $F_{31}$ ,  $F_{12}$ ,  $F_{32}$ ,  $F_{13}$ ,  $F_{23}$ ,  $F_{14}$ ,  $F_{24}$ , dan  $F_{34}$  adalah total deteksi kendaraan yang salah (*False Positives* atau *False Negatives*). Terakhir, akan ditarik kesimpulan dari penelitian yang telah dilakukan berdasarkan hasil pengujian yang didapatkan.

## BAB IV

### HASIL DAN PEMBAHASAN

#### 4.1 Implementasi

##### 4.1.1 Pengumpulan Data

Pertama akan dilakukan proses perekaman lalu lintas, didapatkan empat video dengan durasi sekitar lima sampai belasan menit. Empat video ini berisi rekaman lalu lintas pada jalan yang berbeda-beda dengan durasi yang berbeda-beda juga, di antaranya adalah Jalan Cibaduyut Bandung (berdurasi 18 menit 13 detik), Jalan Cipaganti Bandung (berdurasi 10 menit 27 detik), Jalan Tol Pasteur Bandung (berdurasi 19 menit 56 detik), dan Jalan Gor Satria Barat Banyumas (berdurasi lima menit 12 detik). Selanjutnya, rekaman lalu lintas akan dikonversi menjadi beberapa *frame* citra berekstensi “.jpg” yang akan dijadikan sebagai data pelatihan jaringan deteksi kendaraan YOLOv3. Berikut baris kode program untuk melakukan konversi rekaman lalu lintas menjadi *frame* citra yang ditunjukkan pada Gambar 4.1.

```
#import libraries
import cv2

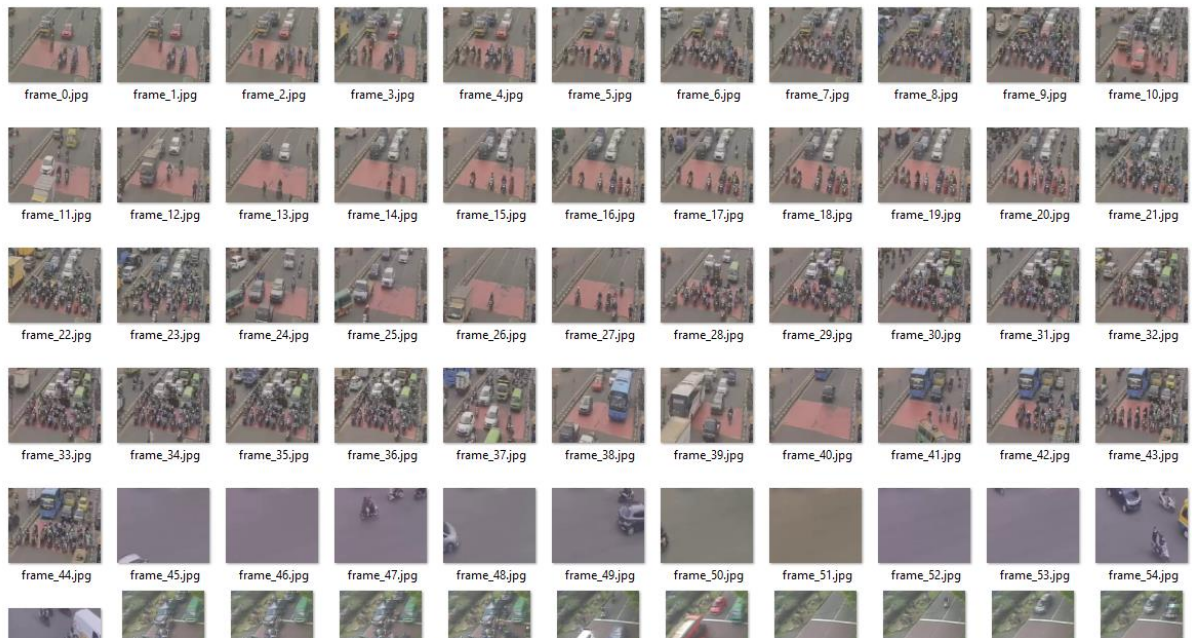
def exportJpg(filepath, angkaUjung):
    vidcap = cv2.VideoCapture(filepath)
    success,image = vidcap.read()
    count = 0
    total = angkaUjung

    while success:
        vidcap.set(0, count)
        cv2.imwrite("jpg4/frame_%d.jpg" % total, image)    #simpan gambar
        success,image = vidcap.read()
        print('Read a new frame: ', success)
        count += 20000
        total += 1
```

Gambar 4.1 Kode program untuk konversi video rekaman menjadi *frame* citra

Pada kode program Gambar 4.1, akan di-*import library* cv2 yang digunakan untuk membantu proses yang berhubungan dengan tugas *computer vision* (seperti penyimpanan dan pembacaan citra atau video). Selanjutnya fungsi exportJpg() akan melakukan proses *export* per seperdetik *frame file* video rekaman lalu lintas berekstensi .mp4 menjadi beberapa *file* citra berekstensi .jpg. Fungsi exportJpg() mempunyai dua parameter yaitu “filepath” yang berisi lokasi file rekaman lalu lintas dan “angkaUjung” yang menunjukkan indeks citra yang diekspor. Fungsi exportJpg() akan membaca video rekaman lalu lintas dan selanjutnya

dilakukan proses *export* dalam perulangan *while* selama dapat dibaca *frame* baru. Pembacaan *frame* baru dilakukan per 20000 milisekon (20 sekon). Berikut Gambar 4.2 yang berisi contoh *frames* citra yang telah dikonversi.



Gambar 4.2 *Frames* citra hasil konversi

Dari proses konversi empat video rekaman lalu lintas ini didapatkan total 167 *frames* citra rekaman lalu lintas dengan:

- a. 56 *frames* citra dari rekaman lalu lintas Jalan Cibaduyut Bandung.
- b. 33 *frames* citra dari rekaman lalu lintas Jalan Cipaganti Bandung.
- c. 62 *frames* citra dari rekaman lalu lintas Jalan Tol Pasteur Bandung.
- d. 17 *frames* citra dari rekaman lalu lintas Jalan Gor Satria Banyumas.

Selanjutnya akan dibuang citra yang kurang memiliki informasi signifikan (seperti citra pada Gambar 4.3) dan citra yang memiliki informasi yang hampir sama dengan citra lainnya (seperti citra pada Gambar 4.4)



Gambar 4.3 *Frames* citra yang kurang memiliki informasi signifikan



Gambar 4.4 *Frames* citra yang memiliki informasi yang hampir sama

Dari pembuangan kedua citra yang kurang signifikan tersebut, didapatkan 119 *frames* citra yang akan dijadikan sebagai *dataset*. Selanjutnya ke-119 *frames* citra ini akan dibagi menjadi 85 *frames* citra *training set* dan 34 *frames* citra *test set*.

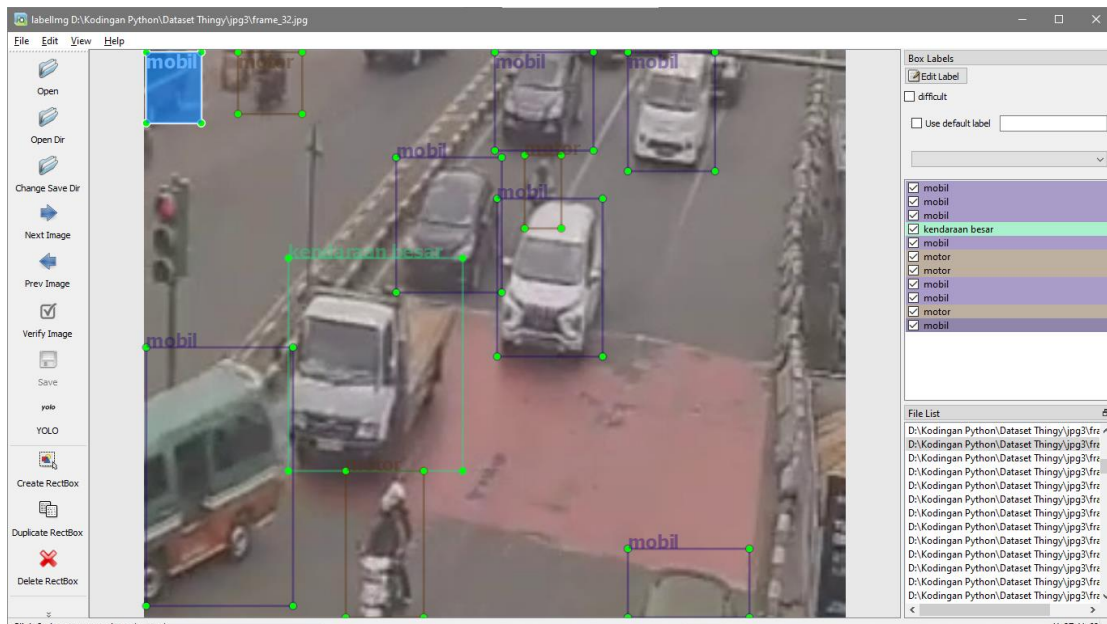
#### 4.1.2 Pelabelan Objek

Setelah melakukan pengumpulan data, selanjutnya akan dilakukan pelabelan objek pada 119 *frames* citra *dataset*. Karena pada penelitian ini ingin dilatih jaringan YOLO v3, maka pelabelan akan dilakukan sesuai format anotasi YOLO yang terdiri dari nomor kelas objek keluaran, koordinat x dan y objek kelas keluaran, serta lebar dan tinggi objek kelas keluaran.

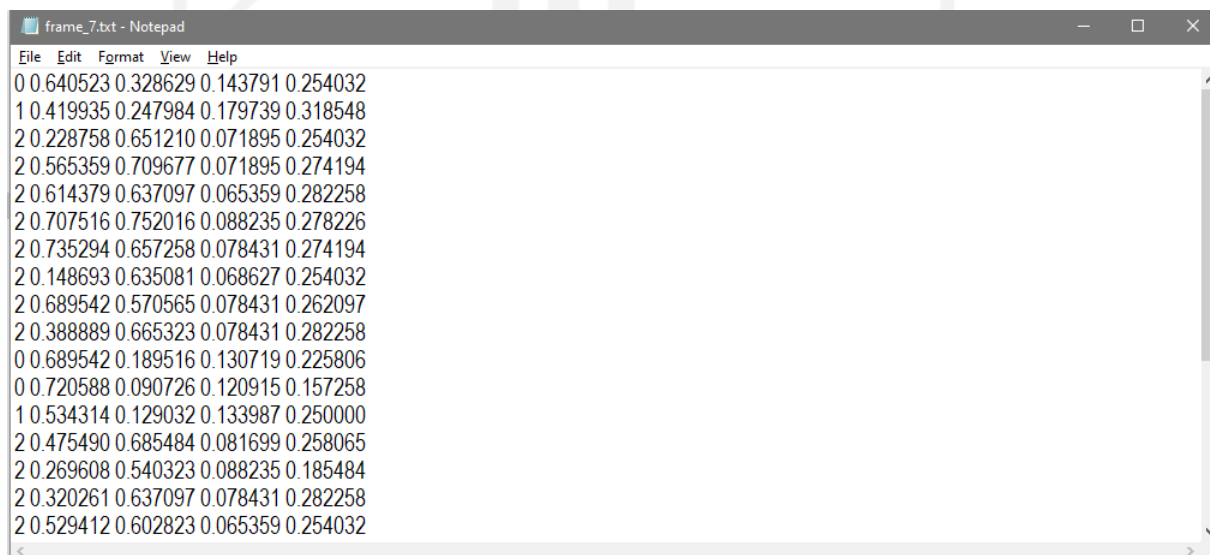
Proses pelabelan objek untuk mobil dilakukan dari bagian atas kiri hingga ujung bawah kanan pada objek mobil. Pelabelan untuk sepeda motor dilakukan dari bagian kepala atau helm pengendara sampai roda atau ban sepeda motor. Terakhir, pelabelan untuk objek kendaraan besar dilakukan dari bagian atas kiri hingga ujung bawah kanan pada objek kendaraan besar.



Proses pelabelan pada penelitian ini dilakukan menggunakan aplikasi labelImg (Tzutalin, 2015) dengan format anotasi YOLO. Hasil dari pelabelan akan disimpan pada *folder* yang juga berisikan *file* citra lalu lintas. Berikut Gambar 4.5 yang menunjukkan proses pelabelan citra dan Gambar 4.6 serta Gambar 4.7 yang menunjukkan isi *file* “.txt” yang didapatkan dari dua proses pelabelan dengan skenario yang berbeda.



Gambar 4.5 Proses pelabelan citra



Gambar 4.6 Isi dari *file* “.txt” pelabelan skenario satu

```

frame_7.txt - Notepad
File Edit Format View Help
0 0.385621 0.661290 0.084967 0.306452
0 0.562092 0.703629 0.078431 0.245968
0 0.617647 0.635081 0.071895 0.286290
0 0.699346 0.750000 0.098039 0.282258
1 0.732026 0.653226 0.071895 0.258065
0 0.318627 0.627016 0.081699 0.262097
2 0.645425 0.322581 0.153595 0.241935
3 0.415033 0.245968 0.196078 0.298387
4 0.532680 0.131048 0.156863 0.254032
4 0.689542 0.191532 0.143791 0.229839
4 0.718954 0.094758 0.124183 0.173387
4 0.035948 0.068548 0.065359 0.129032
0 0.686275 0.580645 0.071895 0.233871
0 0.232026 0.653226 0.084967 0.274194
0 0.147059 0.641129 0.071895 0.274194
0 0.527778 0.602823 0.068627 0.254032
0 0.480392 0.677419 0.078431 0.290323

```

Gambar 4.7 Isi dari *file* “.txt” pelabelan skenario dua

Pada Gambar 4.6, dapat dilihat isi dari *file* “.txt” pelabelan skenario satu yaitu:

- Nomor kelas objek kelas keluaran, 0 untuk motor, 1 untuk mobil, 2 untuk kendaraan besar.
- Koordinat x dan y objek kelas keluaran pada urutan dua dan tiga.
- Lebar dan tinggi objek kelas keluaran (dari kiri ke kanan) pada urutan empat dan lima.

Pada Gambar 4.7, dapat dilihat isi dari *file* “.txt” pelabelan skenario dua yaitu:

- Nomor kelas objek kelas keluaran, 0 untuk motor dilihat dari kanan, 1 untuk motor terlihat separuh, 2 untuk mobil dilihat dari kanan, 3 untuk kendaraan besar dilihat dari kanan, 4 untuk mobil terlihat separuh, 5 untuk kendaraan besar terlihat separuh, 6 untuk kendaraan besar terlihat dari kiri, 7 untuk mobil terlihat dari kiri, 8 untuk motor terlihat dari depan, 9 untuk mobil terlihat dari depan, dan 10 untuk kendaraan besar terlihat dari depan.
- Koordinat x dan y objek kelas keluaran pada urutan dua dan tiga.
- Lebar dan tinggi objek kelas keluaran (dari kiri ke kanan) pada urutan empat dan lima.

#### 4.1.3 Data Augmentation

Setelah proses pelabelan 119 *frames* citra *dataset*, akan dilakukan proses data *augmentation* pada *train set* yang bertujuan untuk menambah variasi data pada *train set*, diharapkan dengan bertambahnya variasi data yang dimiliki akan mampu meningkatkan performa jaringan yang akan dilatih. *Data augmentation* akan dilakukan pada *platform* Google Collaboratory dan menggunakan *library* CLoDSA (Joheras, 2019). *Library* CLoDSA berisikan berbagai macam fungsi untuk melakukan *image augmentation*.

Sebelum dilakukan proses *data augmentation*, *train set* yang sudah dilabeli akan dikonversi menjadi “.zip” dan di-*upload* ke Google Drive penulis. Selanjutnya, akan dilakukan *mounting* (pemasangan) Google Drive pengguna dengan *virtual machine* dari *runtime google colab* seperti pada Gambar 4.8.

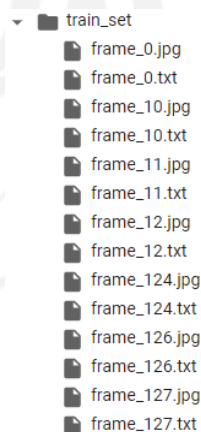
```
from google.colab import drive
drive.mount('/content/gdrive')
!ln -s /content/gdrive/My\ Drive/ /mydrive
!ls /mydrive
```

Gambar 4.8 Kode program mount Google Drive

Fungsi `mount()` akan melakukan *mounting* Google Drive pengguna dengan *virtual machine* dari *runtime*. Selanjutnya *command* “`!ln`” akan membuat *link* antara *virtual machine* dari *runtime* dengan Google Drive pengguna, dan *command* “`!ls`” akan menampilkan seluruh isi file dari Google Drive pengguna, hal ini ditujukan untuk mengecek apakah Google Drive yang disambungkan dengan *virtual machine* sudah benar. Setelah itu, akan dilakukan proses *unzipping file .zip dataset* citra yang telah diberi label dengan *command* “`!unzip`” seperti pada Gambar 4.9 beserta citra dan anotasinya yang telah di-*unzip* pada Gambar 4.10.

```
!unzip "/mydrive/Skripsi/DatasetAugmentation/train2.zip" -d
/content/train_set
```

Gambar 4.9 Kode program *unzip file dataset*



Gambar 4.10 *Dataset* citra yang telah di-*unzip*

Selanjutnya, perlu dilakukan perhitungan total *file* berekstensi “.jpg” (*file* citra) dan “.txt” (*file* anotasi) karena proses augmentasi dengan *library* CLoDSA tidak dapat dilakukan apabila

total *file* citra dan anotasi tidak sama. Dilakukan pengecekan total *file* citra dan anotasi dengan menjalankan kode program pada Gambar 4.11 beserta keluarannya pada Gambar 4.12.

```
print("Number of images in the folder")
!ls -l /content/train_set/*.jpg | wc -l

print("Number of annotations in the folder")
!ls -l /content/train_set/*.txt | wc -l
```

Gambar 4.11 Kode program menghitung total *file* citra dan anotasi

```
Number of images in the folder
85

Number of annotations in the folder
85
```

Gambar 4.12 Total file citra dan anotasi

Dari keluaran pada Gambar 4.12, diketahui bahwa total *file* citra dan anotasi memiliki nilai yang sama. Oleh karena itu, proses *augmentation* dapat diteruskan. Pertama, akan di-*import libraries* yang digunakan seperti pada Gambar 4.13.

```
from matplotlib import pyplot as plt
from clodsa.augmentors.augmentorFactory import createAugmentor
from clodsa.transformers.transformerFactory import transformerGenerator
from clodsa.techniques.techniqueFactory import createTechnique
import xml.etree.ElementTree as ET
import cv2
%matplotlib inline
```

Gambar 4.13 Kode program *import libraries*

Selanjutnya akan dibangun objek “Augmentor” dengan fungsi `createAugmentor()` seperti pada Gambar 4.14. Objek ini akan menyimpan beberapa parameter yang mengatur proses augmentasi, di antaranya adalah:

- “PROBLEM” yang mendeskripsikan masalah yang ingin diselesaikan, penelitian ini ingin menyelesaikan permasalahan deteksi citra, oleh karena itu parameter ini bernilai “detection”.
- “ANNOTATION\_MODE” yang mendeskripsikan format anotasi, format anotasi yang digunakan pada penelitian ini adalah YOLO, oleh karena itu parameter ini bernilai “yolo”.
- “INPUT\_PATH” yang mendeskripsikan lokasi *file dataset* citra yang ingin dilakukan augmentasi, oleh karena itu parameter ini bernilai “train\_set”.

- d. “GENERATION\_MODE” yang mendeskripsikan bagaimana augmentasi diterapkan, karena pada penelitian ini ingin diterapkan augmentasi ke semua citra, maka parameter ini bernilai “linear”.
- e. “OUTPUT\_MODE” yang mendeskripsikan bagaimana format anotasi *file* hasil augmentasi, karena pada penelitian ini ingin digunakan format YOLO, maka parameter ini bernilai “yolo”.
- f. “OUTPUT\_PATH” yang mendeskripsikan lokasi *file* hasil augmentasi, maka parameter ini bernilai “augmented\_output”.

```

PROBLEM = "detection"
ANNOTATION_MODE = "yolo"
INPUT_PATH = "train_set"
GENERATION_MODE = "linear"
OUTPUT_MODE = "yolo"
OUTPUT_PATH= "augmented_output"

augmentor =
createAugmentor (PROBLEM, ANNOTATION_MODE, OUTPUT_MODE, GENERATION_MODE, INPUT_P
ATH, {"outputPath":OUTPUT_PATH})

```

Gambar 4.14 Kode program untuk membangun objek “Augmentor”

Selanjutnya, akan ditambahkan beberapa teknik *augmentation* ke dalam objek “Augmentor” yaitu *vertical flip*, *horizontal flip*, *horizontal and vertical flip*, *rotation*, *average blurring*, dan *raise hue*. Penambahan teknik *augmentation* ini dapat dilakukan dengan menjalankan kode program pada Gambar 4.15.

```

#define transformer
transformer = transformerGenerator (PROBLEM)

#vertical flip
vFlip = createTechnique ("flip", {"flip":0})
augmentor.addTransformer (transformer (vFlip))

#horizontal flip
hFlip = createTechnique ("flip", {"flip":1})
augmentor.addTransformer (transformer (hFlip))

#horizontal and vertical flip
hvFlip = createTechnique ("flip", {"flip":-1})
augmentor.addTransformer (transformer (hvFlip))

#rotation
rotate = createTechnique ("rotate", {"angle" : 90})
augmentor.addTransformer (transformer (rotate))

#average blurring

```

```

avgBlur = createTechnique("average_blurring", {"kernel" : 5})
augmentor.addTransformer(transformer(avgBlur))

#raise hue
hue = createTechnique("raise_hue", {"power" : 0.9})
augmentor.addTransformer(transformer(hue))

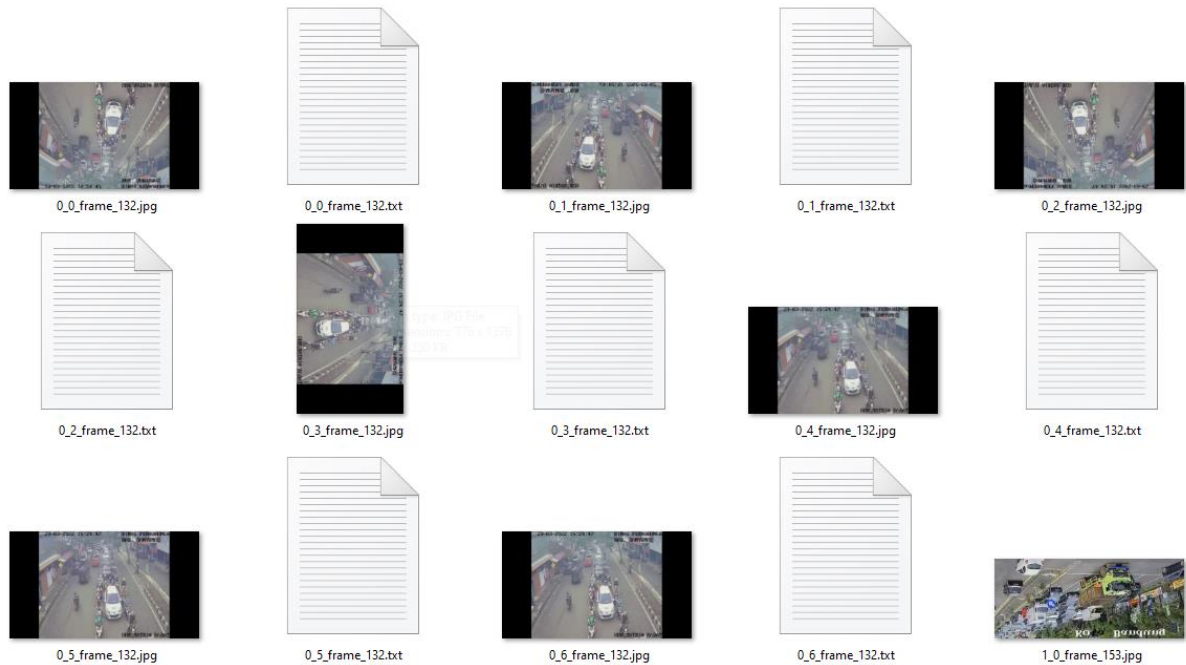
#none
none = createTechnique("none", {})
augmentor.addTransformer(transformer(none))

#apply transformer
augmentor.applyAugmentation()

```

Gambar 4.15 Kode program untuk menambahkan teknik augmentasi ke objek “Augmentor”

Agar *dataset* citra yang asli (tidak diterapkan augmentasi) tidak hilang, dibuat objek “none” yang mengaplikasikan *transformer* kosong. Fungsi `applyAugmentation()` akan menerapkan augmentasi yang telah ditentukan kepada *dataset* citra. Setelah diterapkan proses augmentasi, didapatkan total 595 citra *train set* beserta anotasinya yang telah teraugmentasi seperti Gambar 4.16.



Gambar 4.16 Citra hasil *augmentation*

#### 4.1.4 Pelatihan Jaringan

Setelah melakukan pelabelan dan augmentasi dataset terhadap seluruh data pelatihan untuk setiap skenario yang ada, tahap selanjutnya adalah melakukan pelatihan atau *training* jaringan deteksi kendaraan YOLO v3. Proses pelatihan ini akan dilakukan sebanyak dua kali, pelatihan pertama untuk skenario satu dan pelatihan kedua untuk skenario dua. Pada pelatihan ini, akan digunakan 595 *frames* citra *train set* rekaman lalu lintas beserta bobot *Pretrained* “darknet53.conv.74”. Proses pelatihan jaringan YOLO v3 akan dilakukan pada platform Google Collaboratory dengan menggunakan *framework* Darknet (Bochkovskiy, 2014). Pertama, perlu dilakukan proses *mounting* Google Drive dengan menjalankan kode program pada seperti pada Gambar 4.8. Setelah itu, akan dilakukan proses clone framework Darknet beserta konfigurasinya dengan menjalankan kode program Gambar 4.17.

```
# Clone
!git clone https://github.com/AlexeyAB/darknet

# Configure
%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile

# Compile
!make
```

Gambar 4.17 Kode program *clone*, *configure*, dan *make* Darknet

Pemberian nilai 1 pada ketiga *hyperparameter* tersebut berarti bahwa proses pelatihan akan menggunakan OpenCV, GPU, dan CUDNN, dengan penggunaan ketiga komponen ini, kecepatan proses pelatihan akan meningkat.

Selain konfigurasi ketiga *hyperparameter* Darknet, perlu dilakukan juga konfigurasi untuk *hyperparameter* jaringan YOLO v3 yang akan dilatih. *Hyperparameter* ini di antaranya adalah:

- a. “classes” yang mendeskripsikan total kelas keluaran, karena pada penelitian ini terdapat dua skenario pelabelan yang memiliki total kelas yang berbeda, maka nilai “classes” untuk skenario satu adalah tiga (motor, mobil, dan kendaraan besar) dan untuk skenario dua adalah sebelas (motor dilihat dari kanan, motor dilihat dari depan, motor terlihat separuh, mobil dilihat dari kanan, mobil dilihat dari kiri, mobil dilihat dari depan, mobil terlihat separuh, kendaraan besar dilihat dari kanan, kendaraan besar dilihat dari kiri, kendaraan besar dilihat dari depan, dan kendaraan besar terlihat separuh).

- b. “max\_batches” yang mendeskripsikan total maksimal iterasi yang dilakukan pada saat pelatihan. “max\_batches” bernilai total kelas dikalikan dengan 2000 (Tiwari, 2020) yang berarti untuk skenario satu, “max\_batches” bernilai 6000 dan 22000 untuk skenario dua. Nilai 2000 ini merupakan rekomendasi total minimal iterasi proses pembelajaran untuk satu kelas (Robins & Krok, 2019).
- c. “filters” yang mendeskripsikan total *kernel filter* yang digunakan pada *layer* pendeteksian YOLO, “filters” bernilai  $(total\ kelas + 5) * 3$  (Tiwari, 2020) yang artinya “filters” pada skenario satu adalah 24 dan 48 untuk skenario dua. Nilai lima di sini didapatkan dari total komponen yang ada pada sebuah *bounding box* yang berjumlah lima yaitu panjang, lebar, koordinat x, koordinat y, dan prediksi dari sebuah *bounding box*. Sedangkan nilai tiga di sini didapatkan dari total banyaknya prediksi yang dilakukan pada jaringan YOLO v3 yaitu sebanyak tiga kali.

Konfigurasi ini akan dilakukan pada *file* “yolov3\_training.cfg” yang nantinya akan digunakan pada proses pelatihan. Berikut kode program Gambar 4.18 dan Gambar 4.19 yang digunakan melakukan konfigurasi jaringan YOLO v3 skenario satu dan skenario dua.

```
# Make a copy of yolov3_training.cfg
!cp cfg/yolov3.cfg cfg/yolov3_training.cfg

# Change lines in yolov3.cfg file
!sed -i 's/batch=1/batch=32/' cfg/yolov3_training.cfg
!sed -i 's/subdivisions=1/subdivisions=8/' cfg/yolov3_training.cfg
!sed -i 's/max_batches = 500200/max_batches = 6000/' cfg/yolov3_training.cfg
!sed -i '610 s@classes=80@classes=3@' cfg/yolov3_training.cfg
!sed -i '696 s@classes=80@classes=3@' cfg/yolov3_training.cfg
!sed -i '783 s@classes=80@classes=3@' cfg/yolov3_training.cfg
!sed -i '603 s@filters=255@filters=24@' cfg/yolov3_training.cfg
!sed -i '689 s@filters=255@filters=24@' cfg/yolov3_training.cfg
!sed -i '776 s@filters=255@filters=24@' cfg/yolov3_training.cfg
```

Gambar 4.18 Kode program untuk konfigurasi *hyperparameter* jaringan YOLO v3 untuk *dataset* pelabelan skenario satu

```
# Make a copy of yolov3_training.cfg
!cp cfg/yolov3.cfg cfg/yolov3_training.cfg

# Change lines in yolov3.cfg file
!sed -i 's/batch=1/batch=32/' cfg/yolov3_training.cfg
!sed -i 's/subdivisions=1/subdivisions=8/' cfg/yolov3_training.cfg
!sed -i 's/max_batches = 500200/max_batches = 22000/'
cfg/yolov3_training.cfg
!sed -i '610 s@classes=80@classes=11@' cfg/yolov3_training.cfg
!sed -i '696 s@classes=80@classes=11@' cfg/yolov3_training.cfg
!sed -i '783 s@classes=80@classes=11@' cfg/yolov3_training.cfg
!sed -i '603 s@filters=255@filters=48@' cfg/yolov3_training.cfg
```



```
!sed -i '689 s@filters=255@filters=48@' cfg/yolov3_training.cfg
!sed -i '776 s@filters=255@filters=48@' cfg/yolov3_training.cfg
```

Gambar 4.19 Kode program untuk konfigurasi *hyperparameter* jaringan YOLO v3 untuk *dataset* pelabelan skenario dua

Pada kode program Gambar 4.18 dan Gambar 4.19, perintah “!cp” akan membuat salinan dari *file* “yolov3\_training.cfg” yang berisikan konfigurasi jaringan YOLO v3 yang akan dilatih. Hal ini dilakukan agar *framework* Darknet yang telah diunduh ke dalam penyimpanan *virtual machine runtime* tidak terganggu. Selanjutnya, perintah “!sed” akan mengubah beberapa baris pada salinan *file* “yolov3\_training.cfg” tersebut. Baris-baris yang diubah antara lain adalah baris-baris yang berisikan nilai *hyperparameter* “classes”, “max\_batches”, dan “filters”. Perubahan dilakukan pada nilai ketiga *hyperparameter* tersebut sesuai dengan skenario pelabelannya.

Setelah itu, akan dibuat *file* “obj.names” yang berisikan nama kelas dan *file* “obj.data” yang berisikan *metadata* seperti total kelas keluaran, *path file* train.txt, test.txt dan lokasi penyimpanan *file* bobot “.weight” yang terakhir dilatih. Berikut Gambar 4.20 dan Gambar 4.21 untuk membuat *file* “obj.names” dan “obj.data” untuk skenario satu dan skenario dua.

```
!echo -e 'mobil\nkendaraan besar\nmotor' > data/obj.names
!echo -e 'classes= 3\ntrain = data/train.txt\nvalid = data/test.txt\nnames
= data/obj.names\nbackup = /mydrive/yolov3' > data/obj.data
```

Gambar 4.20 Kode program untuk membuat *file* “obj.names” dan “obj.data” skenario

satu

```
!echo -e
'motor_kanan\nmotor_separuh\nmobil_kanan\nkendaraan_besar_kanan\nmobil_sepa
ruh\nkendaraan_besar_separuh\nkendaraan_besar_kiri\nmobil_kiri\nmotor_depan
\nmobil_depan\nkendaraan_besar_depan' > data/obj.names
!echo -e 'classes= 11\ntrain = data/train.txt\nvalid =
data/test.txt\nnames = data/obj.names\nbackup = /mydrive/yolov3' >
data/obj.data
```

Gambar 4.21 Kode program untuk membuat *file* “obj.names” dan “obj.data” skenario dua

Pada kode program Gambar 4.20 dan Gambar 4.21, perintah “!echo” akan menulis nama-nama kelas ke dalam *file* “obj.names” serta *metadata* sesuai dengan skenario pelabelannya ke dalam *file* “obj.data”.

Setelah itu, akan disimpan *file* “yolov3\_testing.cfg” yang berisi konfigurasi jaringan YOLO v3 yang akan dibangun dan *file* “classes.txt” yang berisi nama kelas keluaran

menggunakan perintah “!cp” seperti kode program Gambar 4.22. Kedua *file* ini akan digunakan untuk melakukan pengujian nantinya.

```
!cp cfg/yolov3_training.cfg "/mydrive/Skripsi/skenario2/yolov3_testing.cfg"
!cp data/obj.names "/mydrive/Skripsi/skenario2/classes.txt"
```

Gambar 4.22 Kode program untuk menyimpan file “yolov3\_testing.cfg” dan “classes.txt” ke dalam Google Drive penulis

Selanjutnya, akan dibuat *folder* untuk menyimpan *dataset* yang digunakan untuk pelatihan serta *unzipping dataset* yang telah di-*upload* ke Google Drive penulis seperti pada kode program Gambar 4.23.

```
!mkdir data/obj
!unzip "/mydrive/Skripsi/kodingan training jaringan
YOLO/augmented_train1.zip" -d data/obj
```

Gambar 4.23 Kode program untuk membuat *folder* “obj” dan *unzipping dataset*

Selanjutnya, akan dibuat *file* “train.txt” yang berisi seluruh nama *path file dataset* citra yang akan dilatih. *File* “train.txt” akan digunakan untuk menunjukkan *dataset* citra pelatihan beserta anotasinya kepada *framework* Darknet. Untuk mendapatkan *path file dataset* citra yang akan dilatih, digunakan fungsi *glob()* seperti Gambar 4.24.

```
import glob
images_list = glob.glob("data/obj/augmented_output/*.jpg")
with open("data/train.txt", "w") as f:
    f.write("\n".join(images_list))
```

Gambar 4.24 Kode program untuk membuat *file* “train.txt” dan menulis seluruh *file path dataset* ke dalam *file* “train.txt”

Selanjutnya, akan dilakukan pengunduhan bobot *pretrained* “darknet53.conv.74” yang akan digunakan saat proses pelatihan jaringan seperti pada Gambar 4.25.

```
!wget https://pjreddie.com/media/files/darknet53.conv.74
```

Gambar 4.25 Kode program untuk mengunduh bobot *pretrained* “darknet53.conv.74”

Dengan perintah “!wget” akan dilakukan pengunduhan satu *file pretrained weight* yang berjudul “darknet53.conv.74”. Terakhir, akan dilakukan proses pelatihan jaringan deteksi

kendaraan YOLO v3 menggunakan *dataset* citra rekaman lalu lintas dan bobot *pretrained* “darknet53.conv.74” seperti pada Gambar 4.26.

```
!./darknet detector train data/obj.data cfg/yolov3_training.cfg
darknet53.conv.74 -dont_show
```

Gambar 4.26 Kode program untuk pelatihan jaringan deteksi kendaraan YOLO v3

Perintah “!./darknet” akan membuka *file* “darknet” yang berisikan berbagai fungsi untuk melatih jaringan YOLO v3. Fungsi “detector train” seperti pada kode program Gambar 4.26 akan melakukan pelatihan terhadap *train set* yang *path*-nya terdapat pada *file* “obj.data” dengan konfigurasi jaringan yang sesuai dengan konfigurasi yang telah diatur pada *file* “yolov3\_training.cfg” serta menggunakan *pretrained weight* “darknet53.conv.74” yang digunakan untuk proses *Transfer Learning*.

Selama proses pelatihan, akan disimpan *file* “yolov3\_training\_last.weights” yang berisi bobot yang terakhir dilatih dalam interval waktu tertentu ke dalam *path backup* yang telah ditentukan dalam *file* “obj.data”. Jadi, jika proses pelatihan terganggu dikarenakan berbagai alasan, dapat di-*load file* “.weights” terakhir ke dalam proses pelatihan yang akan dilakukan seperti kode program Gambar 4.27.



```
!./darknet detector train data/obj.data cfg/yolov3_training.cfg
/mydrive/yolov3/yolov3_training_last.weights -dont_show
```

Gambar 4.27 Kode program untuk pelanjutan pelatihan jaringan deteksi kendaraan YOLO v3 yang terganggu

Kode program pada Gambar 4.27 menjalankan perintah dan fungsi yang sama dengan kode program Gambar 4.26, bedanya adalah pada kode program Gambar 4.27, digunakan bobot yang terakhir kali disimpan selama proses pelatihan sebelum terganggu yaitu *file* bobot “yolov3\_training\_last.weights”.

Saat proses pelatihan selesai dilakukan, akan dihasilkan *file* “yolov3\_training\_final.weights” yang berisi bobot final dari jaringan yang telah dilatih seperti pada Gambar 4.28. Lamanya proses pelatihan ini tergantung dengan banyaknya data, total kelas keluaran, dan *max\_batches* yang telah ditentukan. Untuk proses pelatihan *dataset* skenario satu, dibutuhkan waktu sekitar 16 jam untuk menyelesaikan proses pelatihan, dan untuk proses pelatihan *dataset* skenario dua, dibutuhkan waktu sekitar 22 jam untuk menyelesaikan proses

pelatihan. Perbedaan waktu pelatihan ini disebabkan oleh perbedaan nilai *hyperparameter* “classes” (total kelas) dan *max\_batches* masing-masing skenario. Selama proses pelatihan berlangsung, terdapat beberapa kali gangguan di antaranya yang lumayan sering adalah jatah pemakaian GPU yang habis dan koneksi internet yang buruk yang menyebabkan pemutusan *runtime*.

 yolov3_training_last.weights	me	Apr 15, 2022 me	235.1 MB
 yolov3_training_final.weights	me	Apr 15, 2022 me	235.1 MB

Gambar 4.28 File “yolov3\_training\_last.weights” dan “yolov3\_training\_final.weight” hasil pelatihan jaringan

#### 4.1.5 Proses Deteksi Jaringan YOLO v3

Untuk melakukan proses deteksi menggunakan jaringan YOLO v3 yang telah dilatih, pertama dilakukan pengunduhan *file* “yolov3\_training\_final.weights” yang berisi bobot final jaringan YOLO v3 yang telah dilatih, *file* “yolov3\_testing.cfg” yang berisi konfigurasi jaringan YOLO v3 yang telah dilatih, dan *file* “classes.txt” yang berisi nama-nama kelas objek keluaran dari Google Drive penulis. Pembuatan *file* “yolov3\_testing.cfg” dan “classes.txt” dapat dilihat pada Gambar 4.22.

Selanjutnya akan dilakukan proses deteksi kendaraan pada citra *test set* menggunakan bobot jaringan yang telah dilatih. Pertama akan di-*import* beberapa *libraries* yang dibutuhkan seperti pada Gambar 4.29.

```
import cv2
import numpy as np
```

Gambar 4.29 Kode program import libraries

*Library* *cv2* digunakan untuk mempermudah proses-proses yang berhubungan dengan citra dan *library* *numpy* digunakan untuk proses perhitungan matematis. Selanjutnya akan di-*load* jaringan deteksi kendaraan YOLO v3 yang telah dilatih menggunakan fungsi *readNet()*. Akan dimasukkan dua parameter pada fungsi *readNet()* yaitu *file* bobot final dan konfigurasi jaringan YOLO v3 seperti pada Gambar 4.30.

```
net = cv2.dnn.readNet('yolov3_training_final.weights',
'yolov3_testing.cfg')
```

Gambar 4.30 Kode program load jaringan deteksi kendaraan YOLO v3 skenario satu

Selanjutnya akan di-*load* nama kelas keluaran ke dalam *list* “classes” menggunakan fungsi `read()` seperti kode program Gambar 4.31.

```
classes = []
with open("nama_file_kelas.txt", "r") as f:
    classes = f.read().splitlines()
```

Gambar 4.31 Kode program *load* nama kelas keluaran ke dalam *list* “classes”

Setelah itu akan dilakukan proses pembacaan 10 *frames* citra *test set* yang ingin diuji dengan menggunakan fungsi `imread()`, selain itu akan di-*define* variabel “height” yang berisi dimensi tinggi citra dan “width” yang berisi dimensi lebar citra. Dari sini akan dimulai perulangan “while” yang akan terus berlangsung selama pengguna tidak menekan tombol “esc” pada *keyboard*. Akan di-*define* variabel “font” yang mengatur jenis *font* yang digunakan *bounding box* serta “colors” yang mengatur warna *bounding box*. Kode program Gambar 4.32 ditujukan untuk menunjukkan hal ini.

```
font = cv2.FONT_HERSHEY_PLAIN
colors = np.random.uniform(0, 255, size=(len(classes), 3))

while True:
    img = cv2.imread("nama_file_citra.jpg")
    height, width, _ = img.shape
```

Gambar 4.32 Kode program pembacaan citra yang ingin diuji

Selanjutnya akan di-*define* objek “blob” menggunakan fungsi `blobFromImage()` seperti pada kode program Gambar 4.33. Objek “blob” ini akan dijadikan masukan untuk jaringan dengan menggunakan fungsi `setInput()`.

```
blob = cv2.dnn.blobFromImage(img, 1/255, (416, 416), (0,0,0),
swapRB=True, crop=False)
net.setInput(blob)
```

Gambar 4.33 Kode program *define* “blob” dan menjadikan “blob” sebagai masukan jaringan

Selanjutnya akan dilakukan *forward pass* untuk menghitung keluaran dari jaringan YOLO v3 yang telah dilatih seperti pada Gambar 4.34. Lapisan deteksi ini akan terlebih dahulu di-*define* menggunakan fungsi `getUnconnectedOutLayersNames()`.

```
output_layers_names = net.getUnconnectedOutLayersNames()
layerOutputs = net.forward(output_layers_names)
```

Gambar 4.34 Kode program *define* “blob” dan menjadikan “blob” sebagai masukan jaringan

Setelah itu akan dibuat *bounding box*/kotak pembatas dengan menjalankan kode program Gambar 4.35.

```
boxes = []
confidences = []
class_ids = []

for output in layerOutputs:
    for detection in output:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.2:
            center_x = int(detection[0]*width)
            center_y = int(detection[1]*height)
            w = int(detection[2]*width)
            h = int(detection[3]*height)

            x = int(center_x - w/2)
            y = int(center_y - h/2)

            boxes.append([x, y, w, h])
            confidences.append(float(confidence))
            class_ids.append(class_id)

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.2, 0.4)
```

Gambar 4.35 Kode program pembuatan *bounding box*

Pertama akan di-*define* “boxes” yang akan menyimpan nilai titik koordinat x dan y serta nilai w (lebar) dan h (tinggi) *bounding box* kendaraan, “confidences” yang akan menyimpan nilai *confidence*/keyakinan kendaraan yang dideteksi, dan “class\_ids” yang akan menyimpan ID kelas keluaran kendaraan yang dideteksi. Selanjutnya, akan dilakukan perulangan “for” untuk mengambil semua hasil deteksi yang telah dilakukan proses *forward pass* sebelumnya. Penulis menentukan nilai 0,2 sebagai nilai *threshold confidence*, jadi jika objek yang dideteksi mempunyai nilai *confidence* yang kurang dari 0,2 maka hasil deteksi tersebut akan dibuang.

Setelah itu akan dilakukan operasi *Non-maximum Supression* (NMS) yang bertujuan untuk membuang *bounding box* yang lokasinya kurang tepat. Ditentukan nilai 0,2 sebagai *threshold confidences* dan 0,4 sebagai *threshold NMS*. Selanjutnya dilakukan proses penulisan *bounding box* ke dalam *frame* citra deteksi seperti pada Gambar 4.36.

```

motor = 0
mobil = 0
kendaraan_besar = 0

if len(indexes)>0:
    for i in indexes.flatten():
        x, y, w, h = boxes[i]
        label = str(classes[class_ids[i]])
        confidence = str(round(confidences[i],2))
        color = colors[class_ids[i]].tolist()
        cv2.rectangle(img, (x,y), (x+w, y+h), color, 2)
        cv2.putText(img, label + " " + confidence, (x, y+20), font, 1,
                    (255,255,255), 2)
        if class_ids[i] == 2:
            motor += 1
        if class_ids[i] == 0:
            mobil += 1
        if class_ids[i] == 1:
            kendaraan_besar += 1

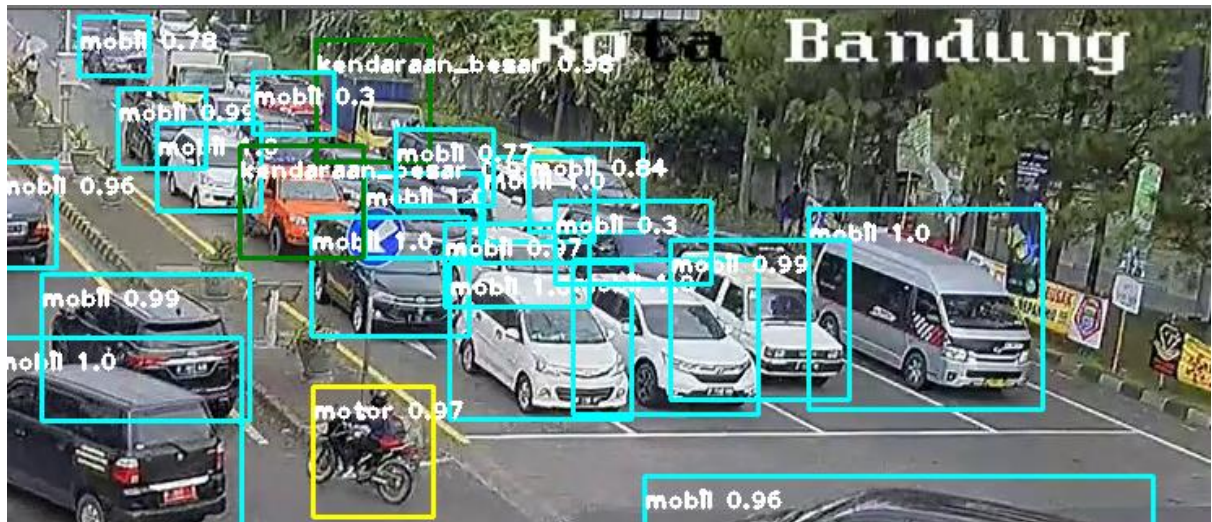
    cv2.imshow('Image', img)
    key = cv2.waitKey(1)
    if key==27:
        break

print('Sepeda Motor: ' + str(motor))
print('Mobil: ' + str(mobil))
print('Kendaraan Besar: ' + str(kendaraan_besar))

```

Gambar 4.36 Kode program penulisan *bounding box* ke citra

Fungsi `rectangle()` akan membuat *bounding box* berbentuk persegi panjang, dan fungsi `putText()` akan menaruh *bounding box* yang sudah dibuat ke dalam citra deteksi. Variabel “motor”, “mobil”, dan “kendaraan\_besar” digunakan untuk menyimpan total kendaraan yang berhasil dideteksi. Total kendaraan ini akan dihitung pada setiap iterasi deteksi dan total keseluruhannya akan ditampilkan saat semua iterasi deteksi telah selesai dilakukan. Logika “if” digunakan untuk mengakhiri *display window* yang menampilkan citra hasil deteksi jika pengguna menekan tombol “esc” pada *keyboard*. Gambar 4.37 menampilkan citra hasil deteksi jaringan YOLO v3.



Gambar 4.37 Hasil citra deteksi

```

Sepeda Motor: 1
Mobil: 19
Kendaraan Besar: 2

```

Gambar 4.38 Total kendaraan deteksi

## 4.2 Pengujian Performa Jaringan

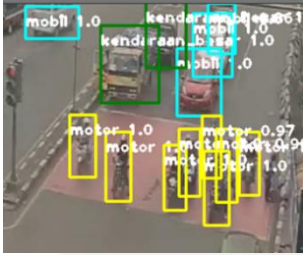
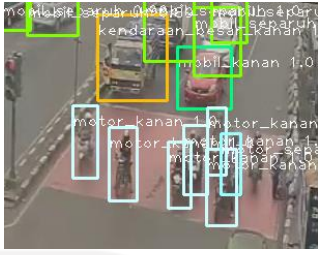


Pengujian performa jaringan YOLO v3 akan dilakukan dengan menghitung nilai *confusion matrix*, *recall*, *precision*, serta *F1 score* jaringan deteksi kendaraan yang sudah dilatih. Akan dilakukan pengujian performa kedua jaringan yang telah dilatih dengan menggunakan sepuluh *frame* dari *test set* penulis. Kesepuluh *frame* yang diambil dari *test set* penulis merupakan *frame* citra lalu lintas yang merepresentasikan keadaan paling sepi dan paling padat dari keempat rekaman jalan lalu lintas yang diambil.

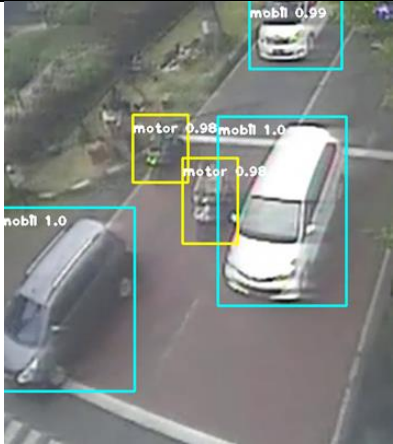
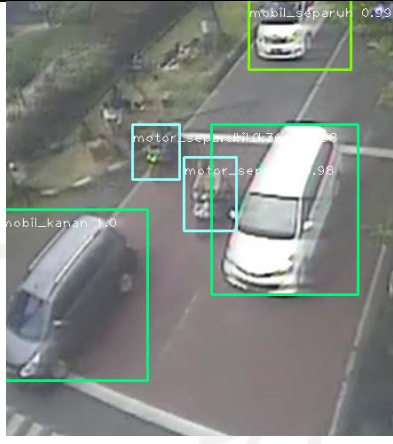
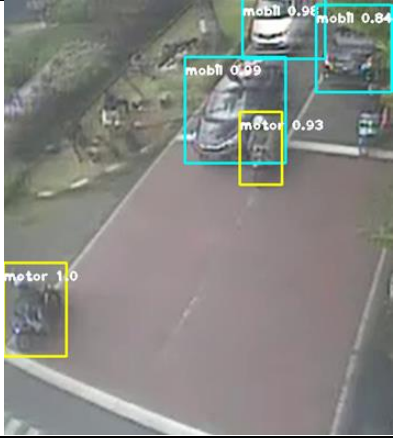
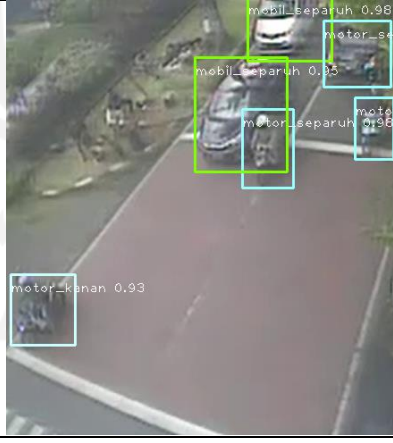
### 4.2.1 Pengujian Performa Jaringan pada *Dataset* Penulis

Akan dilakukan pengujian performa jaringan skenario satu dan skenario dua penulis dengan membandingkan total kendaraan aktual dengan total kendaraan prediksi dari jaringan dengan menggunakan *dataset* penulis seperti yang dipaparkan Tabel 4.1.

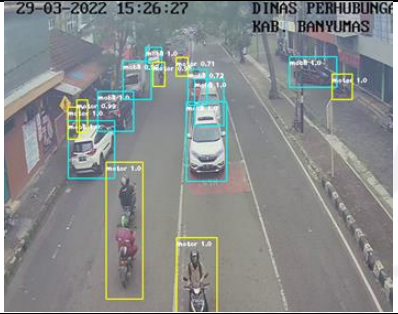
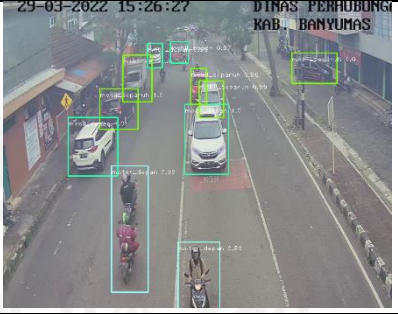
Tabel 4.1 Tabel perbandingan total kendaraan aktual dan total kendaraan prediksi ketiga jaringan pada *dataset* penulis

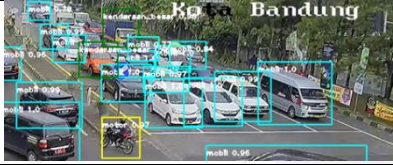
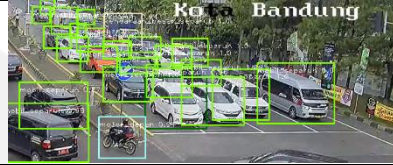


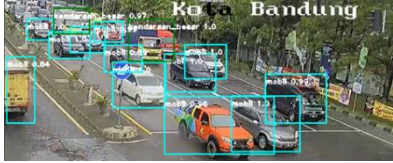



Aktual	Jaringan Skenario Satu	Jaringan Skenario Dua
<i>Frame 2</i>		
Sepeda Motor: 8  Mobil: 5  Kendaraan Besar: 2		
	Sepeda Motor: 8 Mobil: 4 Kendaraan Besar: 2	MtD: 0 MtKn: 6 MtS: 1 MbD: 0 MbKn: 1 MbKr: 0 MbS: 4 KbD: 0 KbKn: 1 KbKn: 0 KbS: 0
	Pada <i>frame</i> citra ini, terdapat satu mobil yang tidak berhasil dideteksi, hal ini dikarenakan mobil tersebut tidak terlihat sepenuhnya.	Pada <i>frame</i> citra ini, terdapat satu motor yang tidak berhasil dideteksi, hal ini dikarenakan objek kendaraan terlalu berdempatan.
<i>Frame 33</i>		
Sepeda Motor: 27  Mobil: 8  Kendaraan Besar: 1		
	Sepeda Motor: 17 Mobil: 6 Kendaraan Besar: 2	MtD: 0 MtKn: 11 MtS: 9 MbD: 0 MbKn: 1 MbKr: 0 MbS: 7 KbD: 0 KbKn: 0 KbKn: 0 KbS: 0
	Pada <i>frame</i> citra ini, terdapat sepuluh sepeda motor, dua mobil, dan satu kendaraan besar yang tidak berhasil dideteksi, hal ini	Pada <i>frame</i> citra ini, terdapat tujuh motor dan satu kendaraan besar yang tidak berhasil dideteksi, hal ini dikarenakan objek kendaraan terlalu berdempatan.

	dikarenakan objek kendaraan terlalu berdempatan.	
<i>Frame 61</i>		
Sepeda Motor: 2  Mobil: 3  Kendaraan Besar: 0		
Sepeda Motor: 2 Mobil: 3 Kendaraan Besar: 0	MtD: 0 MtKn: 0 MtS: 2 MbD: 0 MbKn: 2 MbKr: 0 MbS: 1 KbD: 0 KbKn: 0 KbKs: 0	
	Pada <i>frame</i> citra ini, semua kendaraan berhasil dideteksi.	Pada <i>frame</i> citra ini, semua kendaraan berhasil dideteksi.
<i>Frame 65</i>		
Sepeda Motor: 3  Mobil: 3  Kendaraan Besar: 0		
Sepeda Motor: 2 Mobil: 3 Kendaraan Besar: 0	MtD: 0 MtKn: 1 MtS: 3 MbD: 0 MbKn: 0 MbKr: 0 MbS: 2 KbD: 0	



	<p>Sepeda Motor: 19 (1 mobil dideteksi sebagai motor) Mobil: 11 Kendaraan Besar: 0</p>	<p>MtD: 5 MtKn: 0 MtS: 12 MbD: 3 MbKn: 0 MbKr: 0 MbS: 6 KbD: 0 KbKn: 0 KbKn: 0 KbS: 0</p>
	<p>a. Pada <i>frame</i> citra ini, terdapat delapan sepeda motor, satu mobil, dan satu kendaraan besar yang tidak berhasil dideteksi, hal ini dikarenakan objek kendaraan terlalu berdempatan dan terlalu jauh atau kecil. b. Terdapat juga satu mobil yang dideteksi sebagai sepeda motor, hal ini mungkin terjadi karena, mobil yang seharusnya dideteksi sebagai mobil mempunyai ciri atau fitur yang mirip dengan sepeda motor pada citra ini.</p>	<p>Pada <i>frame</i> citra ini, terdapat sebelas sepeda motor, empat mobil, dan satu kendaraan besar yang tidak berhasil dideteksi, hal ini dikarenakan objek kendaraan terlalu berdempatan dan terlalu jauh atau kecil.</p>
<i>Frame 137</i>		
<p>Sepeda Motor: 10 Mobil: 8 Kendaraan Besar: 1</p>		
	<p>Sepeda Motor: 8 Mobil: 8 Kendaraan Besar: 0</p>	<p>MtD: 2 MtKn: 0 MtS: 0 MbD: 4 MbKn: 0 MbKr: 0 MbS: 5 KbD: 0 KbKn: 0 KbKn: 0 KbS: 0</p>
	<p>a. Pada <i>frame</i> citra ini, terdapat dua sepeda motor dan satu</p>	<p>a. Pada <i>frame</i> citra ini, terdapat delapan sepeda motor dan satu</p>

	<p>kendaraan besar yang tidak berhasil dideteksi, hal ini dikarenakan objek kendaraan terlalu berdempatan dan terlalu jauh atau kecil.</p> <p>b. Terdapat juga satu <i>bounding box</i> motor yang membatasi dua objek motor dalam satu <i>bounding box</i> tersebut, hal ini mungkin terjadi karena kedua objek motor tersebut terlalu berdempatan dan mempunyai fitur atau ciri yang sama seperti satu objek motor.</p>	<p>kendaraan besar yang tidak berhasil dideteksi, hal ini dikarenakan objek kendaraan terlalu berdempatan dan terlalu jauh atau kecil.</p> <p>b. Terdapat juga satu <i>bounding box</i> motor yang membatasi dua objek motor dalam satu <i>bounding box</i> tersebut, hal ini mungkin terjadi karena kedua objek motor tersebut terlalu berdempatan dan mempunyai fitur atau ciri yang sama seperti satu objek motor.</p>
<i>Frame 169</i>		
<p>Sepeda Motor: 1</p> <p>Mobil: 28</p> <p>Kendaraan Besar: 2</p>		
	<p>Sepeda Motor: 1</p> <p>Mobil: 17</p> <p>Kendaraan Besar: 2 (1 mobil dideteksi sebagai kendaraan besar)</p>	<p>MtD: 1</p> <p>MtKn: 0</p> <p>MtS: 0</p> <p>MbD: 0</p> <p>MbKn: 0</p> <p>MbKr: 1</p> <p>MbS: 22</p> <p>KbD: 0</p> <p>KbKn: 0</p> <p>KbKk: 0</p> <p>KbS: 1</p>
	<p>a. Pada <i>frame</i> citra ini, terdapat sebelas mobil yang tidak berhasil dideteksi, hal ini dikarenakan objek kendaraan terlalu berdempatan dan terlalu jauh atau kecil.</p> <p>b. Terdapat juga satu mobil yang dideteksi sebagai kendaraan besar, hal ini mungkin terjadi karena, mobil yang seharusnya dideteksi sebagai mobil mempunyai ciri atau fitur yang mirip dengan kendaraan besar pada citra ini.</p>	<p>Pada <i>frame</i> citra ini, terdapat lima mobil dan satu kendaraan besar yang tidak berhasil dideteksi, hal ini dikarenakan objek kendaraan terlalu berdempatan dan terlalu jauh atau kecil.</p>
<i>Frame 176</i>		

<p>Sepeda Motor: 0</p> <p>Mobil: 20</p>		
<p>Kendaraan Besar: 2</p>	<p>Sepeda Motor: 0</p> <p>Mobil: 12</p> <p>Kendaraan Besar: 2</p>	<p>MtD: 0</p> <p>MtKn: 0</p> <p>MtS: 0</p> <p>MbD: 0</p> <p>MbKn: 0</p> <p>MbKr: 3</p> <p>MbS: 14</p> <p>KbD: 0</p> <p>KbKn: 0</p> <p>KbKn: 0</p> <p>KbS: 2</p>
	<p>Pada <i>frame</i> citra ini, terdapat delapan mobil yang tidak berhasil dideteksi, hal ini dikarenakan objek kendaraan terlalu berdempatan dan terlalu jauh atau kecil.</p>	<p>Pada <i>frame</i> citra ini, terdapat tiga mobil yang tidak berhasil dideteksi, hal ini dikarenakan objek kendaraan terlalu berdempatan dan terlalu jauh atau kecil.</p>
<i>Frame 200</i>		
<p>Sepeda Motor: 0</p> <p>Mobil: 15</p>		
<p>Kendaraan Besar: 2</p>	<p>Sepeda Motor: 0</p> <p>Mobil: 9</p> <p>Kendaraan Besar: 2</p>	<p>MtD: 0</p> <p>MtKn: 0</p> <p>MtS: 0</p> <p>MbD: 0</p> <p>MbKn: 0</p> <p>MbKr: 1</p> <p>MbS: 7</p> <p>KbD: 0</p> <p>KbKn: 0</p> <p>KbKn: 1</p> <p>KbS: 1</p>
	<p>Pada <i>frame</i> citra ini, terdapat enam mobil yang tidak berhasil dideteksi, hal ini dikarenakan objek kendaraan terlalu berdempatan dan terlalu jauh atau kecil.</p>	<p>Pada <i>frame</i> citra ini, terdapat tujuh mobil yang tidak berhasil dideteksi, hal ini dikarenakan objek kendaraan terlalu berdempatan dan terlalu jauh atau kecil.</p>
<b>Total</b>		
<p>Sepeda Motor: 98</p> <p>Mobil: 105</p>	<p>Sepeda Motor: 66</p> <p>Mobil: 76</p> <p>Kendaraan Besar: 10</p>	<p>Sepeda Motor: 63</p> <p>Mobil: 87</p> <p>Kendaraan Besar: 6</p>

Kendaraan Besar: 11		
------------------------	--	--

Keterangan:

- a. Motor Depan : MtD
- b. Motor Kanan : MtKn
- c. Motor Separuh : MtS
- d. Mobil Depan : MbD
- e. Mobil Kanan : MbKn
- f. Mobil Kiri : MbKr
- g. Mobil Separuh : MbS
- h. Kendaraan Besar Depan : KbD
- i. Kendaraan Besar Kanan : KbKn
- j. Kendaraan Besar Kiri : KbKr
- k. Kendaraan Besar Separuh : KbS

Berdasarkan Tabel 4.1 akan dibuat *Confusion Matrix* hasil deteksi kendaraan dari jaringan skenario satu pada *dataset* penulis seperti pada Tabel 4.2.

Tabel 4.2 Tabel *Confusion Matrix* hasil deteksi kendaraan jaringan skenario satu pada *dataset* penulis

<i>Confusion Matrix</i>		Aktual		
		Motor	Mobil	Kendaraan Besar
Prediksi	Motor	66	1	-
	Mobil	-	76	1
	Kendaraan Besar	-	-	10
	Tidak Terdeteksi	32	29	1

Berdasarkan Tabel 4.2 dapat dicari nilai *True Positive*, *False Positive*, *False Negative*, *Precision*, *Recall*, dan *F-1 Score* dari jaringan skenario satu. Berikut Persamaan 4.1 sampai 4.6 yang memaparkan persamaan untuk mencari nilai-nilai tersebut.

$$True\ Positive = 66 + 76 + 10 = 151 \quad (4.1)$$

$$False\ Positive = 1 + 1 = 2 \quad (4.2)$$

$$False\ Negative = 32 + 29 + 1 = 62 \quad (4.3)$$

$$Precision = \frac{TP}{TP + FP} = \frac{152}{152 + 2} = 0,99 \quad (4.4)$$

$$Recall = \frac{TP}{TP + FN} = \frac{152}{152 + 62} = 0,71 \quad (4.5)$$

$$F1\ Score = 2 \times \left( \frac{(Precision \times Recall)}{(Precision + Recall)} \right) = 2 \times \left( \frac{(0,99 \times 0,71)}{(0,99 + 0,71)} \right) = 0,83 \quad (4.6)$$

Didapatkan nilai *True Positive* sebesar 151 yang artinya terdapat 151 kendaraan yang berhasil dideteksi pada kelas kendaraan yang benar, *False Positive* sebesar 2 yang artinya terdapat 2 kendaraan yang dideteksi secara salah, *False Negative* sebesar 62 yang artinya terdapat 62 kendaraan yang tidak berhasil dideteksi, *Precision* sebesar 0,99 yang artinya tingkat deteksi kendaraan ke dalam kelas yang benar adalah sebesar 0,99, *Recall* sebesar 0,71 yang artinya tingkat deteksi kendaraan dalam mendeteksi keseluruhan objek secara benar adalah sebesar 0,71 dan *F1-Score* yang bernilai 0,83. Dapat dilihat bahwa rata-rata nilai pengujian performa jaringan skenario satu dengan menggunakan *dataset* penulis mempunyai nilai yang tinggi.

Selanjutnya akan dibuat juga *Confusion Matrix* hasil deteksi kendaraan dari jaringan skenario dua pada *dataset* penulis seperti pada Tabel 4.3. Pada pengujian ini, jumlah kelas dari kendaraan yang sama, misal motor\_depan, motor\_kanan, dan motor\_separuh akan dijumlahkan ke dalam satu kelas yang sama yaitu kelas sepeda motor, hal yang sama juga dilakukan pada kelas mobil dan kendaraan besar.



Tabel 4.3 Tabel *Confusion Matrix* hasil deteksi kendaraan jaringan skenario dua pada *dataset* penulis

<i>Confusion Matrix</i>		Aktual		
		Motor	Mobil	Kendaraan Besar
Prediksi	Motor	63	-	-
	Mobil	-	87	-
	Kendaraan Besar	-	-	6
	Tidak Terdeteksi	35	18	5

Berdasarkan Tabel 4.3 dapat dicari nilai *True Positive*, *False Positive*, *False Negative*, *Precision*, *Recall*, dan *F-1 Score* dari jaringan skenario dua. Berikut Persamaan 4.7 sampai 4.12 yang memaparkan persamaan untuk mencari nilai-nilai tersebut.

$$\text{True Positive} = 63 + 87 + 6 = 156 \quad (4.7)$$

$$\text{False Positive} = 0 \quad (4.8)$$

$$\text{False Negative} = 35 + 18 + 5 = 58 \quad (4.9)$$

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{156}{156 + 0} = 1 \quad (4.10)$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{156}{156 + 58} = 0,73 \quad (4.11)$$

$$\text{F1 Score} = 2 \times \left( \frac{(\text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})} \right) = 2 \times \left( \frac{(1 \times 0,73)}{(1 + 0,73)} \right) = 0,84 \quad (4.12)$$

Didapatkan nilai *True Positive* sebesar 156 yang artinya terdapat 156 kendaraan yang berhasil dideteksi kendaraan yang benar, *False Positive* sebesar 0 yang artinya terdapat 0 kendaraan yang dideteksi secara salah, *False Negative* sebesar 58 yang artinya terdapat 58

kendaraan yang tidak berhasil dideteksi, *Precision* sebesar 1 yang artinya tingkat deteksi kendaraan ke dalam kelas yang benar adalah sebesar 1, *Recall* sebesar 0,73 yang artinya tingkat deteksi kendaraan dalam mendeteksi keseluruhan objek secara benar adalah sebesar 0,73 dan *F1 Score* yang bernilai 0,84. Dapat dilihat bahwa rata-rata nilai pengujian performa jaringan skenario dua dengan menggunakan *dataset* penulis mempunyai nilai yang tinggi.

#### 4.2.2 Rangkuman Pengujian Performa Jaringan

##### Rangkuman Total Kendaraan yang Berhasil Dideteksi

Untuk memudahkan proses perbandingan pengujian performa jaringan, dibuat rangkuman total kendaraan yang berhasil dideteksi oleh model pada masing-masing *dataset* seperti pada Tabel 4.4.

Tabel 4.4 Tabel rangkuman perbandingan total deteksi kendaraan pada masing-masing bobot jaringan

<i>Dataset</i>	Bobot Jaringan	Aktual			Prediksi			Rata-rata akurasi		
		Sepeda Motor	Mobil	Kendaraan Besar	Sepeda Motor	Mobil	Kendaraan Besar	Sepeda Motor	Mobil	Kendaraan Besar
Penulis	Skenario Satu	98	105	11	66	76	10	0,67	0,72	0,91
	Skenario Dua	98	105	11	63	87	6	0,64	0,83	0,55
Rata-rata Deteksi								0,46	0,57	0,74

Dari Tabel 4.4, dihitung nilai rata-rata akurasi deteksi objek sepeda motor, mobil, dan kendaraan besar pada masing-masing jaringan dan *dataset*. Dapat dilihat bahwa kendaraan mobil merupakan objek yang paling dapat dideteksi di antara semua dataset dan bobot jaringan, dengan nilai rata-rata deteksi yang mencapai 0,57 dari total objek aktualnya. Hal yang sama juga terjadi pada pendeteksian objek kendaraan besar dengan

nilai rata-rata deteksi yang mencapai 0,74 dari total objek aktualnya. Hal ini mungkin terjadi karena objek mobil dan kendaraan besar memiliki ukuran yang lebih besar daripada motor dan juga memiliki fitur fisik yang hampir mirip antara satu sama lain dalam berbagai sudut pandang. Selain itu juga kedua kendaraan ini apabila tertutup oleh kendaraan lain, sebagian besar bagiannya masih dapat terlihat dan masih bisa dicirikan. Ketiga alasan inilah yang membuat kedua objek ini lebih mudah untuk dideteksi oleh jaringan. Berbeda dengan sepeda motor yang memiliki nilai rata-rata deteksi sebesar 0,46 dari total objek aktualnya. Hal ini dikarenakan objek sepeda motor memiliki ukuran yang kecil dan variasi fitur fisik yang kebanyakan tidak mirip antara satu sama lain.

Selanjutnya, akan dibuat Tabel 4.5 yang memuat rangkuman performa deteksi jaringan pada masing-masing bobot jaringan dan *dataset*.

Tabel 4.5 Tabel rangkuman perbandingan performa jaringan pada masing-masing bobot jaringan

<i>Dataset</i>	Bobot Jaringan	Performa Jaringan					
		TP	FP	FN	<i>Precision</i>	<i>Recall</i>	<i>F1 Score</i>
Penulis	Skenario Satu	151	2	62	0,99	0,71	0,83
	Skenario Dua	156	0	58	1	0,73	0,84

Pada pengujian dengan bobot jaringan skenario satu dan skenario dua pada dataset penulis, ditemukan perbedaan nilai performa jaringan yang tidak terlalu signifikan yaitu peningkatan nilai *Precision* sebesar 0,1, nilai *Recall* sebesar 0,2, dan nilai *F1 Score* sebesar 0,1. Nilai TP tertinggi dimiliki oleh bobot jaringan skenario dua, diikuti oleh bobot jaringan skenario satu. Untuk nilai FP tertinggi dimiliki oleh bobot jaringan skenario satu, diikuti oleh bobot jaringan skenario dua. Sedangkan untuk nilai FN tertinggi dimiliki oleh bobot jaringan skenario satu, diikuti oleh bobot jaringan skenario dua. Dari sini dapat dilihat bahwa bobot jaringan skenario dua merupakan bobot jaringan yang paling mampu untuk mendeteksi kendaraan pada *dataset* penulis.

## BAB V

### KESIMPULAN DAN SARAN

#### 5.1 Kesimpulan

Dari penelitian yang dilakukan, terdapat beberapa poin kesimpulan di antaranya adalah:

- a. Telah dibangun dua jaringan deteksi kendaraan algoritma YOLO v3 dengan menggunakan skenario pelabelan kelas yang berbeda berdasarkan sudut pengambilan citra. Skenario pelabelan satu memiliki total kelas sebesar tiga kelas yaitu sepeda motor, mobil, dan kendaraan besar dan skenario pelabelan dua memiliki total kelas sebesar sebelas kelas yaitu sepeda motor dari kanan, sepeda motor dari depan, sepeda motor separuh, mobil dari kanan, mobil dari depan, mobil dari kiri, mobil separuh, kendaraan besar kanan, kendaraan besar kiri, kendaraan besar depan, dan kendaraan besar separuh.
- b. Dilakukan pengujian menggunakan kedua bobot jaringan tersebut pada sepuluh *frame test set* penulis, didapatkan nilai performa jaringan yang cukup tinggi dengan nilai *Precision* yang sempurna, nilai *Recall* di atas 70%, dan *F1 Score* di atas 80%.
- c. Dari kedua pengujian bobot jaringan tersebut, diketahui bahwa objek mobil dan kendaraan besar merupakan objek yang paling berhasil dapat dideteksi oleh ketiga jaringan tersebut. Hal ini disebabkan oleh ukuran dan fitur fisik objek mobil dan kendaraan besar yang hampir mirip antara satu sama lain.

#### 5.2 Saran

Untuk pengembangan dari penelitian ini, penulis memberikan beberapa saran sebagai berikut:

- a. Memperbanyak total citra dan variasinya, variasi yang dimaksud seperti sudut pengambilan gambar, resolusi citra dan waktu pengambilan citra (siang atau malam hari). Hal ini ditujukan agar jaringan yang akan dibangun dapat mengenali lebih banyak fitur dari kendaraan dalam berbagai macam situasi.
- b. Menggunakan algoritma deteksi objek lainnya, seperti YOLO v4, YOLO v5, SSD dan lain-lain. Dengan digunakannya algoritma deteksi objek yang lain diharapkan terdapat perbedaan yang cukup signifikan.



## DAFTAR PUSTAKA

- Alderliesten, K. (2020). YOLOv3 — Real-time object detection. <https://medium.com/analytics-vidhya/yolov3-real-time-object-detection-54e69037b6d0#:~:text=YOLOv3 uses a variant of,106-layer fully convolutional architecture.>
- Amwin, A. (2021). Deteksi Dan Klasifikasi Kendaraan Berbasis Algoritma You Only Look Once ( Yolo ). Universitas Islam Indonesia.
- Andika, M. (2018). Karakteristik Kendaraan pada Ruas Jalan Nasional Sumatera Utara. Universitas Sumatera Utara.
- Aningtiyas, P. R., Sumin, A., & Wirawan, S. (2020). Pembuatan Aplikasi Deteksi Objek Menggunakan TensorFlow Object Detection API dengan Memanfaatkan SSD MobileNet V2 Sebagai Model Pra-Terlatih. *Ilmiah KOMPUTASI*, 19(September), 421–430.
- Assidhiqi, F. (2021). Pengembangan Sistem Deteksi Hunian Parkir Menggunakan Metode Convolutional Neural Network.
- Bin Issa, R., Das, M., Rahman, M. S., Barua, M., Rhaman, M. K., Ripon, K. S. N., & Alam, M. G. R. (2021). Double Deep Q-Learning and Faster R-CNN-Based Autonomous Vehicle Navigation and Obstacle Avoidance in Dynamic Environment. *Sensors*, 21(4). <https://doi.org/10.3390/s21041468>
- Budiarti, A. (2006). Bab 2 landasan teori. *Aplikasi Dan Analisis Literatur Fasilkom UI*, 4–25.
- Departemen Pekerjaan Umum. (2005). *Modul Rekayasa Lalu Lintas*. Jakarta : Departemen Pekerjaan Umum.
- Ding, Xiangwu, Yang, & Ruidi. (2019). Vehicle and Parking Space Detection Based on Improved YOLO Network Model. *Journal of Physics: Conference Series*, 1325, 012084. <https://doi.org/10.1088/1742-6596/1325/1/012084>
- Direktorat Jenderal Bina Marga Kementerian Pekerjaan Umum dan Perumahan Rakyat. (2016). *Prosedur Pemeliharaan Jalan SOP/UPM/DJBM-12* (pp. 1–20).
- Fandisyah, A. F., Iriawan, N., & Winahju, W. S. (2021). Deteksi Kapal di Laut Indonesia Menggunakan YOLOv3. *Jurnal Sains Dan Seni ITS*, 10(1), D25–D32.
- Gao, H., Cheng, B., Wang, J., Li, K., Zhao, J., & Li, D. (2018). Object Classification Using CNN-Based Fusion of Vision and LIDAR in Autonomous Vehicle Environment. *IEEE*

- Transactions on Industrial Informatics, 14(9), 4224–4231.  
<https://doi.org/10.1109/TII.2018.2822828>
- Gunawan, D. N. (2019). Kemacetan Lalu Lintas Kendaraan Bermotor dan Pengaruhnya terhadap Perekonomian Kota-Kota Besar di Indonesia. In Digital Repository Universitas Jember (Issue September 2019). Universitas Jember.
- Gunawan, Zuama, R. A., Saepulrohman, R., Sulaiman, H., & Ghanil, M. A. (2019). Deteksi Otomatis Kanker Payudara Menggunakan Metode Morphological Reconstruction Dengan K-Means Clustering. *Jurnal Swabumi*, 7(1), 67–71.
- Hammam, H., Asyhar, A., Wibowo, S. A., & Budiman, G. (2020). Implementasi Dan Analisis Performansi Metode You Only Look Once (Yolo) Sebagai Sensor Pornografi Pada Video Implementation and Performance Analysis of You Only Look Once (Yolo) Method As Porn Censorship in Video. *E-Proceeding of Engineering*, 7(2), 3631–3638.
- Hartanto, S. (2020). PERBAIKAN DETEKSI ALAT KLASIFIKASI KENDARAAN OTOMATIS MENGGUNAKAN SENSOR INFRAMERAH TAMBAHAN. *EPIC (Journal of Electrical Power, Instrumentation and Control)*, 3(2), 189–195.  
<https://doi.org/10.32493/epic.v3i2.7737>
- Hasairin, A., & Siregar, R. (2018). Deteksi Kandungan Gas Karbon Monoksida (Co) Hubungan Dengan Kepadatan Lalu-Lintas Di Medan Sunggal, Kota Medan. *Jurnal Biosains*, 4(1), 62. <https://doi.org/10.24114/jbio.v4i1.9841>
- Herawati, Y. (2021). Enggak Nyangka, Segini Jumlah Kendaraan di Indonesia. [https://www.viva.co.id/otomotif/1401896-enggak-nyangka-segini-jumlah-kendaraan-di-indonesia?page=all&utm\\_medium=all-page](https://www.viva.co.id/otomotif/1401896-enggak-nyangka-segini-jumlah-kendaraan-di-indonesia?page=all&utm_medium=all-page)
- Hidayati, Q. (2017). Kendali Lampu Lalu Lintas dengan Deteksi Kendaraan Menggunakan Metode Blob Detection. *Jurnal Nasional Teknik Elektro Dan Teknologi Informasi (JNTETI)*, 6(2). <https://doi.org/10.22146/jnteti.v6i2.318>
- Ilahiyah, S., & Nilogiri, A. (2018). Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network. *JUSTINDO (Jurnal Sistem Dan Teknologi Informasi Indonesia)*, 3(2), 49–56.
- Irfan, M., Ardi Sumbodo, B. A., & Candradewi, I. (2017). Sistem Klasifikasi Kendaraan Berbasis Pengolahan Citra Digital dengan Metode Multilayer Perceptron. *IJEIS (Indonesian Journal of Electronics and Instrumentation Systems)*, 7(2), 139.  
<https://doi.org/10.22146/ijeis.18260>

- Jain, V., Verma, Y. K., & Jain, V. K. (2018). Automatic Traffic Light Control System By Using Digital Image Processing. 8(7), 73–76. <https://doi.org/10.9790/9622-0807037376>
- Jiang, P., Ergu, D., Liu, F., Cai, Y., & Ma, B. (2022). A Review of Yolo Algorithm Developments. *Procedia Computer Science*, 199, 1066–1073. <https://doi.org/10.1016/j.procs.2022.01.135>
- Joheras. (2019). CLoDSA. <https://github.com/joheras/CLoDSA>
- Julianto, E. N. (2010). Hubungan Antara Kecepatan, Volume Dan Kepadatan Lalu Lintas Ruas Jalan Siliwangi Semarang. *Jurnal Teknik Sipil Dan Perencanaan*, 12(2), 151–160.
- Kathuria, A. (2018). How to implement a YOLO (v3) object detector from scratch in PyTorch: Part 1. <https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/>
- Kemajou, V. N., Bao, A., & Germain, O. (2019). Wellbore schematics to structured data using artificial intelligence tools. *Proceedings of the Annual Offshore Technology Conference*, 2019-May(May). <https://doi.org/10.4043/29490-ms>
- Kim, S. D. (2020). Situation-cognitive traffic light control based on object detection using YOLO algorithm. *International Journal of Computational Vision and Robotics*, 10(2), 133–142. <https://doi.org/10.1504/IJCVR.2020.105682>
- Koeh, K. E. (2020). Confusion Matrix for Object Detection. <https://towardsdatascience.com/confusion-matrix-and-object-detection-f0c9cb634157>
- Leriansyah, M. (2020). DETEKSI DAN PERHITUNGAN KENDARAAN UNTUK MENGETAHUI ARUS KEPADATAN LALU LINTAS SECARA OTOMATIS MENGGUNAKAN YOLO-V3. Universitas Islam Indonesia.
- Lina, Q. (2019). Apa itu Convolutional Neural Network? <https://medium.com/@166111110/apa-itu-convolutional-neural-network-836f70b193a4>
- Liu, P. L. (2020). Deep-Learning based Object Detection in Crowded Scenes. <https://towardsdatascience.com/deep-learning-based-object-detection-in-crowded-scenes-1c9fd9bd7bc4>
- Lohia, A., Kadam, K. D., Joshi, R. R., & M. Bongale, D. A. (2021). Bibliometric Analysis of One-stage and Two-stage Object Detection. *Library Philosophy and Practice (e-Journal)*, February.
- Ma'ali, A. M. (2019). Rancang Bangun Sistem Pengendali Lampu Lalu Lintas Berdasarkan Pengenalan Citra Digital Kendaraan Menggunakan Metode Faster R-Cnn.



- Nasriyati, T., & Utami, S. (2018). Morfologi Talus Lichen *Dirinaria Picta* (Sw.) Schaer. Ex Clem pada Tingkat Kepadatan Lalu Lintas yang Berbeda di Kota Semarang. *Jurnal Akademika Biologi*, 7(4), 20–27.
- Oudat, E., Mousa, M., & Claudel, C. (2015). Vehicle Detection and Classification Using Passive Infrared Sensing. 2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems, 443–444. <https://doi.org/10.1109/MASS.2015.62>
- Pantha, N. (2019). Understanding Object Detection using YOLO. <https://medium.com/mpercept-academy/understanding-object-detection-using-yolo-bf2d0f75747a>
- Parmar, T. (2021). Vehicular Characteristics in Traffic Engineering. <https://www.civilgiant.com/vehicular-characteristics/>
- Pratama, Y., & Rasywir, E. (2021). Eksperimen Penerapan Sistem Traffic Counting dengan Algoritma YOLO (You Only Look Once) V.4. *Jurnal Media Informatika Budidarma*, 5(4), 1438–1446. <https://doi.org/10.30865/mib.v5i4.3309>
- Rajagede, R. A. (2021). TRANSFER LEARNING: SOLUSI DEEP LEARNING DENGAN DATA SEDIKIT. <https://structilmy.com/blog/2021/01/31/transfer-learning-solusi-deep-learning-dengan-data-sedikit/>
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (n.d.). You Only Look Once: Unified, Real-Time Object Detection. Retrieved February 4, 2022, from <http://pjreddie.com/yolo/>
- Redmon, J., & Farhadi, A. (2018). YOLO v.3. Tech Report, 1–6. <https://pjreddie.com/media/files/papers/YOLOv3.pdf>
- Robins, C., & Krok, J. (2019). Object Detection with Satellite Imagery: Using YOLO to Detect and Localize Objects in Aerial Satellite Imagery.
- Rosebrock, A. (2016). Intersection over Union (IoU) for object detection. <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>
- Said, L. B., H, S. M., & Sriwati. (2019). Pengaruh Pertumbuhan Kendaraan Dan Kapasitas Jalan Terhadap Kemacetan Di Ruas Jalan Perintis Kemerdekaan. *Fly Over*, 3(1), 79–86.
- Saputra, A. D. (2017). Studi Tingkat Kecelakaan Lalu Lintas Jalan di Indonesia Berdasarkan Data KNKT (Komite Nasional Keselamatan Transportasi) Dari Tahun 2007-2016. *Warta Penelitian Perhubungan*. <https://doi.org/10.1016/j.injury.2011.11.002>

- Shinde, M. Y., & Powar, M. H. (2017). Intelligent Traffic Light Controller Using IR Sensors for Vehicle Detection. *Iarjset*, 4(2), 88–90. <https://doi.org/10.17148/iarjset/ncetete.2017.28>
- Soman, R., & Radhakrishnan, P. K. (2018). Traffic Light Control and Violation Detection Using Image Processing. 08(4), 23–27.
- Tamara, S., & Sasana, H. (2017). Analisis Dampak Ekonomi Dan Sosial Akibat Kemacetan Lalu Lintas Di Jalan Raya Bogor-Jakarta. *Jurnal REP (Riset Ekonomi Pembangunan)*, 2(2), 185–196. <https://doi.org/10.31002/rep.v2i3.529>
- Tang, C., Ling, Y., Yang, X., Jin, W., & Zheng, C. (2018). Multi-view object detection based on deep learning. *Applied Sciences (Switzerland)*, 8(9). <https://doi.org/10.3390/app8091423>
- Tiwari, N. (2020). YOLOv3 Custom Object Detection with Transfer Learning. *TFUG Mumbai Weekly*. <https://tiwarinitin1999.medium.com/yolov3-custom-object-detection-with-transfer-learning-47186c8f166d>
- Tzotalin. (2015). *LabelImg*. <https://github.com/tzotalin/labelImg>
- Velazquez-Pupo, R., Sierra-Romero, A., Torres-Roman, D., Shkvarko, Y. V, Santiago-Paz, J., Gómez-Gutiérrez, D., Robles-Valdez, D., Hermosillo-Reynoso, F., & Romero-Delgado, M. (2018). Vehicle Detection with Occlusion Handling, Tracking, and OC-SVM Classification: A High Performance Vision-Based System. *Sensors*, 18(2). <https://doi.org/10.3390/s18020374>
- Zou, Z., Shi, Z., Guo, Y., & Ye, J. (2019). Object Detection in 20 Years: A Survey. 1–39. <http://arxiv.org/abs/1905.05055>