DETEKSI DAN KLASIFIKASI KENDARAAN BERBASIS ALGORITMA *YOU ONLY LOOK ONCE* (YOLO)

Disusun Oleh:

N a m a : Aldhiyatika Amwin

NIM : 17523176

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2021

HALAMAN PENGESAHAN DOSEN PEMBIMBING

DETEKSI DAN KLASIFIKASI KENDARAAN BERBASIS ALGORITMA *YOU ONLY LOOK ONCE* (YOLO)

TUGAS AKHIR

Disusun Oleh:

N a m a : Aldhiyatika Amwin

NIM : 17523176

Yogyakarta, 6 Agustus 2021

Pembimbing,

(Septia Rani, S.T., M.Cs.)

HALAMAN PENGESAHAN DOSEN PENGUJI

DETEKSI DAN KLASIFIKASI KENDARAAN BERBASIS ALGORITMA *YOU ONLY LOOK ONCE* (YOLO)

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,

Tim Penguji

Septia Rani, S.T., M.Cs.

Anggota 1

Irving Vitra Paputungan, S.T., M.Sc., Ph.D.

Anggota 2

Sheila Nurul Huda, 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 : Aldhiyatika Amwin

NIM : 17523176

Tugas akhir dengan judul:

DETEKSI DAN KLASIFIKASI KENDARAAN BERBASIS ALGORITMA *YOU ONLY LOOK ONCE* (YOLO)

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, 6 Agustus 2021

(Aldhiyatika Amwin)

HALAMAN PERSEMBAHAN

Tugas akhir ini penulis persembahkan kepada kedua orang tua, kakak, dan adik penulis yang tidak pernah berhenti mendukung, memberikan nasihat, mendo'akan penulis selama pengerjaan tugas akhir ini. Dan juga tugas akhir ini penulis persembahkan untuk diri penulis sendiri yang sudah berhasil menyelesaikan tugas akhir.

HALAMAN MOTO

"Be nice or be good"

"Take your time, make mistakes, and explore. Growth is not always glamorous"

"Rome wasn't built in a day, but they were laying bricks every hour. You don't have to do it all today. Just lay a brick"

(James Clear)

KATA PENGANTAR

Bismillahirrahmanirrahim.

Assalamu'alaikum Warahmatullahi Wabarakatuh

Alhamdulillahirobbil'alamin, segala puji dan syukur penulis panjatkan kepada Allah SWT yang selalu memberikan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan laporan Tugas Akhir yang berjudul "**Deteksi dan Klasifikasi Kendaraan Berbasis Algoritma *You Only Look Once* (YOLO)**". Shalawat serta salam kepada junjungan Nabi Muhammad SAW, sebagai suri tauladan bagi setiap umat manusia yang telah membawa umat manusia dari zaman jahiliyah ke zaman yang penuh dengan ilmu pengetahuan seperti saat ini.

Laporan Tugas Akhir ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata-1 (S1) Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia. Penulis menyadari bahwa dalam penulisan laporan ini tidak terlepas dari bimbingan, do'a, dukungan, dan motivasi dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Allah SWT yang selalu ada disaat yang lain tidak ada.
2. Kedua orang tua bapak Darwin dan ibu Tawar Rami, untuk segala do'a, dukungan, dan kepercayaan yang sudah diberikan kepada penulis untuk menempuh pendidikan di Yogyakarta.
3. Kakak Nabila Lathifa Amwin dan adik Arwa Husna Amwin yang memberikan dukungan, do'a dan semangat kepada penulis.
4. Bapak Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia.
5. Bapak Hendrik, S.T., M.Eng., selaku Ketua Jurusan Informatika Universitas Islam Indonesia.
6. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., selaku Ketua Program Studi Informatika Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia.
7. Ibu Septia Rani, S.T., M.Cs., selaku dosen pembimbing tugas akhir yang telah membantu penulis dalam memberikan pengarahan serta masukan dalam mengerjakan tugas akhir.

8. Bapak Hanson Prihantoro Putro, S.T., M.T., selaku dosen pembimbing akademik yang telah memberikan bimbingan kepada penulis selama menjalani perkuliahan di Informatika.
9. Bapak dan Ibu dosen Program Studi Informatika yang sangat berjasa dalam memberikan ilmu yang bermanfaat kepada penulis, semoga semua ilmu yang telah diberikan menjadi amal jariyah di akhirat kelak.
10. Annisa Zahra, Annisa Nauli Hasibuan, dan Muhammad Zikri Khatami Sagala yang berbagi kesenangan dan kesedihan bersama.
11. Tri Handayani dan Syarifah Elza Ramadhania, teman se-perbimbingan yang saling membantu, memberikan dukungan.
12. Seluruh pihak yang tidak bisa penulis sebutkan satu persatu, terima kasih atas semua dukungannya.

Semoga semua ilmu, do'a, dukungan, dan bimbingan yang telah diberikan kepada penulis mendapat balasan yang lebih baik dari Allah SWT. Penulis menyadari bahwa laporan Tugas Akhir ini masih banyak kekurangan, oleh karena itu saran dan kritik yang membangun akan diterima dengan baik. Semoga laporan Tugas Akhir ini dapat bermanfaat bagi semua pihak.

Yogyakarta, 6 Agustus 2021

(Aldhiyatika Amwin)

SARI

Lalu lintas menjadi salah satu permasalahan yang paling menantang dan sulit dalam melakukan manajemen kota terutama pada negara berkembang. Kemacetan lalu lintas menjadi permasalahan yang serius secara global. Beberapa tahun belakangan, penelitian mengenai kecerdasan buatan seperti deteksi objek dapat memudahkan para peneliti untuk mengenali objek yang terdapat pada sebuah gambar. Dalam penelitian ini penulis menggunakan algoritma YOLO untuk melakukan pendeteksian dan klasifikasi kendaraan. Penelitian ini menggunakan dataset sebanyak 531 gambar dengan lima kelas yaitu mobil, sepeda motor, truk, bus, dan becak. Hasil penelitian menunjukan algoritma *You Only Look Once* (YOLO) dapat mengenali objek pada video CCTV yang dipasang di Simpang Air Mancur-Imanuel Kota Medan dengan menggunakan *pre-trained weights* yang telah dilatih sendiri dengan nilai *mean Average Precision* (mAP) sebesar 99,35%.

Kata kunci: Deteksi Kendaraan, Klasifikasi, *You Only Look Once*, CCTV

GLOSARIUM

Bounding box	kotak pembatas yang berisi sebuah objek.
CCTV	*Closed Circuit Television*.
Confidence score	nilai keyakinan yang menyatakan model tersebut.
Dataset	kumpulan data.
Pre-trained Weights	model *weights* yang telah di-*training* sebelumnya pada jaringan YOLO untuk suatu dataset.
Transfer Learning	suatu teknik dengan menggunakan model yang telah di-*training* sebelumnya yang dapat digunakan untuk klasifikasi dataset baru tanpa harus melakukan *training* data dari awal.
YOLO	salah satu algoritma dalam mendeteksi objek.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Masalah mengenai lalu lintas menjadi salah satu permasalahan yang paling menantang dan sulit dalam melakukan manajemen kota terutama pada negara berkembang. Secara global, kemacetan lalu lintas menjadi permasalahan yang serius. Permasalahan ini menjadi menarik perhatian dari beberapa kalangan seperti ahli tata kota, pemerintahan, teknisi, dan para peneliti untuk mencari solusi mengenai kemacetan. (Nurhawanti, 2019). Sebagian kota di Indonesia sudah menggunakan CCTV (Closed Control Television) untuk melakukan pemantauan arus lalu lintas. Dalam melakukan proses pemantauan arus lalu lintas, CCTV akan melakukan ekstraksi informasi dari gambar seperti kecepatan laju kendaraan, kemacetan, bentuk dan jenis kendaraan, nomor kendaraan, dan pelanggaran atau kecelakaan lalu lintas (Harahap et al., 2019).

Beberapa tahun belakangan, penelitian mengenai kecerdasan buatan seperti deteksi objek dapat memudahkan para peneliti untuk mengenali objek yang terdapat pada sebuah gambar. Deteksi objek adalah salah satu bidang pada *computer vision*. *Computer vision* merupakan ilmu yang mempelajari bagaimana komputer dapat melihat serta melakukan analisa pada suatu objek yang terdapat pada sebuah gambar.

Penelitian tentang pemantauan lalu lintas sudah banyak diusulkan oleh para peneliti dengan menggunakan beberapa pendekatan, seperti yang diusulkan oleh (Alamsyah, 2017) penelitian menggunakan model *Histogram of Oriented Gradient* (HOG) dengan *Support Machine Vector* (SVM). Penelitian dengan menggunakan metode deep learning juga diusulkan oleh beberapa peneliti untuk melakukan pemantauan lalu lintas dengan menggunakan pendekatan seperti, *Faster* R-CNN (Arinaldi, Pradana, & Gurusinga, 2018), *Single Shoot Multibox Detector* (SSD) (Liu et al., 2016), *Convolutional Neural Network* (CNN) (Fadlia & Kosasih, 2019), serta *You Only Look Once* (YOLO) (Fachrie, 2020).

Metode *You Only Look Once* (YOLO) menjadi salah satu metode yang cepat dan akurat dalam melakukan pendeteksian objek. Metode ini mampu melakukan deteksi objek hingga 2 kali lebih cepat daripada algoritma yang lain. Pada penelitian ini, penulis menggunakan metode YOLOv3 karena memiliki beberapa peningkatan dalam mendeteksi objek dan nilai akurasi yang lebih tinggi daripada versi sebelumnya. Dengan menggunakan metode YOLO,

diharapkan penelitian ini dapat memberikan hasil yang baik untuk mendeteksi dan mengklasifikasi kendaraan melalui rekaman CCTV lalu lintas yang diapatkan dari *Website* ATCS Dishub Kota Medan pada Simpang Air Mancur-Imanuel.

1.2 Rumusan Masalah

Dari permasalahan latar belakang di atas, perumusan masalah untuk penelitian yang akan dilakukan adalah:

a. Bagaimana penerapan algoritma *You Only Look Once* (YOLO) untuk mendeteksi dan mengklasifikasi kendaraan melalui rekaman CCTV lalu lintas?
b. Bagaimana hasil pengujian *You Only Look Once* (YOLO) untuk mendeteksi kendaraan melalui rekaman CCTV lalu lintas?

1.3 Batasan Masalah

Batasan masalah pada penelitian ini adalah:

a. Menggunakan algoritma *You Only Look Once* (YOLO) untuk mendeteksi kendaraan.
b. Data pada penelitian ini menggunakan data yang direkam melalui Website ATCS Dishub Kota Medan pada Simpang Air Mancur-Imanuel.
c. Kelas-kelas kendaraan pada penelitian ini adalah mobil, sepeda motor, truk, becak, dan bus.
d. Menggunakan bahasa pemrograman Python.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk mendeteksi dan mengklasifikasi kendaraan menggunakan algoritma *You Only Look Once* (YOLO) melalui rekaman CCTV lalu lintas.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah:

a. Dapat memudahkan pihak yang berwewenang dalam melakukan pemantauan arus lalu lintas.
b. Hasil dari penelitian ini dapat dijadikan referensi untuk penelitian di masa depan dalam melakukan pendeteksian kendaraan menggunakan algoritma *You Only Look Once* (YOLO).

1.6 Sistematika Penulisan

Sistematika penelitian ini adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini berisi latar belakang mengenai permasalahan yang mendasari penelitian. Yang meliputi latar belakang, rumusan masalah penelitian, batasan masalah penelitian, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

BAB II LANDASAN TEORI

Bab ini berisi teori-teori yang menjadi referensi dalam melakukan penelitian yang diperoleh dari berbagai sumber. Teori-teori yang dijelaskan pada Bab II ini meliputi kendaraan, video, deteksi objek, *deep learning*, *You Only Look Once*, OpenCV.

BAB III METODOLOGI

Bab ini berisi tahapan-tahapan penelitian sebagai acuan untuk mencapai solusi yang ditawarkan pada penelitian ini. Bab ini terdiri dari studi literatur, perencanaan, pengumpulan data, implementasi, pengujian, dan analisis.

BAB IV HASIL DAN PEMBAHASAN

Bab ini berisi hasil dan pembahasan dari setiap proses dalam sistem.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan mengenai hasil dari penelitian yang telah sesuai dengan tujuan penelitian serta saran untuk peneliti di masa depan berdasarkan hasil penelitian.

BAB II
LANDASAN TEORI

2.1 Landasan Teori

2.1.1 Kendaraan

Berdasarkan Undang-Undang Nomor 22 Tahun 2009 tentang Lalu Lintas dan Angkutan Jalan, kendaraan adalah suatu sarana angkut di jalan yang terdiri atas kendaraan bermotor dan kendaraan tidak bermotor.

Dari pengertian tersebut, kendaraan dibedakan menjadi:

a. Kendaraan bermotor, adalah setiap kendaraan yang digerakkan oleh peralatan mekanik berupa mesin selain kendaraan yang berjalan di atas rel.
b. Kendaraan tidak bermotor, adalah setiap kendaraan yang digerakkan oleh tenaga manusia dan/atau hewan.

2.1.2 Video

Video adalah bentuk dinamis dari gambar statis. Video bisa disebut juga serangkaian gambar statis pada urutan tertentu. Setiap gambar disebut dengan *frame*. *Frame* akan diproyeksikan secara terus-menerus ke layar dengan kecepatan konstan, sehingga akan menghasilkan efek dinamik yang mempengaruhi tingkat kehadiran visual. Video juga dapat dikategorikan sebagai analog dan digital. Karena video adalah serangkaian gambar statis, jadi kita dapat memproses video seperti kita memproses gambar. Gambar adalah kumpulan piksel dalam ruang warna yang berbeda. Gambar lengkap sebagai set yang terdiri dari sampel kecil. Sampel-sampel ini yang disebut piksel. Piksel adalah elemen terkecil dalam gambar (Salim, 2020).

2.1.3 Deteksi Objek

Deteksi objek menentukan keberadaan sebuah objek, ruang lingkupnya, dan lokasi pada gambar. Objek deteksi mengidentifikasi kelas objek yang terdapat pada database yang telah di-*training*. Deteksi objek diawali oleh pengenalan suatu objek. Hal ini dapat diperlakukan sebagai pengenalan objek kelas dua, dimana satu kelas mewakili kelas objek dan kelas lain mewakili kelas non-objek. Deteksi objek dibagi menjadi dua, yaitu *soft detection* dan *hard detection*. *Soft detection* hanya dapat mendeteksi keberadaan suatu objek sedangkan *hard*

detection mendeteksi keberadaan objek dan lokasi objek pada gambar (Jalled & Voronkov, 2016).

Objek deteksi biasanya dilakukan dengan mencari setiap bagian dari gambar untuk melokalisasi bagian, yang bersifat fotometrik atau geometrinya cocok dengan objek target dalam *training* basis data. Hal ini dapat dicapai dengan memindai template objek pada seluruh gambar di lokasi yang berbeda, skala, dan rotasi, dan deteksi dinyatakan jika kesamaan antara template dan gambar cukup tinggi. Kesamaan antara template dan wilayah gambar dapat diukur dengan korelasinya. Beberapa tahun terakhir telah terbukti bahwa objek deteksi berbasis gambar sensitif terhadap data *training* (Jalled & Voronkov, 2016).

Proses dalam deteksi objek pada video sama dengan proses deteksi objek pada gambar. Video terdiri dari beberapa gambar atau frame. Dalam satu video, dapat dipecah menjadi beberapa frame lagi, dimana pada setiap frame ini melakukan proses deteksi objek. Kemudian, frame yang telah diproses disatukan kembali menjadi video utuh. Pemrosesan deteksi objek pada video dapat dilakukan dengan menggunakan Graphical Processing Unit (GPU) (Salim, 2020).

2.1.4 *Deep Learning*

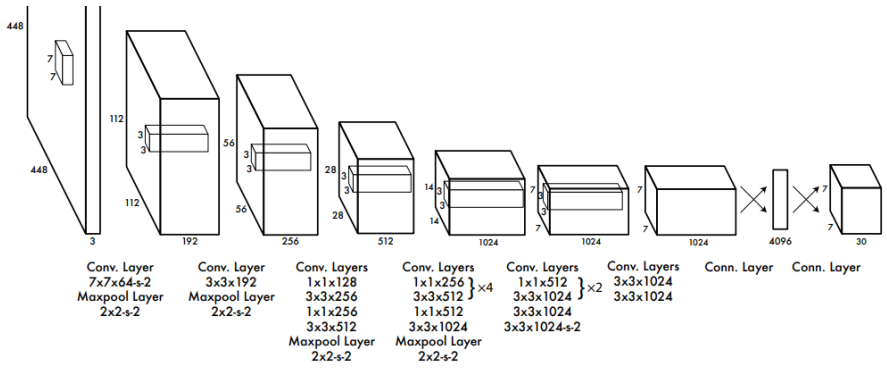
Deep Learning atau pembelajaran mendalam adalah salah satu bidang pada *machine learning* yang menggunakan banyak lapisan pada pengolahan informasi nonlinier dalam melakukan ekstraksi fitur, pengenalan pola, dan klasifikasi. Menurut goodfellow, *deep learning* atau pembelajaran mendalam adalah salah satu pendekatan yang menggunakan konsep hierarki untuk menyelesaikan masalah pada sistem pembelajaran komputer. Konsep hierarki ini menjadikan komputer dapat mempelajari konsep yang rumit dengan melakukan penggabungan dari konsep-konsep yang lebih sederhana. Jika diilustrasikan pada sebuah graf tentang bagaimana konsep tersebut dibangun di atas konsep-konsep yang lainnya, graf ini akan memiliki banyak lapisan, hal inilah yang menjadi alasan disebut pelajaran mendalam (*deep learning*) (Dewi, 2018).

Keuntungan utama *deep learning* adalah dapat merubah data dari *non-linearly separable* menjadi *linearly separable* melalui serangkaian transformasi (*hidden layers*). *Deep learning* juga dapat mencari *decision boundary* yang terbentuk non-linier, serta mengsimulasikan interaksi non-linier antar fitur. Jadi, masukan akan ditransformasikan secara non-linier sampai akhirnya untuk keluaran berbentuk distribusi *class-assignment*. Pada *training*, *deep learning* menggunakan *back propagation* (Umar et al., 2020).

2.1.5 *You Only Look Once* (YOLO)

You Only Look Once (YOLO) adalah salah satu pendekatan untuk melakukan pendeteksian objek secara *real-time* berbasis *Convolutional Neural Network*. YOLO menggunakan pendekatan jaringan syaraf tunggal (*Single neural network*) untuk melakukan pendeteksian objek pada sebuah citra. Jaringan ini menggunakan fitur dari semua gambar untuk memprediksi setiap *bounding box* yang dapat melakukan prediksi pada kotak-kotak pembatas dan probabilitas secara langsung dalam satu evaluasi (Redmon, Divvala, Girshick, & Farhadi, 2016).

Jaringan deteksi YOLO memiliki 24 lapisan konvolusi (*convolutional layer*) yang diikuti oleh 2 lapisan yang terhubung penuh (*fully connected layer*). Beberapa lapisan konvolusi menggunakan lapisan reduksi 1x1 sebagai alternative dalam mengurangi kedalaman *feature maps* yang diikuti oleh 3x3 lapisan konvolusional (*convolutional layer*) seperti pada Gambar 2.1.

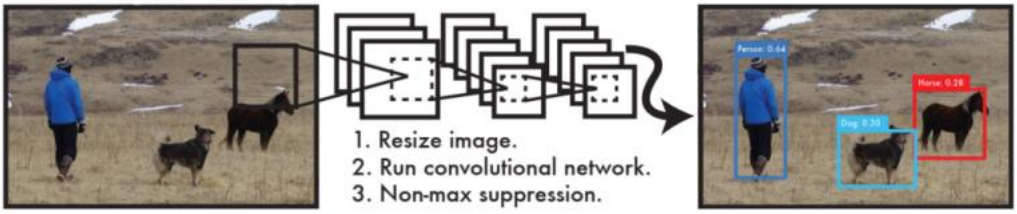


Gambar 2.1 Arsitektur YOLO

Sumber: (Redmon et al., 2016)

Terdapat tiga langkah dalam mendeteksi objek menggunakan YOLO yang diilustrasikan pada Gambar 2.2 seperti berikut:

a. Mengubah ukuran masukan gambar menjadi 448x448
b. Menjalankan jaringan syaraf tunggal (Single neural network) pada gambar
c. Melakukan threshold pada hasil deteksi berdasarkan nilai confidence



Gambar 2.2 Sistem deteksi YOLO

(Redmon et al., 2016)

YOLO membagi gambar input menjadi *grid* S x S. Jika pusat suatu objek jatuh ke dalam sel *grid*, maka sel *grid* bertanggung jawab untuk mendeteksi objek tersebut. Setiap sel *grid* memprediksi kotak pembatas (*bounding boxes*) B dan nilai keyakinan (*confidence score*) untuk kotak tersebut, serta probabilitas kelas kondisional C. Nilai keyakinan (*confidence score*) ini menggambarkan seberapa akurat kotak tersebut menurut perkiraannya. YOLO mendefinisikan *confidence* sebagai $\Pr(Object) \, . \, IoU_{pred}^{truth}$.

Jika tidak ada objek yang terdeteksi pada sel, nilai keyakinan akan bernilai nol. Jika tidak, sistem ingin nilai keyakinan sama dengan *Intersection Over Union* (IoU) antara kotak prediksi dan *ground truth*.

Setiap kotak pembatas B terdiri dari 5 komponen x, y, w, h, seperti pada Gambar 2.3, dan *confidence*. Nilai koordinat (x, y) menyatakan pusat kotak, *relative* terhadap batas kotak *grid*. Nilai koordinat kemudian dinormalisasi untuk jatuh di antara 0 dan 1. Lebar (w) dan tinggi (h) *relative* terhadap keseluruhan gambar, dan dinormalisasikan juga. Nilai keyakinan (*confidence score*) menyatakan seberapa yakin model tersebut, bahwa kotak pembatas B berisi sebuah objek dan seberapa akurat menurutnya kotak yang ia prediksi. Oleh karena itu, prediksi YOLO memiliki keluaran *vector* $S, S, B \, . \, 5 + C$.

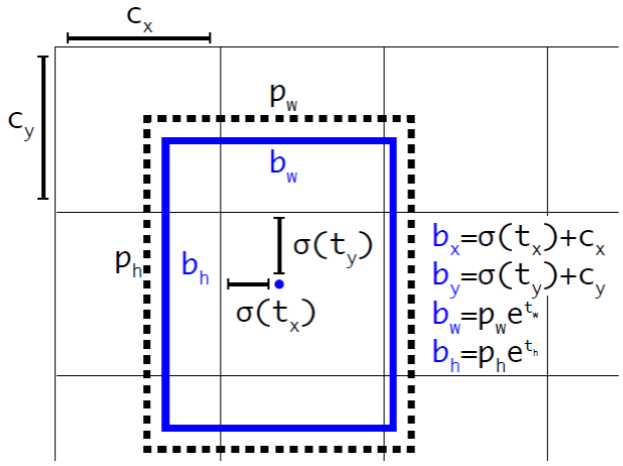
Faktor penentuan untuk mendapatkan prediksi akhir adalah *class confidence score*, berdasarkan probabilitas kondisional kelas dan *box confidence score*. *Class confidence score* mengukur nilai kepercayaan terhadap klasifikasi dan lokalisasi objek. *Class confidence score* memberi nilai kepercayaan kelas spesifik untuk setiap kotak, yang mengkodekan kemungkinan kelas yang muncul di kotak dan seberapa sesuainya kotak yang diprediksi dengan objek. Persamaan pada *class confidence score* untuk setiap kotak prediksi ditunjukkan pada Persamaan (2.1).

$$\Pr(Class_i|Object) \, . \, \Pr(Object) \; . IoU_{pred}^{truth} = \Pr(Class_i) \, . \, IoU_{pred}^{truth} \qquad (2.1)$$

Keterangan:

$\Pr(Class_i|Object)$: probabilitas kondisional kelas i.

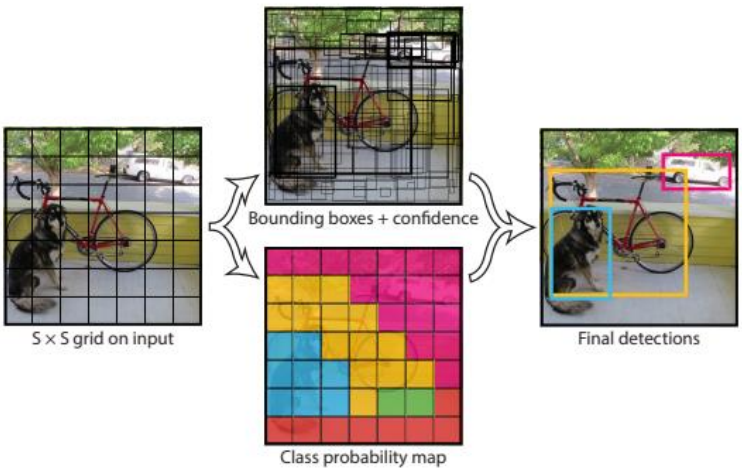
$\Pr(Object)$: probabilitas kelas i.



Gambar 2.3 Bounding box pada YOLO

Sumber: (Redmon & Farhadi, 2018)

Dari persamaan tersebut, akan mendapatkan nilai confidence dari kelas spesifik. Nilai ini akan merepresentasikan probabilitas kelas yang muncul didalam kotak dan seberapa akurat kotak yang diprediksi. Seperti pada Gambar 2.4 YOLO mendeteksi model sebagai regresi. Hal ini membagi gambar menjadi grid dan setiap grid memprediksi bounding boxes, nilai confidence dari setiap kotak dan kelas probabilitas.



Gambar 2.4 Proses deteksi pada YOLO

(Redmon et al., 2016)

Penelitian ini menggunakan YOLOv3 dimana Darknet53 sebagai *feature extractor*. Darknet53 merupakan jaringan baru untuk melakukan ekstraksi fitur. Dengan menggunakan 53 layer seperti pada Gambar 2.5 berbeda dengan versi sebelumnya YOLOv2 yang menggunakan Darknet-19. Pada jaringan Darknet53 menggunakan berturut-turut 3x3 dan 1x1 lapisan konvolusi.

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

Gambar 2.5 Darknet53

(Redmon & Farhadi, 2018)

2.1.6 OpenCV

OpenCV merupakan salah satu *library* yang memiliki fungsi-fungsi pemrograman pada *computer vision* secara *real-time*. OpenCV bersifat *open-source* yang dapat digunakan untuk membantu hal-hal yang bersifat akademis dan komersil. Pada OpenCV terdapat antarmuka untuk C, C++, Python, dan Java yang dapat dijalankan pada Windows, Mac, Linux, dan Android (Budiarjo, 2020). OpenCV memiliki lebih dari 2500 algoritma yang telah dioptimalkan.

2.1.7 *Intersection over Union* (IoU)

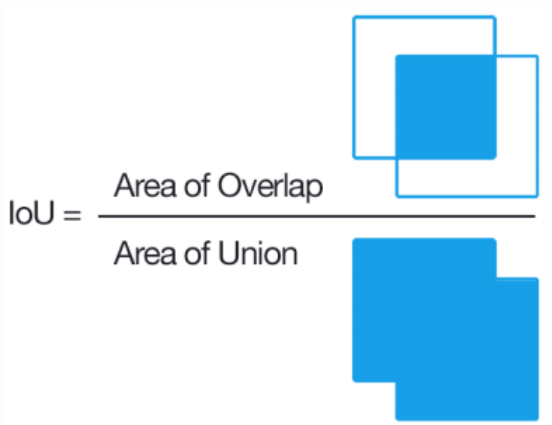
Intersection over Union (IoU) adalah metric evaluasi untuk mengukur keakuratan detektor objek pada dataset tertentu. IoU dapat digunakan dengan ketentuan sebagai berikut:

a. Memiliki *ground-truth bounding box* pada dataset objek

b. Prediksi *bounding box* pada dataset objek

Intersection over Union (IoU) merupakan perbandingan antara *ground-truth bounding box* dengan prediksi *bounding box* seperti pada persamaan 2.3 ilustrasi persamaan IoU dapat dilihat pada Gambar 2.6 (Umar et al., 2020).

$$IoU = \frac{A \cap B}{A \cup B} \tag{2.3}$$



Gambar 2.6 Intersection over Union

Sumber: (Umar et al., 2020)

2.1.8 *Mean Average Precision* (mAP)

Mean average precision merupakan nilai rata-rata dari *average precision* (AP) yang membentuk metrik evaluasi untuk mengukur kinerja dari sebuah deteksi objek. Nilai AP didapatkan dari perhitungan *precision* pada persamaan 2.6. dan perhitungan *recall* pada persamaan 2.7. yang selanjutnya dilakukan perhitungan seperti pada persamaan 2.4 (Umar et al., 2020).

$$AP = \sum(recall_{n+1} - recall_n).precision_{interp}.(recall_{n+1}) \tag{2.4}$$

2.1.9 *F-Measure*

F-Measure bertujuan untuk menghitung kombinasi dari presisi dan *recall*. *F-Measure* akan menggunakan *harmonic mean* dari presisi dan *recall*. Nilai *F-Measure* dihitung menggunakan persamaan 2.5 (Umar et al., 2020).

$$F - measure = \frac{2.recall.precision}{recall.precision} \tag{2.5}$$

2.1.10 *Precision*

Nilai presisi dihitung dengan cara membagi total sampel positif yang diklasifikasikan dengan benar dengan total sampel positif yang diprediksi seperti pada persamaan 2.6. Presisi tinggi menunjukan contoh berlabel positif memang positif (FP sedikit) (Umar et al., 2020).

$$Precision = \frac{TP}{TP + FP} \tag{2.6}$$

a. *High Recall*, *Low Precision* adalah beberapa sampel positif yang dapat dikenali (FN rendah) tetapi terdapat banyak positif palsu.
b. *Low Recall*, *High Precision* adalah kehilangan banyak sampel positif (FN tinggi) namun yang diprediksi positif benar-benar positif (FP rendah).

2.1.11 *Recall*

Recall adalah rasio dari total keseluruhan sampel positif yang diklasifikasikan dengan benar kemudian dibagi dengan total sampel positif. *High recall* menunjukan bahwa kelas dideteksi dengan benar (FN sedikit). Recall dapat dihitung dengan persamaan 2.7 (Umar et al., 2020).

$$Recall = \frac{TP}{TP + FN} \tag{2.7}$$

2.1.12 *Confusion Matrix*

Confusion matrix atau *error matrix* adalah ringkasan hasil prediksi pada permasalah klasifikasi. Jumlah klasifikasi yang benar dan yang salah dikumpulkan dengan nilai hitung kemudian dipecah oleh setiap kelas. Sehingga *confusion matrix* tidak hanya memberikan

informasi kesalahan yang dibuat classifier tetapi juga jenis kesalahannya (Umar et al., 2020). *Confusion matrix* dapat dilihat dengan beberapa ketentuan sebagai berikut:

a. *Positive* (P): aktual bernilai positif
b. *Negative* (N): aktual bernilai negatif
c. *True Positive* (TP): aktual bernilai positif, dan diprediksi positif
d. *True Negative* (TN): aktual bernilai negative, dan diprediksi negatif
e. *False Positive* (FP): aktual bernilai negatif, tetapi diprediksi negatif
f. *False Negative* (FN): aktual bernilai negatif, tetapi diprediksi negatif juga

2.2 Penelitian Terkait

Berikut beberapa rangkuman tentang penelitian terdahulu yang pernah dilakukan sebelumnya menggunakan data dan metode yang telah digunakan seperti pada Tabel 2..

Penelitian mengenai "Deteksi Kendaraan Secara Real Time Menggunakan Metode YOLO Berbasis Android" yang dilakukan oleh (Hutauruk, Matulatan, & Hayaty, 2020). Pada penelitian ini melakukan pendeteksian kendaraan yang masuk ke jalur yang tidak diperuntukkan bagi kendaraan tersebut. Analisa yang dilakukan menggunakan jumlah dataset sebanyak 200, 4 kelas, 10 *batch*, dan 200 *epoch*. Proses pelatihan dilakukan hingga 4.000 *step*, dan penyimpanan *checkpoint* ke bentuk *protobuf file* dilakukan pada *step* 800, 1.000, 1.200, 1.400, 1.600, 1.800, 2.000, 3.000, dan 4.000. *Bounding box* berhasil mendeteksi dan mengklasifikasi objek secara tepat. Pengujian ini dilakukan menggunakan perangkat *smartphone* Xiaomi Redmi 4X dengan resolusi video berukuran 764x432 piksel.

Penelitian mengenai "Sistem Pengenalan Plat Nomor Otomatis Menggunakan YOLOV3 Dengan Framework Keras" yang dilakukan oleh (Syuhaila, 2020). Penelitian ini melakukan deteksi plat nomor kendaraan otomatis menggunakan algoritma YOLO dengan *framework* darknet. Pada tahapan pendeteksian dimulai dari lokalisasi plat nomor kendaraan kemudian dilanjutkan dengan pembacaan karakter pada plat nomor. Hasil dari penelitian ini menghasilkan tingkat akurasi mencapai 98,52% dengan dengan rata-rata waktu pembacaan 3,03 detik dan untuk hasil OCR dengan Software Tesseract di dapatkan hasil deteksi 20%, hal ini menunjukkan system telah berhasil mengenali seluruh karakter pada plat mobil yang berupa Alphanumerik sebanyak 6-7 karakter.

Penelitian mengenai "Deteksi Jenis Mobil Menggunakan Metode YOLO Dan Faster R-CNN" yang dilakukan oleh (Shianto, Gunadi, & Setyati, 2019). Penelitian ini menggunakan dua metode, yaitu Faster R-CNN dan YOLO untuk mengidentifikasi jenis mobil secara cepat

dan tepat. Tujuan menggunakan kedua metode dalam arsitektur yang dibangun untuk meningkatkan akurasi kebenaran identifikasi jenis mobil.

Penelitian mengenai "Deteksi Plat Nomor Kendaraan Bergerak Berbasis Metode You Only Look Once (YOLO)" (Pratama, 2020). Penelitian ini melakukan pendeteksian plat nomor kendaraan. Pendeteksian dimulai dari input video, akuisisi, input ROI, proses YOLO, *tracking*, memilih *confidence* tertinggi dan cropping. Hasil yang didapatkan pada *Mean Average Precision* (mAP%) 87, 89% dan lokasi deteksi plat nomor terlihat secara empiris memiliki rata-rata akurasi sebesar 85,81%.

Penelitian mengenai "Sistem Klasifikasi Jenis Kendaraan Melalui Teknik Olah Citra Digital" yang dilakukan oleh (Pribadi & Naseer, 2016). Penelitian ini bertujuan untuk mendeteksi pengendara sepeda motor yang melakukan pelanggaran jalur masuk di tol pulau Bali. Sistem dapat mengklasifikasi kendaraan bermotor yang dapat digunakan untuk memperingati pengendara yang berkendara tidak pada tempatnya. Hasil dari penelitian ini, sistem mampu mendeteksi dan mengklasifikasi jenis kendaraan dengan tingkat akurasi yang berbeda untuk jarak tertentu.

Penelitian mengenai "Sistem Cerdas Pemantauan Arus Lalu Lintas Dengan YOLO (You Only Look Once v3)" yang dilakukan oleh (Harahap et al., 2019). Sistem ini dirancang untuk mendeteksi kendaraan di jalan raya. Menggunakan dataset bersumber dari CCTV ATCS Dishub Kota Medan. Sistem dapat mendeteksi mobil, sepeda motor, bus, truk, sepeda dan orang. YOLOv3 mampu mengklasifikasi kendaraan dengan mAP (mean Average Precision) pada CCTV Fix yang paling tertinggi yaitu 97% sedangkan pada CCTV PTZ sebesar 99%.

Penelitian mengenai "Visual Object Detection and Tracking using YOLO and SORT" yang dilakukan oleh (Bathija, 2019). Penelitian ini bertujuan untuk menganalisis pelacakan objek dengan pendekatan deteksi menggunakan YOLO dan pelacakan menggunakan algoritma SORT. Menggunakan masukan video untuk mengenali kendaraan dan pejalan kaki untuk analisis lalu lintas. Dataset yang digunakan sebanyak 800 gambar dengan 6 kelas. Akurasi dan presisi dapat terdeteksi dengan melatih sistem dengan menggunakan lebih banyak epoch. Kinerja SORT untuk pelacakan tergantung pada kinerja detector.

Penelitian mengenai "Klasifikasi Jenis Kendaraan Menggunakan Metode *Convolutional Neural Network* (CNN)" yang dilakukan oleh (Fadlia & Kosasih, 2019). Penelitian ini bertujuan untuk klasifikasi jenis-jenis kendaraan yang tidak sesuai pada jalurnya. Menggunakan sebanyak 120 dataset citra yang terdiri dari citra mobil, motor, dan sepeda. Hasil

menggunakan package Keras menunjukkan akurasi sebesar 94,4% pada tahap pelatihan dan 73,3% pada tahap pengujian.

Penelitian mengenai "Deteksi dan Penggolongan Kendaraan dengan *Kalman Filter* dan Model *Gaussian* di Jalan Tol" yang dilakukan oleh (Waliulu, 2018). Penelitian ini menggunakan 2 jenis distribusi, yaitu distribusi *Background* dan *Foreground* untuk mendeteksi objek bergerak yang terdapat dalam file video (*.avi) dengan resolusi 640x480. Data rekam yang digunakan berdurasi selama lima menit. Data diuji satu pertama dan empat menit terakhir, dengan jumlah durasi lima menit. Hasil ekstraksi ciri dari kendaraan tersebut digunakan untuk melakukan penggolongan kendaraan berdasarkan dimensi pixel. Hasil segmentasi digunakan *Kalman Filter* untuk menghitung pelacakan posisi objek bergerak. Jika segmentasi *Bit Large Object* tidak terdapat objek yang bergerak, maka akan diteruskan pada *frame* selanjutnya. Hasil akhir deteksi sistem didapatkan dari validasi True Positif, True Negatif, False Positif, dan False Negatif dengan mencari sensitifitas dan spesifisitas pada waktu pagi, siang, dan malam.

Penelitian mengenai "*Detection and classification of vehicles for traffic video analytics*" yang dilakukan oleh (Arinaldi et al., 2018). Penelitian ini bertujuan untuk menganalisis sistem lalu lintas berbasis *Computer Vision*. Statistik ini mencakup perhitungan kendaraan, klasifikasi tipe kendaraan, estimasi kecepatan kendaraan, dan pemantuan penggunaan jalur. Inti dari sistem ini adalah mendeteksi dan mengklasifikasi kendaraan dalam video lalu lintas. Menggunakan 2 dataset lalu lintas yang berbeda. Pertama, dataset Jalan Tol Indonesia adalah dataset milik mereka yang diambil melalui Tol Jagorawi dan Tol Kapuk. Kedua, menggunakan dataset MIT *traffic*. Hasil dari penelitian ini mengatakan bahwa Faster R-CNN lebih unggul dalam mendeteksi kendaraan bergerak dibandingkan dengan model MoG Background Subtraction.

Penelitian mengenai "Rancang bangun penghitung dan pengidentifikasi kendaraan menggunakan *Multiple Object Tracking"* (Rahmawati, Fisika, Sains, & Diponegoro, 2017). Penelitian ini melakukan implementasi sistem penghitung dan pengidentifikasi kendaraan di jalan tol menggunakan *multiple object tracking*. Sistem menggunakan algoritma *Gaussian mixture model* dan *Kalman filter* untuk mendeteksi dan melacak posisi, kecepatan, arah gerak dan ukuran kendaraan dari waktu ke waktu pada tiap frame citra. Hasil dari penelitian menyatakan saat pagi hari menghasilkan hasil terbaik, dan pada malam hari hasil terburuk. Akurasi pada pagi hari sebesar 94%, siang hari 90%, sore hari 85%, dan malam hari 59%.

Berdasarkan penelitian yang telah disebutkan, dengan beberapa metode yang berbeda-beda, serta masukan dan dataset yang berbeda-beda pula. Diketahui bahwa penelitian tentang

deteksi kendaraan menggunakan algoritma *You Only Look Once* (YOLO) dapat menghasilkan nilai akurasi yang tinggi. Oleh karena itu, pada penelitian ini akan dilakukan pendeteksian dan klasifikasi kendaraan menggunakan algoritma *You Only Look Once* (YOLO).

Tabel 2.1 Penelitian Terkait

No	Penelitian	Metode	Uraian Singkat
1	Deteksi Kendaraan Secara Real Time Menggunakan Metode YOLO Berbasis Android (Hutauruk et al., 2020)	*You Only Look Once* (YOLO)	Penelitian ini melakukan pendeteksian kendaraan yang masuk ke jalur yang tidak diperuntukkan bagi kendaraan tersebut. Analisa yang dilakukan menggunakan jumlah dataset sebanyak 200, 4 kelas, 10 *batch*, dan 200 *epoch*. Proses pelatihan dilakukan hingga 4.000 *step*, dan penyimpanan *checkpoint* ke bentuk *protobuf file* dilakukan pada *step* 800, 1.000, 1.200, 1.400, 1.600, 1.800, 2.000, 3.000, dan 4.000. *Bounding box* berhasil mendeteksi dan mengklasifikasi objek secara tepat. Pengujian ini dilakukan menggunakan perangkat *smartphone* Xiaomi Redmi 4X dengan resolusi video berukuran 764x432 piksel.
2	Sistem Pengenalan Plat Nomor Otomatis Menggunakan YOLOV3 Dengan Framework Keras (Syuhaila, 2020)	*You Only Look Once* (YOLO)	Penelitian ini melakukan deteksi plat nomor kendaraan otomatis menggunakan algoritma YOLO dengan *framework* darknet. Pada tahapan pendeteksian dimulai dari lokalisasi plat nomor kendaraan kemudian dilanjutkan dengan pembacaan karakter pada plat nomor. Hasil dari penelitian ini menghasilkan tingkat akurasi mencapai 98,52% dengan dengan rata-rata waktu pembacaan 3,03 detik dan untuk hasil OCR dengan Software Tesseract di dapatkan hasil deteksi 20%, hal ini menunjukkan system telah berhasil mengenali seluruh karakter pada plat mobil yang berupa Alphanumerik sebanyak 6-7 karakter.
3	Deteksi Jenis Mobil Menggunakan Metode YOLO Dan *Faster* R-CNN (Shianto et al., 2019)	*You Only Look Once* (YOLO) dan *Faster* R-CNN	Penelitian ini menggunakan dua metode, yaitu *Faster* R-CNN dan YOLO untuk mengidentifikasi jenis mobil secara cepat dan tepat. Tujuan menggunakan kedua metode dalam arsitektur yang dibangun untuk meningkatkan akurasi kebenaran identifikasi jenis mobil.
4	Deteksi Plat Nomor Kendaraan Bergerak	*You Only Look Once* (YOLO)	Penelitian ini melakukan pendeteksian plat nomor kendaraan. Pendeteksian

	Berbasis Metode *You Only Look Once* (YOLO) (Pratama, 2020)		dimulai dari input video, akuisisi, input ROI, proses YOLO, *tracking*, memilih *confidence* tertinggi dan *cropping*. Hasil yang didapatkan pada *Mean Average Precision* (mAP%) 87, 89% dan lokasi deteksi plat nomor terlihat secara empiris memiliki rata-rata akurasi sebesar 85,81%.
5	Sistem Klasifikasi Jenis Kendaraan Melalui Teknik Olah Citra Digital (Pribadi & Naseer, 2016)	*Optical Flow*	Penelitian ini bertujuan untuk mendeteksi pengendara sepeda motor yang melakukan pelanggaran jalur masuk di tol pulau Bali. Sistem dapat mengklasifikasi kendaraan bermotor yang dapat digunakan untuk memperingati pengendara yang berkendara tidak pada tempatnya. Hasil dari penelitian ini, sistem mampu mendeteksi dan mengklasifikasi jenis kendaraan dengan tingkat akurasi yang berbeda untuk jarak tertentu.
6	Sistem Cerdas Pemantauan Arus Lalu Lintas Dengan YOLO (You Only Look Once v3) (Harahap et al., 2019)	*You Only Look Once* (YOLO)	Sistem ini dirancang untuk mendeteksi kendaraan di jalan raya. Menggunakan dataset bersumber dari CCTV ATCS Dishub Kota Medan. Sistem dapat mendeteksi mobil, sepeda motor, bus, truk, sepeda dan orang. YOLOv3 mampu mengklasifikasi kendaraan dengan mAP (mean Average Precision) pada CCTV Fix yang paling tertinggi yaitu 97% sedangkan pada CCTV PTZ sebesar 99%.
7	Visual Object Detection and Tracking using YOLO and SORT (Bathija, 2019)	*You Only Look Once* (YOLO) dan SORT	Penelitian ini bertujuan untuk menganalisis pelacakan objek dengan pendekatan deteksi menggunakan YOLO dan pelacakan menggunakan algoritma SORT. Menggunakan masukan video untuk mengenali kendaraan dan pejalan kaki untuk analisis lalu lintas. Dataset yang digunakan sebanyak 800 gambar dengan 6 kelas. Akurasi dan presisi dapat terdeteksi dengan melatih sistem dengan menggunakan lebih banyak epoch. Kinerja SORT untuk pelacakan tergantung pada kinerja detector.
8	Klasifikasi Jenis Kendaraan Menggunakan Metode *Convolutional Neural Network* (CNN) (Fadlia & Kosasih, 2019)	*Convolutional Neural Network* (CNN)	Penelitian ini bertujuan untuk klasifikasi jenis-jenis kendaraan yang tidak sesuai pada jalurnya. Menggunakan sebanyak 120 dataset citra yang terdiri dari citra mobil, motor, dan sepeda. Hasil menggunakan package Keras menunjukkan akurasi

			sebesar 94,4% pada tahap pelatihan dan 73,3% pada tahap pengujian.
9	Deteksi dan Penggolongan Kendaraan dengan *Kalman Filter* dan Model *Gaussian* di Jalan Tol (Waliulu, 2018)	*Kalman Filter* dan Model *Gaussian Mixture*. Menggunakan 2 jenis distribusi, yaitu distribusi *Background* dan *Foreground*.	Penelitian ini menggunakan 2 jenis distribusi, yaitu distribusi *Background* dan *Foreground* untuk mendeteksi objek bergerak yang terdapat dalam file video (*.avi) dengan resolusi 640x480. Data rekam yang digunakan berdurasi selama lima menit. Data diuji satu pertama dan empat menit terakhir, dengan jumlah durasi lima menit. Hasil ekstraksi ciri dari kendaraan tersebut digunakan untuk melakukan penggolongan kendaraan berdasarkan dimensi pixel. Hasil segmentasi digunakan *Kalman Filter* untuk menghitung pelacakan posisi objek bergerak. Jika segmentasi *Bit Large Object* tidak terdapat objek yang bergerak, maka akan diteruskan pada *frame* selanjutnya. Hasil akhir deteksi sistem didapatkan dari validasi True Positif, True Negatif, False Positif, dan False Negatif dengan mencari sensitifitas dan spesifisitas pada waktu pagi, siang, dan malam.
10	*Detection and classification of vehicles for traffic video analytics* (Arinaldi et al., 2018)	Menerapkan 2 model yang berbeda, pertama model sistem MoG + SVM dan yang kedua Faster R-CNN	Penelitian ini bertujuan untuk menganalisis sistem lalu lintas berbasis *Computer Vision*. Statistik ini mencakup perhitungan kendaraan, klasifikasi tipe kendaraan, estimasi kecepatan kendaraan, dan pemantuan penggunaan jalur. Inti dari sistem ini adalah mendeteksi dan mengklasifikasi kendaraan dalam video lalu lintas. Menggunakan 2 dataset lalu lintas yang berbeda. Pertama, dataset Jalan Tol Indonesia adalah dataset milik mereka yang diambil melalui Tol Jagorawi dan Tol Kapuk. Kedua, menggunakan dataset MIT *traffic*. Hasil dari penelitian ini mengatakan bahwa Faster R-CNN lebih unggul dalam mendeteksi kendaraan bergerak dibandingkan dengan model MoG Background Subtraction.
11	Rancang bangun penghitung dan pengidentifikasi kendaraan menggunakan *Multiple Object Tracking* (Rahmawati et al., 2017)	*Gaussian mixture model* dan *Kalman filter*	Penelitian ini melakukan implementasi sistem penghitung dan pengidentifikasi kendaraan di jalan tol menggunakan *multiple object tracking*. Sistem menggunakan algoritma *Gaussian mixture model* dan *Kalman filter* untuk mendeteksi dan melacak posisi, kecepatan, arah gerak dan ukuran

			kendaraan dari waktu ke waktu pada tiap frame citra. Hasil dari penelitian menyatakan saat pagi hari menghasilkan hasil terbaik, dan pada malam hari hasil terburuk. Akurasi pada pagi hari sebesar 94%, siang hari 90%, sore hari 85%, dan malam hari 59%.

BAB III
METODOLOGI

3.1 Langkah-langkah Penelitian

Pada bagian ini dilakukan langkah-langkah penelitian digambarkan pada Gambar 3.1 yang terdiri dari studi literatur, perencanaan, pengumpulan data, implementasi, dan pengujian.

Gambar 3.1 Langkah-langkah penelitian

3.2 Uraian Penelitian

3.2.1 Studi Literatur

Tahap awal pada penelitian ini adalah studi literatur. Studi literatur dilakukan untuk pencarian masalah, mencari sumber terkait dengan penelitian dan landasan teori sebagai acuan untuk melakukan penelitian. Studi literatur dilakukan dengan melakukan tinjauan literatur yang berhubungan dengan pendeteksian dan klasifikasi kendaraan dengan menggunakan algoritma YOLO, dengan data masukan berupa gambar atau video. Proses pencarian literatur dilakukan dengan menggunakan *Google Scholar* dan *ResearchGate* dengan memasukan kata kunci yang sesuai. Adapun kata kunci utama yang dicari dibagi menjadi lima bagian, yaitu: a) "*vehicle detection*", b) "*classification vehicle*", c) "*image based*", d) "*video based*", e) "*You Only Look Once* (YOLO)". Hal-hal yang menjadi kriteria dalam memilih literatur sebagai berikut:

a. Literatur merupakan publikasi ilmiah.
b. Literatur membahas tentang pendeteksian dan klasifikasi kendaraan.
c. Literatur berasal dari publikasi ilmiah berbahasa Indonesia dan bahasa Inggris.
d. Literatur menggunakan pendekatan *image based* atau *video based*.

3.2.2 Perencanaan

Langkah selanjutnya adalah melakukan perencanaan dalam melakukan penelitian. Perencanaan meliputi literatur yang menjadi acuan penelitian, pemilihan *software* yang akan digunakan, dan pemilihan *hardware* yang akan digunakan sebagai berikut:

Tabel 3.1 Tabel spesifikasi perangkat keras dan perangkat lunak

Perangkat keras	RAM 8 GB
	Intel® Core™ i5-1035G1
Perangkat lunak	OS Windows 10 64 bit
	Python
	Library OpenCV

3.2.3 Pengumpulan Data

Pada langkah ini dilakukan pengumpulan data-data, alat, dan bahan yang diperlukan dalam penelitian. Data direkam dari Website *Area Traffic Control System* (ATCS) Dishub Kota Medan. Data tersebut berupa video rekaman CCTV arus lalu lintas di Simpang Air Mancur-Imanuel Kota Medan. Data direkam dari 4 hari dan jam yang berbeda-beda dengan durasi yang berbeda-beda juga. Setelah didapatkan hasil rekaman CCTV, dilakukan perubahan pada frame size menjadi 1280x962.

Tabel 3.2 Pembagian dataset

Tanggal/Jam	Durasi	Frame Size	Frame Rate
14-04-2021 17:22	38 detik	1280x962	30 fps
29-06-2021 09:03	2 menit	1280x962	30 fps
07-07-2021 12:08	44 detik	1280x962	30 fps
08-07-2021 09:50	1 menit 11 detik	1280x962	30 fps

Data masukan pada penelitian ini berupa video dengan format .mp4. untuk melakukan deteksi dibutuhkan berupa data citra, maka dari itu video perlu diubah menjadi bentuk citra agar bisa diproses ke tahap selanjutnya.

Gambar 3.2 Contoh gambar data CCTV tanggal 14-04-2021

Gambar 3.3 Contoh gambar data CCTV tanggal 29-06-2021

Gambar 3.4 Contoh gambar data CCTV tanggal 07-07-2021

Gambar 3.5 Contoh gambar data CCTV tanggal 08-07-2021

Penulis juga menggunakan beberapa data yang diambil dari internet yang terdiri dari jalan raya untuk melakukan proses training yang didalamnya terdapat mobil, sepeda motor, truk, bus, dan becak.

Gambar 3.6 Contoh gambar data dari internet

3.2.4 Implementasi

Setelah melakukan pengumpulan data, selanjutnya dilakukan anotasi data. Anotasi data bertujuan untuk memberikan label pada gambar dengan memberikan kotak pembatas (*bounding box*) dan nama kelas pada setiap objek. Kemudian, melakukan *training data* yang bertujuan untuk melatih komputer dengan mengolah data yang telah di anotasikan agar terbentuk suatu karakteristik sebagai pertimbangan untuk mencapai sebuah prediksi. Langkah

selanjutnya adalah mengimplementasikan algoritma *You Only Look Once* (YOLO) untuk melakukan deteksi dan klasifikasi kendaraan. Implementasi dilakukan dengan mengimplementasikan hasil desain ke dalam perangkat lunak menggunakan bahasa yang dapat dibaca oleh computer. Dijelaskan pada Bab IV.

3.2.5 Pengujian

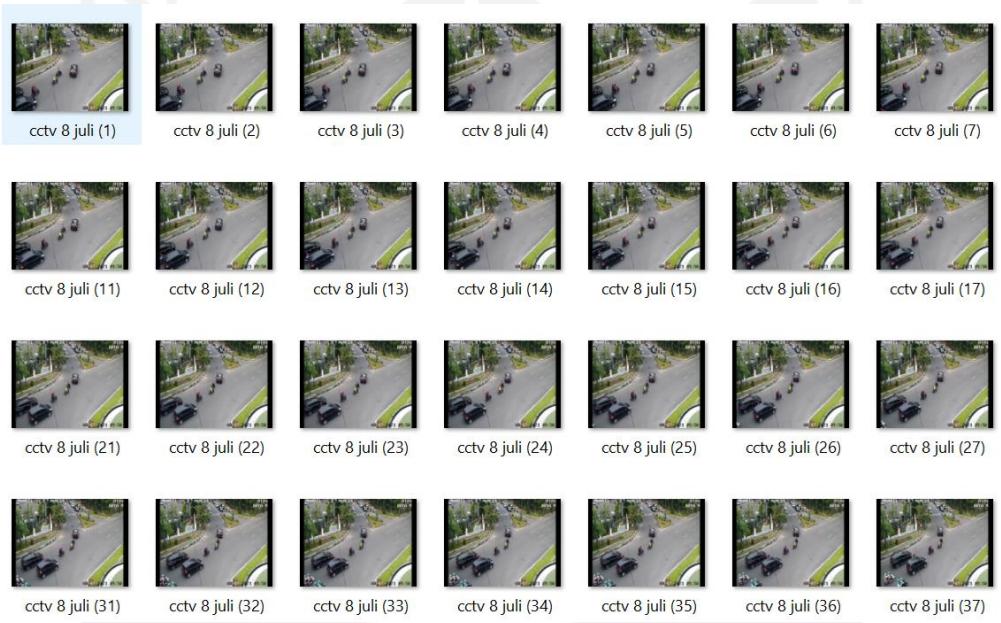
Proses pengujian dilakukan untuk mengetahui hasil dari deteksi dan klasifikasi kendaraan. Pengujian ini dilakukan dengan menggunakan masukan video CCTV Simpang Air Mancur – Imanuel. Hasil dari pengujian ini untuk mengetahui seberapa akurat dari algoritma YOLO. Dijelaskan pada Bab IV.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Implementasi

4.1.1 Anotasi Objek

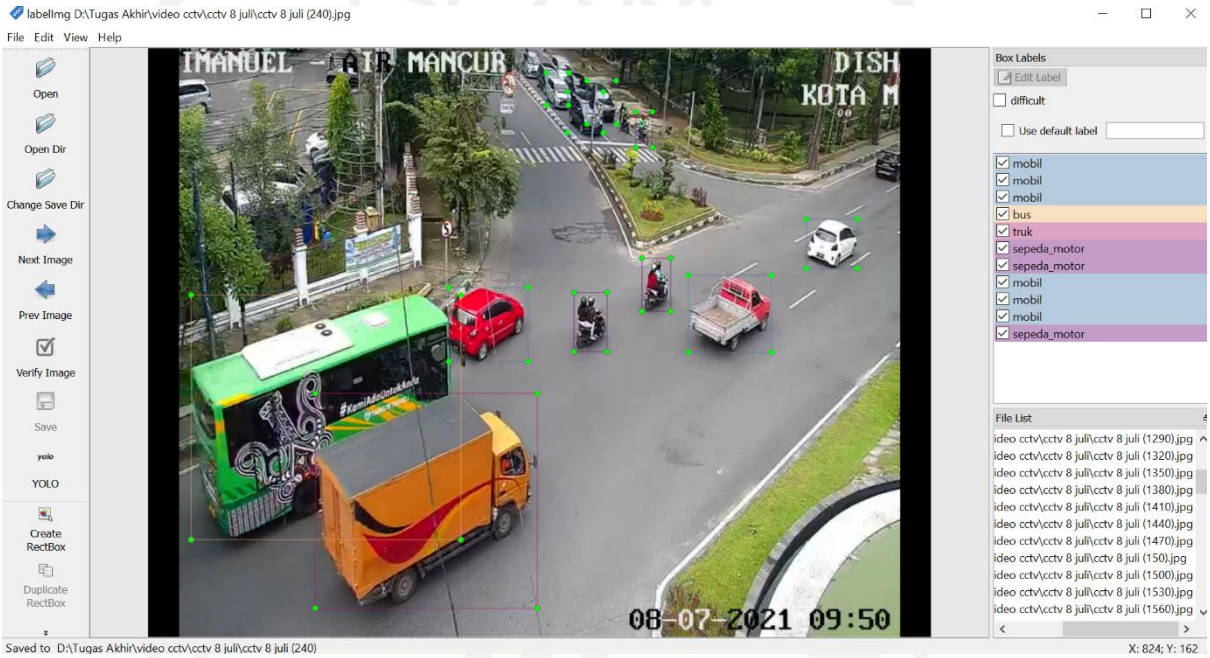
Anotasi atau pelabelan objek adalah proses pembuatan label pada gambar dengan cara memberikan kotak pembatas (*bounding box*) beserta nama kelas pada objek. Tahap pelabelan objek dilakukan ketika data yang berupa video rekaman CCTV dari *Website* ATCS Dishub Kota Medan telah diolah menjadi gambar pada masing-masing video seperti pada Gambar 4.1.



Gambar 4.1 Hasil transformasi video menjadi gambar

Pelabelan objek dibagi menjadi lima kelas, yaitu mobil, sepeda motor, bus, truk, dan becak. Objek yang dilabel hanya tampak belakang sesuai hasil dari video rekaman. Proses pelabelan mobil dimulai dari atap mobil sampai roda mobil, pelabelan pada sepeda motor dimulai dari kepala sampai roda sepeda motor, kemudian pelabelan pada truk dimulai dari bagian atas bak truk sampai roda truk, lalu pelabelan pada bus dimulai dari atap bus sampai roda bus, dan pelabelan pada becak dimulai dari bagian atas becak sampai roda becak. Skenario pelabelan secara visual dapat dilihat pada Gambar 4.2. Ketika proses anotasi sudah selesai, maka akan terlihat seperti Gambar 4.3 yang terdapat file gambar dan file .txt.

Proses anotasi citra menggunakan software LabelImg. Pelabelan ini menggunakan anotasi dalam format anotasi YOLO. Hasil dari anotasi tersebut adalah data yang terdapat informasi letak kotak pembatas dan labelnya dalam bentuk .txt. Pada .txt file terdapat baris file yang memiliki format <*object-class*> <*x_center*> <*y_center*> <*width*> <*height*>, dimana pada <*object-class*> merupakan bilangan bulat yang menyatakan kelas objek, <*x_center*> dan <*y_center*> adalah koordinat pusat persergi kotak pembatas, <*width*> dan <*height*> adalah nilai *float* relatif terhadap dimensi gambar. Hasil dari .txt file dapat dilihat pada Gambar 4.4.



Gambar 4.2 Proses anotasi dataset menggunakan LabelImg

Berikut merupakan hasil gambar yang sudah diberikan label:

Gambar 4.3 Gambar Dataset sudah diberikan label

```
0 0.183984 0.753119 0.141406 0.204782
0 0.198828 0.616944 0.088281 0.107069
```

Gambar 4.4 Hasil dari .txt file

4.1.2 *Training Data*

Setelah anotasi selesai dilakukan, langkah selanjutnya melakukan proses *training*. Proses ini bertujuan untuk melatih komputer dengan cara mengolah gambar dan anotasi yang sudah dibuat sehingga terbentuk pola atau karakteristik dari setiap kelas yang akan menjadi bahan pertimbangan komputer dalam mencapai sebuah keputusan atau prediksi. Pada bagian ini menggunakan *pre-trained weights* YOLOv3. Dengan menggunakan teknik *transfer learning*. *Transfer learning* adalah suatu teknik yang menggunakan model yang telah di-*training* sebelumnya (*pre-trained model*) yang dapat digunakan untuk klasifikasi dataset baru tanpa harus melakukan *training* data dari awal. Proses *transfer learning* pada Darknet menggunakan data file, cfg file, dan *pre-trained weigths*. Data file berisi lokasi gambar yang digunakan untuk *train* dan *test*. Cfg file berisi bentuk jaringan yang digunakan untuk *training*, dan *pre-trained weights* berisi model *weight* yang telah dilatih sebelumnya pada jaringan YOLO.

Karena proses komputasi yang berat saat melakukan proses *training*, maka penulis menggunakan *Google Colaboratory* yang merupakan sebuah platform yang telah disediakan oleh *Google*. Ketika proses *training* sedang berjalan, dibutuhkan koneksi internet yang stabil agar runtime saat proses *training* tidak terputus. Proses *training* ini dilakukan menggunakan *framework neural network Darknet* dengan konfigurasi seperti pada Tabel 4.1.

Tabel 4.1 Konfigurasi pada Darknet

Jenis Konfigurasi	**Keterangan**
Load model	Darknet
Load weight	Yolov3
OPENCN	1
GPU	1
CUDNN	1

Proses training menggunakan Darknet-53 sebagai load model dengan susunan layer seperti Gambar 2.5. dan YOLOv3 sebagai *load weight* dengan konfigurasi seperti pada Tabel 4.2. Jumlah *batch* menentukan jumlah gambar yang akan diproses sebelum *network weight* mengalami pembaharuan. *Subdivision* bertujuan untuk memproses sebagian kecil *batch size* sekaligus pada GPU. *Max_batch* merupakah batas iterasi dalam melakukan *training* yang didapatkan dengan persamaan. Ketika iterasi sudah mencapai 10000, maka proses *training* akan otomatis berhenti. Nilai *max_batches* ditentukan pada persamaan 4.1.

$$\max\_batches = jumlahclass \,.\, 2000 \tag{4.1}$$

Nilai step didapatkan pada persamaan 4.2.

$$steps = (80\%\max\_batches), (90\%\max\_batches) \tag{4.2}$$

Height dan Weight merupakan dimensi gambar masukan yang akan dilatih. Classes merupakan jumlah kelas yang akan dideteksi. Nilai filter didapatkan pada persamaan 4.3.

$$filter = (jumlahclass + 5) \times 3 \tag{4.3}$$

Tabel 4.2 Konfigurasi pada *weights* YOLOv3

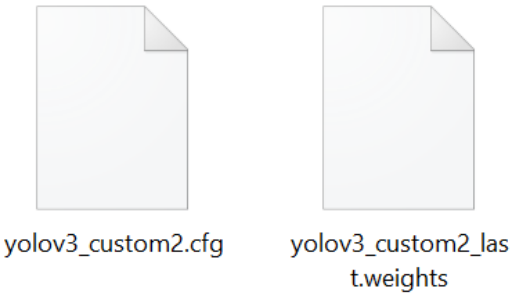
Jenis Konfigurasi	**Keterangan**
batch	64
subdivisions	16
width	416
height	416
max_batches	10000
steps	8000, 9000

classes	5
filters	30

4.1.3 Instalasi Software

a. Download file weight dan file configuration YOLO

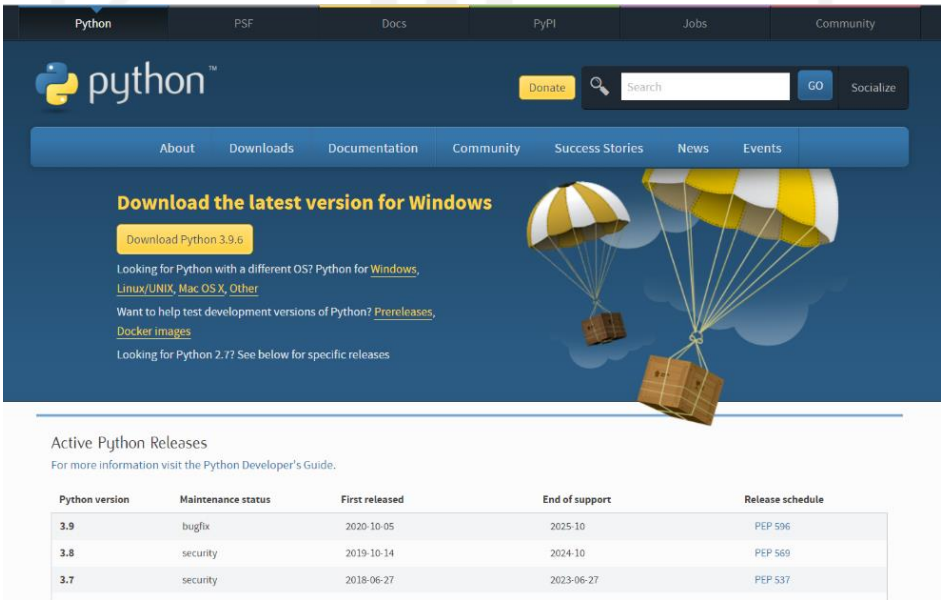
Langkah selanjutnya adalah melakukan download *file weight* dan *file configuration* YOLOv3 yang sudah selesai di-*training* seperti pada Gambar 4.5. File ini adalah inti dari algoritma YOLO untuk mendeteksi objek dan *file configuration* adalah pengaturan dari algoritma YOLO.



Gambar 4.5 File weight dan cfg YOLO

b. Instalasi Python

Pada penelitian ini, penulis menggunakan bahasa Python untuk melakukan pendeteksian.



Gambar 4.6 Tampilan *website* python

c. Instalasi OpenCV

Selanjutnya melakukan instalasi OpenCV. Disini penulis menggunakan OpenCV 3 dengan code seperti pada Gambar 4.7.

```
>pip install opencv-python
```

Gambar 4.7 Kode program instalasi OpenCV

4.1.4 Deteksi Kendaraan dengan Algoritma You Only Look Once (YOLO)

Pada proses deteksi, setelah semua *software* yang dibutuhkan telah ter-instalasi dengan baik. Langkah selanjutnya adalah menggabungkan file *weight*, *configuration*, dan dataset pada satu folder.

a. Import library

Menggunakan library cv2, numpy, dan time. Library cv2 berfungsi untuk melakukan *computer vision task*. Library numpy berfungsi untuk pengolahan data numerical.

```
import cv2
import numpy as np
import time

print(cv2.__version__)
```

Gambar 4.8 Kode program import library

```
C:\Users\HP\anaconda3\envs\OpenCV\python.exe C:/Users/HP/PycharmProjects/pythonProject/detect.py
4.5.1
```

Gambar 4.9 Hasil dari keluaran versi OpenCV

b. Membaca Masukan Video

Pada bagian ini dilakukan membaca masukan video dengan mendefinisikan cv2.VideoCapture() untuk mendapatkan objek dari video capture pada kamera sesuai dengan lokasi penyimpanan. Lalu menggunakan variabel '*writer*' yang digunakan untuk menulis *frame* yang diproses. Kemudian menggunakan variabel 'h' dan 'w' untuk dimensi *frame*.

```
video = cv2.VideoCapture('cctv8juli.mp4')
writer = None
h, w = None, None
```

Gambar 4.10 Kode program membaca masukan video

c. Memuat *network* YOLOv3

Menggunakan file *weight*, cfg, dan *name files*. *Weight file* adalah model yang sudah di-*training*, inti dari algoritma YOLO untuk mendeteksi objek. Cfg file adalah file konfigurasi, dimana semua pengaturan algoritma YOLO terdapat pada file tersebut. Dan *Name files* adalah file yang berisi nama-nama objek yang dapat dideteksi menggunakan algoritma YOLO. Disini penulis menggunakan *name files* yang berisikan lima kelas yang akan dideteksi. Setelah itu mengatur *minimum* probabilitas untuk mengeliminasi prediksi rendah dengan nilai 0,5. Serta melakukan pengaturan untuk menyaring kotak pembatas yang rendah dengan nilai threshold 0,5 jika lebih besar maka objek akan terdeteksi dengan benar, jika tidak maka akan dilewatkan. Nilai threshold berubah dari 0 ke 1. Semakin dekat ke 1 maka semakin besar akurasi deteksi, jika semakin dekat ke 0 maka semakin sedikit akurasi tetapi juga semakin besar jumlah objek yang terdeteksi. Untuk menghasilkan warna pada setiap objek yang terdeteksi, menggunakan fungsi *randint* dengan seperti pada Gambar 4.11.

```
with open('kendaraan.names') as f:
    labels = [line.strip() for line in f]

print('List with labels names:')
print(labels)

network = cv2.dnn.readNetFromDarknet('yolov3_custom2.cfg',
                                     'yolov3_custom2_last.weights')
layers_names_all = network.getLayerNames()
layers_names_output = \
    [layers_names_all[i[0] - 1] for i in network.getUnconnectedOutLayers()]
probability_minimum = 0.5
threshold = 0.5
colours = np.random.randint(0, 255, size=(len(labels), 3), dtype='uint8')
```

Gambar 4.11 Kode program memuat *network* YOLOv3

```
List with labels names:
['mobil', 'sepeda motor', 'becak', 'truk', 'bus']
```

Gambar 4.12 Hasil keluaran *labels*

d. Membaca *frame* untuk perulangan

Kemudian melakukan perulangan pada *frame*. Disini akan dilakukan fungsi perulangan untuk menangkap *frame-by-frame* dengan fungsi *read*, yaitu membaca *frame* dari video

masukan. Jika *frame* tidak diambil, misalnya pada akhir video, maka program akan menghentikan perulangannya.

```
f = 0
t = 0

while True:
    ret, frame = video.read()
    if not ret:
        break
    if w is None or h is None:
        h, w = frame.shape[:2]
```

Gambar 4.13 Kode program membaca *frame* untuk perulangan

e. Mengambil fungsi *blob* dari *frame*

Kemudian, data akan diolah menggunakan library OpenCV untuk diubah menjadi bentuk *blob* (*Binary Large Object*) dari *frame*. Menggunakan fungsi '*cv2.dnn.blobFromImage*' akan mengembalikan 4-dimensional *blob* dari *frame* saat ini setelah pengurangan rata-rata, normalisasi, dan pertukaran saluran RB (Irwan, Putrada, & Prabowo, 2019). Pada fungsi '*cv2.dnn.blobFromImage*' berisi parameter *frame* yang merupakan masukan gambar yang akan diproses melalui jaringan saraf dalam bentuk klasifikasi. 1/255.0 sebagai *scale factor* untuk melakukan pengurangan rata-rata. Ukuran (416,416) adalah ukuran pada YOLO. *swapRb* memberikan asumsi pada gambar berada dalam saluran BGR, namun nilai mean mengasumsikan kita menggunakan urutan RGB. Untuk mengatasi perbedaan ini, kita dapat menetapkan nilai menjadi '*True*' untuk menukan saluran R dan B. Bentuk yang dihasilkan memiliki jumlah bingkai, jumlah saluran, lebar dan tinggi. *Blob* digunakan untuk mengekstrak fitur gambar dan mengubah ukurannya. Pada YOLO terdapat 3 ukuran *frame*, yaitu 320x320, 416x416, dan 609x609. Disini penulis menggunakan ukuran frame 416x416 agar proses deteksi memiliki kecepatan dan akurasi yang seimbang.

```
blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416), swapRB=True,
                             crop=False)
```

Gambar 4.14 Kode program mengambil fungsi *blob* dari *frame*

f. Menerapkan *Forward Pass*

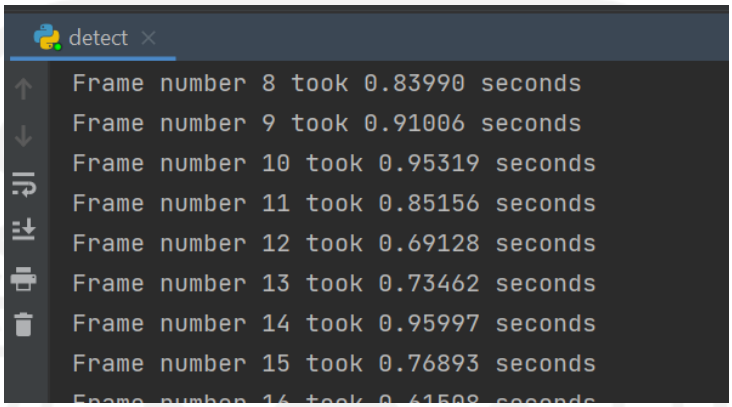
Mengimplementasi *forward pass* dengan '*blob*' melalui lapisan keluaran. Lalu, menghitung waktu yang dibutuhkan untuk *forward pass*.

```
network.setInput(blob)
    start = time.time()
    output_from_network = network.forward(layers_names_output)
    end = time.time()

    f += 1
    t += end - start

    print('Frame number {0} took {1:.5f} seconds'.format(f, end - start))
```

Gambar 4.15 Kode program menerapkan *Forward Pass*



Gambar 4.16 Contoh hasil dari proses deteksi per *frame*

g. Mendapatkan *bounding boxes*

Pada langkah ini dilakukan ekstraksi seluruh informasi objek yang dideteksi dan menampilkannya pada layar. Melakukan inisialisasi list *bounding_boxes* untuk kotak pembatas yang mengelilingi objek. *Confidences* memberikan nilai keyakinan pada YOLO untuk suatu objek, nilai keyakinan (*confidence*) dari objek yang terdeteksi bernilai 0 sampai 1. ClassIDs untuk memberikan label kelas objek yang terdeteksi.

Melakukan perulangan untuk setiap *output_from_network* dan perulangan untuk setiap *detected_object* pada *result* dengan mengekstrak *class_current* dan *confidence_current*. Menggunakan perintah *confidence_current* untuk menyaring prediksi yang lemah dengan memastikan objek yang terdeteksi dengan probabilitas lebih besar dari probabilitas minimum. Kemudian melakukan skala koordinat kotak pembatas untuk menampilkan dengan benar pada gambar asli. Lalu, mengekstraksi koordinat dan dimensi dari kotak pembatas dengan mengembalikan koordinat kotak pembatas dalam bentuk: *x_center*, *y_center*, *box_width*, *box_height*. Menggunakan informasi untuk mendapatkan *top-left (x, y)-coordinates* dari kotak pembatas. Lalu memperbarui list *bounding_boxes*, *confidences*, dan *classIDs*.

```
bounding_boxes = []
confidences = []
classIDs = []

    for result in output_from_network:

        for detected_objects in result:
            scores = detected_objects[5:]
            class_current = np.argmax(scores)
            confidence_current = scores[class_current]

            if confidence_current > probability_minimum:
                box_current = detected_objects[0:4] * np.array([w, h, w, h])
                x_center, y_center, box_width, box_height = box_current
                x_min = int(x_center - (box_width / 2))
                y_min = int(y_center - (box_height / 2))
                bounding_boxes.append([x_min, y_min, int(box_width),
                                       int(box_height)])
               confidences.append(float(confidence_current))
               classIDs.append(class_current)
```

Gambar 4.17 Kode program mendapatkan *bounding boxes*

h. *Non-maximum Suppresion*

Menerapkan *non-maximum suppresion* atau yang disebut juga *non-maxima suppression* (NMS). Objek yang terdapat pada video atau gambar dapat memiliki ukuran dan bentuk yang berbeda, dan untuk menangkap setiap objek maka algoritma akan memuat kotak pembatas (*bounding boxes*). Jadi, untuk setiap objek yang terdapat pada video atau gambar harus memiliki satu kotak pembatas (*bounding boxes*). Untuk mendapatkan kotak pembatas (*bounding boxes*) terbaik dari beberapa kotak pembatas (*bounding boxes*) yang diprediksi, algoritma akan menggunakan fungsi *non-max*. teknik ini digunakan untuk "menekan" kotak pembatas yang kemungkinannya kecil dan hanya menyimpan yang terbaik. Tujuan dari *non-maximum suppression* adalah untuk memilih kotak pembatas (*bounding boxes*) terbaik pada suatu objek. NMS akan mempertimbangkan dua hal, yang pertama skor objektifitas yang diberikan oleh model, dan yang kedua IoU dari kotak pembatas (*bounding boxes*) (Hosang & May, n.d.).

Pada Gambar 4.18 memanfaatkan implementasi modul DNN pada OpenCV, kemudian melakukan *non-maximum suppression* dengan parameter *bounding_boxes*, *confidences*, *probability_minimun*, dan *threshold*.

```
results = cv2.dnn.NMSBoxes(bounding_boxes, confidences,
                           probability_minimum, threshold)
```

Gambar 4.18 Kode program *non-maximum suppresion*

i. Mendapatkan *bounding boxes* dengan *labels*

Menggunakan perintah if untuk memeriksa jika setidaknya ada satu objek yang terdeteksi setelah NMS. Dengan mendapatkan *bounding box* saat ini, lebar, dan tingginya. Kemudian mempersiapkan warna untuk *bounding box* saat ini dan mengkonversi dari *array* numpy. Setelah mendapatkan warna, selanjutnya menggambar *bounding box* pada *frame*. Setelah itu, mempersiapkan teks dengan label dan nilai *confidence* untuk *bounding box* saat ini dan menampilkan hasilnya.

```
if len(results) > 0:
   for i in results.flatten():
       x_min, y_min = bounding_boxes[i][0], bounding_boxes[i][1]
       box_width, box_height = bounding_boxes[i][2], bounding_boxes[i][3]
       colour_box_current = colours[classIDs[i]].tolist()

       cv2.rectangle(frame, (x_min, y_min),(x_min + box_width, y_min +
                     box_height), colour_box_current, 2)
       text_box_current = '{}: {:.4f}'.format(labels[int(classIDs[i])],
                                              confidences[i])
       cv2.putText(frame, text_box_current, (x_min, y_min - 5),
                   cv2.FONT_HERSHEY_SIMPLEX, 0.5, colour_box_current, 2)
```

Gambar 4.19 Kode program mendapatkan *bounding boxes* dengan *labels*

j. Menyimpan video proses deteksi

Pada bagian ini, akan menyimpan hasil prediksi masukan video dengan mengecek '*writer*', kemudian '*writer*' akan diinisialisasi pada iterasi pertama dari perulangan. Mengeluarkan perkiraan waktu yang dibutuhkan untuk proses video. Dan mengeluarkan hasil akhir dari proses deteksi pada masukan video.

```
if writer is None:
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    writer = cv2.VideoWriter('resultcctv8juli.mp4', fourcc, 30,
                             (frame.shape[1], frame.shape[0]), True)
writer.write(frame)

# Printing final results
print()
print('Total number of frames', f)
print('Total amount of time {:.5f} seconds'.format(t))

# Releasing video reader and writer
video.release()
writer.release()
```

Gambar 4.20 Kode program menyimpan video deteksi

```
Total number of frames 237
Total amount of time 139.25257 seconds
```

Gambar 4.21 Contoh hasil keluaran akhir deteksi

4.2 Pengujian

Pada langkah ini, dipaparkan hasil dari pengujian dari implementasi algoritma YOLO.

4.2.1 Pengujian pada performansi model YOLOv3

Pengujian dilakukan bertujuan untuk mengetahui tingkat keakurasian dari model *weight* YOLOv3 yang sudah di-*training* menjadi YOLOv3_custom. Pengujian performa *training* data yolov3_custom menggunakan data *training* sebanyak 425 gambar dan data validasi sebanyak 106 gambar dengan total gambar sebanyak 531 gambar menggunakan lima kelas, yaitu mobil, sepeda motor, becak, truk, dan bus. Data validasi yang sudah diberi anotasi diolah sebagai ground-truth box dibandingkan dan predicted box yang kemudian menghasilkan *confusion matrix*, kemudian dikalkulasi untuk mendapatkan nilai *precision*, *recall*, *average precision* (AP), *F1-score*, *intersection over union* (IoU), dan *mean average precision* yang sudah dijelaskan pada BAB II.

Hasil dari *training* data dapat dilihat pada Tabel 4.3 hasil *training* pada kelas mobil menunjukan jumlah *True Positive* sebesar 315 jauh lebih besar dibandingkan jumlah *False Positive* sebesar 12 dengan nilai AP 99,88%. Pada kelas sepeda motor menunjukan jumlah *True Positive* sebesar 166 jauh lebih besar dibandingkan jumlah *False Positive* sebesar 4 dengan nilai AP 97,79%. Kelas becak menunjukan jumlah *True Positive* sebesar 6 jauh lebih besar dibandingkan jumlah *False Positive* sebesar 0 dengan nilai AP 100%. Kelas truk dan bus menunjukan jumlah *True Positive* sebesar 8 jauh lebih besar dibandingkan jumlah *False Positive* sebesar 0 dengan nilai AP pada kelas truk 100% dan kelas bus 99,09%. Hal ini mencerminkan sistem telah dapat mendeteksi objek dengan benar dari dataset yang telah dilatih. YOLOv3_custom membutuhkan 4 detik waktu proses untuk mendeteksi 5 kelas. Kemudian, diperoleh presisi sistem mengklasifikasikan objek dengan tepat mencapai 97%. Sementara, recall mengukur kemampuan model untuk menemukan seluruh objek positif mencapai nilai 98%. Pada pengujian dilakukan pula pengukuran parameter nilai precision dan recall untuk menentukan keseimbangan nilai diperoleh nilai F1-score sebesar 97%.

Tabel 4.3 Pengujian pada *training data*

Load Model		Yolov3_custom
Mobil	AP	99,88%
	TP	315
	FP	12
Sepeda Motor	AP	97,79%
	TP	166
	FP	4
Becak	AP	100%
	TP	6
	FP	0
Truk	AP	100%
	TP	8
	FP	0
Bus	AP	99,09%
	TP	8
	FP	0
FN		10
Waktu pemrosesan		4 detik
Recall		0.98
Precision		0,97
F1-Score		0,97
IoU		79,12%
mAP@0,5		99,35%

Pada Tabel 4.4 menunjukkan menunjukkan perbandingan akurasi waktu beberapa metode untuk pendeteksian objek. Metode YOLOv3 dengan masukan 608 x 608 menghasilkan nilai mAP dua persen lebih rendah dari metode FPN FRCN, yaitu 57,9% untuk YOLOv3-608, dan 59,1% untuk FPN FRCN, tetapi metode YOLOv3-608 dapat memproses masukan lebih cepat selama 51 ms dibandingkan FPN FRCN selama 172 ms. mAP adalah suatu metrik untuk mengukur akurasi objek detektor dan semakin tinggi mAP berarti pendeteksi objek semakin akurat. Beberapa metode yang lain menghasilkan nilai mAP yang baik, namun waktu pemrosesan lebih lama dibandingkan metode YOLOv3 dengan masukan 608x608, 416x416, dan 320x320 (Redmon & Farhadi, 2018). Sedangkan pada penelitian ini menggunakan *pre-trained weights* model YOLOv3 dengan teknik *transfer learning* menghasilkan nilai mAP sebesar 99,35% dengan waktu pemrosesan selama 4 detik. Hal ini menunjukkan metode YOLOv3 bekerja dengan baik untuk melakukan deteksi objek.

Tabel 4.4 Perbandingan kecepatan/akurasi dari beberapa model deteksi objek (Redmon & Farhadi, 2018)

Metode	mAP@50	Waktu (ms)
SSD321	45,4%	61
DSSD321	46,1%	85
R-FCN	51,9%	85
SSD513	50,4%	125
DSSD513	53,3%	156
FPN FRCN	59,1%	172
RetinaNet-50-500	50,9%	73
RetinaNet-101-500	53,1%	90
RetinaNet-101-800	57,5%	198
YOLOv3-320	51,5%	22
YOLOv3-416	55,3%	29
YOLOv3-608	57,9%	51

4.2.2 Pengujian pada hasil deteksi

Pada pengujian deteksi, menggunakan *pre-trained weights* dengan lima kelas dilakukan untuk mengetahui keakuratan model dalam mendeteksi kelas yang sudah dilatih. Pengujian dapat dilakukan menggunakan program yang telah dibangun dan menggunkan model *weights* yang telah dilatih. Program kemudian di-*run* dengan masukan berupa video rekaman CCTV dari *Website* ATCS Dishub Kota Medan dengan tanggal yang berbeda-beda untuk mendapatkan seberapa akurat model *weights* dapat mendeteksi kendaraan. Keluaran dari hasil deteksi adalah bounding box pada objek yang terdeteksi dan nilai confidence seperti pada Tabel 4.5.

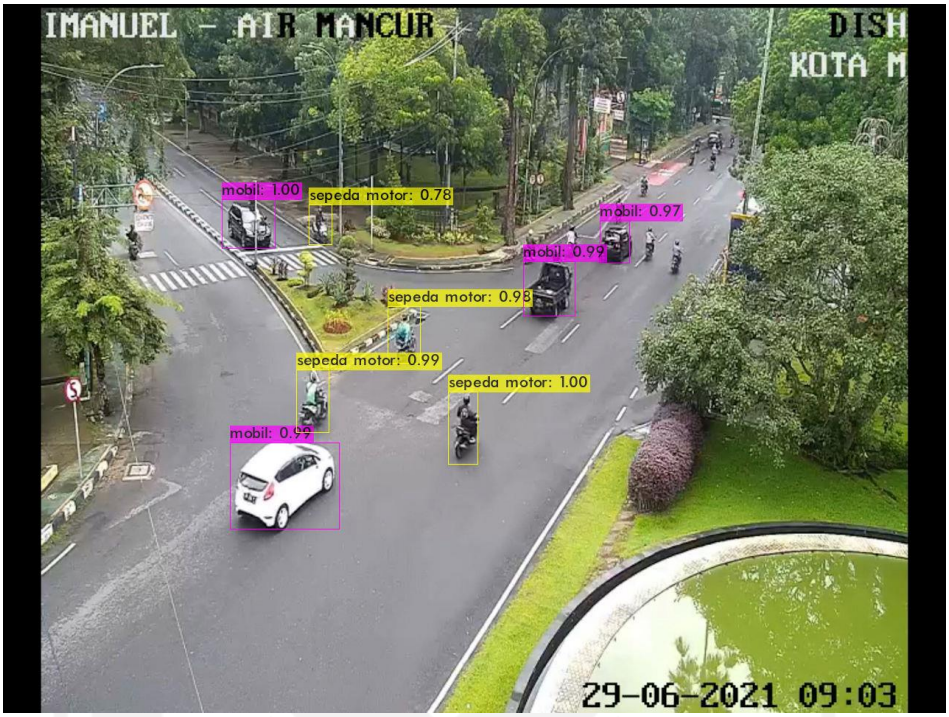
Tabel 4.5 Pengujian pada hasil deteksi

Tanggal	Mobil	Sepeda motor	Bus	Truk	Becak	Rata-rata FPS	Rata-rata waktu pemrosesan
14-04-2021	81,04%	89,88%	-	-	-	35 FPS	4 detik
29-06-2021	87,13%	86,91%	-	-	-	33 FPS	4 detik
07-07-2021	89,56%	84,69%	-	-	-	33 FPS	4 detik
08-07-2021	95,47%	84,30%	90%	86,60%	87%	34 FPS	4 detik

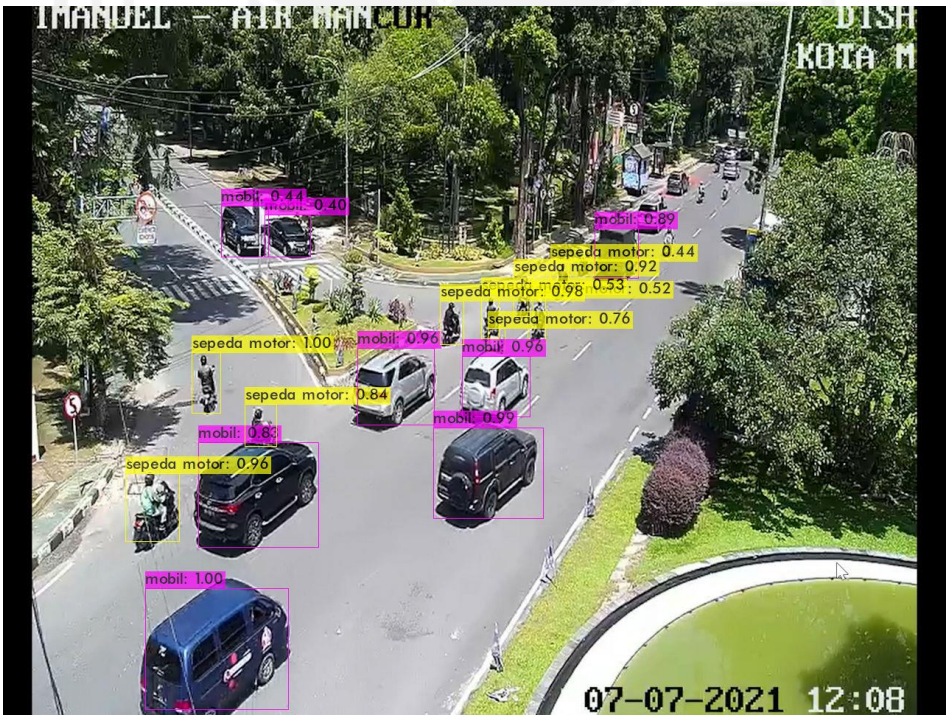
Pada masukan tanggal 14-04-2021 didapatkan rata-rata nilai *confidence* pada kelas mobil sebesar 81,04% dan kelas sepeda motor sebesar 89,88%. Pada masukan tanggal 29-06-2021 didapatkan rata-rata nilai *confidence* pada kelas mobil sebesar 87,13%, dan kelas sepeda motor sebesar 86,91%. Pada masukan tanggal 07-07-2021 didapatkan rata-rata nilai *confidence* pada kelas mobil sebesar 89,56%, dan pada kelas sepeda motor 84,69%. Pada masukan tanggal 08-

07-2021 didapatkan rata-rata nilai *confidence* pada kelas mobil sebesar 95,47%, kelas sepeda motor sebesar 84,30%, kelas bus sebesar 90%, kelas truk sebesar 86,60%, dan kelas becak sebesar 87%. Adapun saat menjalankan program, hasil dari deteksi menggunakan masukan video menghasilkan nilai *confidence* yang berbeda-beda pada setiap *frame*-nya, hal ini disebabkan objek berpindah posisi.

Gambar 4.22 Hasil deteksi CCTV tanggal 14-04-2021



Gambar 4.23 Hasil deteksi CCTV tanggal 29-06-2021



Gambar 4.24 Hasil deteksi CCTV tanggal 07-07-2021

Gambar 4.25 Hasil deteksi CCTV tanggal 08-07-2021

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil penelitian yang sudah dilakukan, dapat ditarik kesimpulan sebagai berikut:

a. Hasil dari pendeteksian kendaraan pada kelas mobil, sepeda motor, bus, truk, dan becak menggunakan algoritma YOLO dapat dinilai bekerja dengan baik. Hasil dari deteksi menggunakan masukan video menghasilkan nilai *confidence* yang berbeda-beda pada setiap *frame*-nya, hal ini disebabkan objek berpindah posisi.
b. Hasil *training* data pada setiap kelas menunjukan jumlah *True Positive* jauh lebih besar dibandingkan jumlah *False Positive* hal ini menunjukan sistem telah dapat mendeteksi objek dengan baik. Nilai AP pada mobil sebesar 99,88%, pada sepeda motor 97,79%, becak sebesar 100%, truk sebesar 100%, dan bus 99,09%. Sedangkan nilai mAP sebesar 99,35% dengan waktu pemrosesan selama 4 detik. Semakin tinggi nilai mAP maka pendeteksi objek semakin akurat.
c. Kualitas video dapat mempengaruhi pendeteksian karena semakin tinggi kualitas video maka hasil klasifikasi dan *bounding box* semakin tinggi pula.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, penulis memberikan beberapa saran untuk penelitian di masa depan, antara lain:

a. Memperbanyak data pada setiap kelas yang akan digunakan untuk *training* untuk memaksimalkan tingkat akurasi model *weights*.
b. Membuat tampilan antarmuka untuk mempermudah dalam melakukan pengoperasian.
c. Video harus berkualitas tinggi agar memudahkan untuk mendeteksi objek. Karena, jika video berkualitas rendah dapat mempengaruhi hasil pendeteksian. Untuk mengatasi hasil video dengan kualitas rendah dapat dilakukan saat proses *render* video dengan memilih resolusi yang tinggi.

DAFTAR PUSTAKA

Alamsyah, D. (2017). Pengenalan Mobil pada Citra Digital Menggunakan HOG-SVM. *Jatisi*, *1*(2), 162–168.

Arinaldi, A., Pradana, J. A., & Gurusinga, A. A. (2018). Detection and classification of vehicles for traffic video analytics. *Procedia Computer Science*, *144*, 259–268. https://doi.org/10.1016/j.procs.2018.10.527

Bathija, A. (2019). Visual Object Detection and Tracking using YOLO and SORT. *International Journal of Engineering Research and Technology (IJERT)*, *8*(11), 705–708. Retrieved from https://www.ijert.org

Budiarjo, D. D. (2020). *IMPLEMENTASI SISTEM CERDAS PADA OTOMATISASI PENDETEKSIAN JENIS KENDARAAN DI JALAN RAYA*.

Dewi, S. R. (2018). Deep Learning Object Detection Pada Video Menggunakan Tensorflow dan Convolutional Neural Network. *Deep Learning Object Detection Pada Video Menggunakan Tensorflow Dan Convolutional Neural Network*, 1–60. Retrieved from https://dspace.uii.ac.id/bitstream/handle/123456789/7762/14611242_Syarifah Rosita Dewi_Statistika.pdf?sequence=1

Fachrie, M. (2020). A Simple Vehicle Counting System Using Deep Learning with YOLOv3 Model. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *4*(3), 462–468. https://doi.org/10.29207/resti.v4i3.1871

Fadlia, N., & Kosasih, R. (2019). Klasifikasi Jenis Kendaraan Menggunakan Metode Convolutional Neural Network (Cnn). *Jurnal Ilmiah Teknologi Dan Rekayasa*, *24*(3), 207–215. https://doi.org/10.35760/tr.2019.v24i3.2397

Harahap, M., Elfrida, J., Agusman, P., Rafael, M., Abram, R., Andrianto, K., … Kendaraan, D. (2019). Sistem Cerdas Pemantauan Arus Lalu Lintas Dengan YOLO (You Only Look Once v3). *Seminar Nasional APTIKOM*, 2019.

Hosang, J., & May, C. V. (n.d.). *Learning non-maximum suppression*.

Hutauruk, J. S. W., Matulatan, T., & Hayaty, N. (2020). Deteksi Kendaraan secara Real Time menggunakan Metode YOLO Berbasis Android. *Jurnal Sustainable: Jurnal Hasil Penelitian Dan Industri Terapan*, *9*(1), 8–14. https://doi.org/10.31629/sustainable.v9i1.1401

Irwan, F., Putrada, A. G., & Prabowo, S. (2019). Live Monitoring Parkiran Mobil Menggunakan Cctv Dan Computer Vision. *EProceedings …*, *6*(2), 9166–9173. Retrieved

from https://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/viewFile/9936/9793

Jalled, F., & Voronkov, I. (2016). *Object Detection using Image Processing*. 1–6. Retrieved from http://arxiv.org/abs/1611.07791

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9905 LNCS*, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2

Nurhawanti, R. (2019). *Sistem Pendeteksi Sepeda Motor Pelanggar Marka Jalan Menggunakan Metode Convolutional Neural Networks (CNNs)*.

Pratama, A. F. (2020). *Deteksi Plat Nomor Kendaraan Bergerak Berbasis Metode You Only Look Once (YOLO)*. (Undergraduate Thesis). Retrieved from https://repository.its.ac.id/id/eprint/79534

Pribadi, B., & Naseer, M. (2016). Sistem Klasifikasi Jenis Kendaraan Melalui Teknik Olah Citra Digital. *Setrum : Sistem Kendali-Tenaga-Elektronika-Telekomunikasi-Komputer*, *3*(2), 103. https://doi.org/10.36055/setrum.v3i2.505

Rahmawati, L., Fisika, D., Sains, F., & Diponegoro, U. (2017). Rancang bangun penghitung dan pengidentifikasi kendaraan menggunakan Multiple Object Tracking. *Youngster Physics Journal*, *6*(1), 70–75.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 779–788. https://doi.org/10.1109/CVPR.2016.91

Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*. Retrieved from http://arxiv.org/abs/1804.02767

Salim, A. (2020). *Estimasi Kecepatan Kendaraan Melalui Video Pengawas Lalu Lintas Menggunakan Parallel Line Model*. 1–100.

Shianto, K. A., Gunadi, K., & Setyati, E. (2019). Deteksi Jenis Mobil Menggunakan Metode YOLO Dan Faster R-CNN. *Jurnal Infra*, *7*(1), 157–163.

Syuhaila, F. Q. (2020). *Sistem Pengenalan Plat Nomor Otomatis Menggunakan YOLOV3 Dengan Framework Keras*. (Undegraduate Thesis).

Umar, Y., Hanafi, Mardi, S., Nugroho, Susiki, & Rachmadi, R. F. (2020). *Deteksi Penggunaan*

Helm Pada Pengendara.

Waliulu, R. F. (2018). Deteksi dan Penggolongan Kendaraan dengan Kalman Filter dan Model Gaussian di Jalan Tol. *Jurnal Sistem Informasi Bisnis*, *8*(1), 1. https://doi.org/10.21456/vol8iss1pp1-8

LAMPIRAN