SISTEM CERDAS DETEKSI PELAT NOMOR UNTUK PELANGGARAN HELM DAN MASKER

by Rohmat Syamsul Huda

Submission date: 10-Aug-2022 07:50PM (UTC+1000)
Submission ID: 1880966428
File name: SKRIPSI_BAB_1_-_5.pdf (1.73M)
Word count: 8425
Character count: 49648

BAB I

PENDAHULUAN

Dalam bab ini akan dijelaskan tentang latar belakang, identifikasi masalah, rumusan masalah pada penelitian, batasan masalah yang dibuat agar lebih terarah, tujuan penelitian, manfaat dan kegunaan penelitian yang dibuat, metode penelitian yang dipakai serta jadwal penelitian yang dilakukan.

A. Latar Belakang

Pada akhir Desember tahun 2019 dunia dikejutkan dengan penemuan virus baru yaitu *coronavirus* yang kemudian disebut sebagai *Coronavirus disease* 2019 (COVID-19) (Yuliana, 2020). COVID-19 dapat menular dari manusia ke manusia melalui *droplet* (percikan pernafasan atau lendir), kontak dengan *droplet* dan melalui penularan *fekal-oral* yang dapat menyebabkan *Acute Respiratory Distress Syndrome* (ARDS), gagal ginjal, disfungsi koagulasi, kegagalan organ dan kematian (Harahap, 2020).

Berdasarkan data yang telah dihimpun sampai tanggal 1 November 2021 dipastikan terdapat 223 negara yang terjangkit COVID-19 dengan jumlah kasus mencapai 247.352.866 di seluruh dunia. Sementara di Indonesia, jumlah kasus mencapai 4.244.358 dengan 143.405 kasus di antaranya meninggal (worldometers, 2021). Kemudian pemerintah menetapkan Kedaruratan Kesehatan Masyarakat COVID-19 melalui Keputusan Presiden Nomor 11 Tahun 2020 (Keppres, 2020), disusul surat edaran yang dikeluarkan oleh Kemenkes RI Nomor HK.02.02/I/385/2020 tentang penggunaan masker dan penyediaan sarana cuci tangan pakai sabun (CTPS) untuk mencegah penularan

COVID-19 dengan melaksanakan gerakan "Semua Pakai Masker" dan penyediaan sarana CTPS sebagai upaya pencegahan penularan COVID-19 (Kemenkes, 2021). Sehingga hal tersebut berimbas terhadap aktivitas berkendara sepeda motor.

Sepeda motor merupakan kendaraan bermotor favorit masyarakat Indonesia dengan jumlah 112.771.136 dari total 133.617.012 (BPS, 2021b). Disisi lain, banyaknya penggunaan sepeda motor menimbulkan risiko rawan terjadinya kecelakaan lalu lintas. Data jumlah kecelakaan pada tahun 2019 adalah 116.411 kasus kecelakaan dengan 25.671 di antaranya merupakan kasus korban meninggal (BPS, 2021a). Selain demi keselamatan pengendara, kewajiban memakai helm sesuai SNI tertuang dalam Undang - Undang Republik Indonesia Nomor 22 Tahun 2009 Tentang Lalu Lintas dan Angkutan Jalan pasal 57 ayat 1 dan 2 (Republik Indonesia, 2009). Dengan demikian penting bagi pengendara sepeda motor memakai helm dan masker secara bersamaan.

Saat ini untuk mengatasi pelanggaran lalu lintas, Indonesia sudah memiliki sistem E-TLE (*Electronic Traffic Law Enforcement*) (Polda Metro Jaya, 2021). Sistem E-TLE mampu untuk mendeteksi pengendara yang tidak memakai helm, sementara untuk pelanggaran masker biasanya masih menggunakan cara tradisional yang melibatkan manusia untuk pendataan sehingga mengurangi efisiensi dan memakan durasi.

Pada penelitian yang berjudul "*Mask and Helmet Detection in Two-Wheelers using YOLOv3 and Canny Edge Detection*" telah berhasil mendeteksi

objek menggunakan *YOLOv3*, objek tersebut yakni sepeda motor, helm, dan masker dengan akurasi deteksi sepeda motor 95.92%, akurasi deteksi helm 90.84% dan akurasi deteksi masker 93.35%. Sementara untuk deteksi pelat nomor sepeda motor menggunakan metode *canny edge detection* (deteksi tepi *canny*) (Maliye dkk., 2021). Kemudian pada penelitian yang berjudul "*Real-time Automatic License Plate Recognition System using YOLOv4*" sudah berhasil mendeteksi pelat nomor kendaraan Korea dengan akurasi 94.25% di mana deteksi karakternya juga menggunakan *YOLOv4* dikarenakan penggunaan huruf lokal pada pelat nomor Korea. Dengan menghilangkan langkah pengenalan *ROI* (*Region Of Interest*) jumlah komputasi akan berkurang sehingga berdampak pada meningkatnya kecepatan. *YOLOv4* mampu mendeteksi pelat nomor dengan baik yang sebelumnya tidak terdeteksi oleh *YOLOv3* atau *SSD* (*Single Shot MultiBox Detector*) (Sung dkk., 2020).

Berdasarkan kajian di atas peneliti tertarik untuk meneliti deteksi pelanggaran masker dan helm pada pengendara sepeda motor menggunakan *YOLOv4* dengan identifikasi pelat nomor sepeda motor menggunakan *Tesseract-OCR*. Penelitian ini sangat penting karena metode YOLO merupakan gebrakan dalam dunia *computer visison,* serta untuk menyongsong Indonesia emas tahun 2045 salah satunya adalah fokus pemerintah dalam pengembangan *Artificial Intelligence* (kecerdasan buatan) (BPPT, 2020).

B. Identifikasi Masalah

Berdasarkan latar belakang masalah yang dikemukakan di atas, masalah dapat di identifikasi sebagai berikut :

1. Banyaknya pelanggaran lalu lintas khususnya tidak memakai helm
2. Kewajiban memakai masker saat di luar ruangan
3. Operasi patuh masker dilakukan dengan cara tradisional
4. Sistem E-TLE belum dapat mendeteksi pelanggaran masker

C. Rumusan Masalah

Berdasarkan identifikasi masalah yang di paparkan di atas, maka rumusan masalahnya sebagai berikut :

1. Bagaimana mendeteksi masker, helm, pelat nomor dan sepeda motor pada pengendara sepeda motor ?
2. Bagaimana mengidentifikasi nomor polisi pada pelat nomor menjadi teks menggunakan Tesseract-OCR ?
3. Bagaimana kinerja metode YOLOv4 untuk mendeteksi masker, helm, pelat nomor dan sepeda motor pada pengendara sepeda motor ?

D. Batasan Masalah

Dalam penelitian ini penulis membatasi cakupan masalah yang akan dibahas. Batasan masalah pada penelitian ini adalah sebagai berikut:

1. Simulasi ini berfokus pada satu citra pengendara sepeda motor.
2. Helm yang digunakan adalah jenis helm *open face* (bagian wajah sampai dagu terbuka) dengan kaca berwarna transparan.
3. Masker yang digunakan adalah jenis masker bedah.
4. Pelat nomor sepeda motor yang digunakan berlatar hitam dan nomor polisi berwarna putih. Pelat terbaca dengan jelas agar diperoleh nomor polisi yang sesuai.

5. Pengambilan data uji dilakukan saat kondisi terang
6. Bahasa pemrograman yang digunakan adalah *python*

E. Tujuan Penelitian

Adapun tujuan penelitian sebagai berikut :

1. Untuk mendeteksi masker, helm, pelat nomor, dan sepeda motor pada pengendara sepeda motor
2. Untuk mengidentifikasi nomor polisi pada pelat nomor menjadi teks menggunakan Tesseract-OCR ?
3. Untuk mengetahui kinerja YOLOv4 dalam mendeteksi masker, helm, pelat nomor, dan sepeda motor pada pengendara sepeda motor.

F. Manfaat Dan Kegunaan

Penelitian ini bermanfaat sebagai langkah lanjutan dalam memutus persebaran virus serta deteksi pelanggaran lalu lintas dengan menerapkan kecerdasan buatan berbasis citra digital untuk menentukan alternatif penanganannya. Penelitian ini dapat digunakan sebagai alat bantu dalam meningkatkan ketertiban dalam berlalu lintas serta penanganan virus. Terakhir, penelitian ini bermanfaat sebagai referensi penggunaan metode YOLO dalam pengidentifikasian objek berbasis citra.

G. Metode Penelitian

Untuk pembuatan sistem ini, penulis menggunakan metode *waterfall* (air terjun). Beberapa tahapan yang penulis lakukan adalah sebagai berikut :

1. Studi Pustaka

   Studi Pustaka dilakukan penulis untuk mendapatkan dan memperoleh informasi sebanyak-banyaknya sebagai masukkan untuk pembuatan sistem. Termasuk pendalaman materi tentang tahap pengolahan citra, serta untuk deteksi citra menggunakan metode YOLO. Literatur yang digunakan yaitu berdiskusi, membaca buku, jurnal, artikel, dan situs yang membahas tentang materi tersebut.

2. Pengelompokan data

   Data yang digunakan dalam penelitian ini antara lain adalah citra sepeda motor, helm, masker dan pelat sepeda motor (4 objek). Data tersebut di dapatkan dari internet dengan perincian setiap objek tersebut berjumlah 150 data. Sehingga total data adalah 600 dengan ukuran 416 x 416 (ukuran yang di sarankan oleh pengembang YOLOv4).

3. Pembagian data

   Tahap pembagian data yaitu membagi keseluruhan data untuk proses *training* (pelatihan) dan *testing* (pengujian). Data yang digunakan untuk pelatihan sebanyak 568 citra, sementara untuk data untuk pengujian sebanyak 32. Sehingga perbandingan datanya adalah 568 : 32.

4. Anotasi data

   Tahap ini data citra dianotasikan dengan membuat *bounding box* (kotak pembatas) pada objek dan memberi kelas yang sesuai. Di mana dalam penelitian ini menggunakan format anotasi *darknet* YOLOv3 yang merepresentasikan kelas dan 4 koordinat (kelas objek, posisi x, posisi y,

lebar, tinggi) dari suatu citra. Untuk menganotasikan data, penulis menggunakan aplikasi *open source Labelimg* yang dapat di akses di https://github.com/tzutalin/labelImg.

5. *Coding*

   Pada tahap ini penulis mengonfigurasi program untuk melatih model yang dibuat berdasarkan data di atas. Program tersebut di akses dari *github* pengembang YOLOv4 https://github.com/AlexeyAB/darknet.

6. Pelatihan Model

   Pada tahap ini program melakukan *training* dan *testing* dengan ketentuan 2000 iterasi untuk setiap kelas sehingga proses *training* dan *testing* model membutuhkan 8000 iterasi (4 kelas x 2000). Pada setiap 1000 iterasi, program akan menghasilkan model sehingga total model 8.

7. Evaluasi Model

   Evaluasi model YOLOv4 dilakukan untuk mengetahui model dengan mAP (presisi rata – rata) atau F1-Score tertinggi kemudian memilih model tersebut untuk diintegrasikan dengan sistem.

8. Implementasi Sistem

   Menggunakan model yang sudah dipilih untuk diintegrasikan dengan sistem inti dengan melakukan *cloning* pada repositori *github* berikut https://github.com/hunglc007/tensorflow-yolov4-tflite.

9. Testing

   Menjalankan uji coba pada sistem menggunakan data citra uji yang di dalamnya memuat objek helm, masker, sepeda motor dan pelat nomor.

10. Evaluasi Sistem

Mengevaluasi kinerja dari sistem untuk mengetahui tingkat keakuratan prediksi objek serta mengevaluasi kinerja identifikasi.

11. Penulisan Laporan

Penyusunan laporan dilakukan setelah semua kegiatan selesai, laporan berisi tentang data - data yang diperoleh dari hasil studi pustaka sampai evaluasi sistem.

H. Jadwal Penelitian

Jadwal penelitian yang terstruktur dari penulis mulai dari awal sampai akhir penelitian seperti pada Tabel 1.1 berikut :

Tabel 1. 1 Jadwal penelitian

No.	Jenis Kegiatan	Bulan 1				Bulan 2				Bulan 3				Bulan 4				Bulan 5				Bulan 6			
	Minggu ke	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Studi Literatur																								
2	Pengelompokan Data																								
3	Pembagian Data																								
4	Anotasi data																								
5	*Coding*																								
6	Pelatihan Model																								
7	Evaluasi Model																								
8	Implementasi Sistem																								
9	Testing																								
10	Evaluasi Sistem																								
11	Penulisan Laporan																								

I. Sistematika Penulisan Laporan

Sistematika penulisan yang digunakan dalam penyusunan laporan tugas akhir ini dibagi menjadi beberapa bab dan sub-bab yang terstruktur, dengan kajian yang saling terkait dan berhubungan agar lebih mudah dipahami,

sehingga dapat menggambarkan dengan jelas sebuah sistem dan data yang akurat. Secara umum sistematika penulisannya adalah sebagai berikut :

Bab I : Pendahuluan

Bab ini berisi latar belakang masalah, identifikasi masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian serta sistematika penulisan.

Bab II : Tinjauan Pustaka

Pada bab ini membahas tentang teori – teori yang relevan dengan metode YOLO, serta membahas beberapa teori yang memiliki hubungan dengan pokok – pokok pembahasan.

Bab III : Analisa Dan Desain Sistem

Bab ini menguraikan penjelasan mengenai hal-hal dalam proses analisis sistem yang mencakup sistem lama, sistem yang diusulkan dan kebutuhan perangkat. Kemudian membahas desain sistem dan desain antar muka.

Bab IV : Implementasi dan Hasil

Bab ini berisi tentang implementasi suatu sistem dari tahapan – tahapan yang telah di tentukan, menguji hasil sistem serta mengevaluasi hasil dari pengujian tersebut.

Bab V : Penutup

Bab ini berisi tentang ringkasan hasil yang dirangkum dalam kesimpulan dan saran penulis untuk penelitian selanjutnya

BAB II

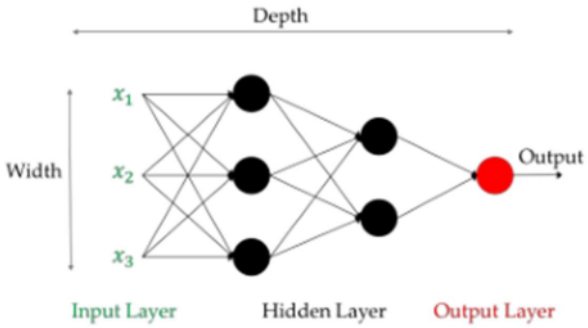
TINJAUAN PUSTAKA

Dalam bab ini akan dijelaskan tentang landasan teori dan kajian pustaka.

A. Landasan Teori

1. *Deep learning*

Deep learning adalah salah satu cabang *machine learning* yang menggunakan *Deep Neural Network (DNN)* untuk menyelesaikan permasalahan pada domain *machine learning* (Putra, 2020). Pada Gambar 2.1 *Depth* (kedalaman) mengacu pada banyaknya layer, sementara *width* (lebar) mengacu pada banyaknya unit pada layer.



Gambar 2. 1 Model *Deep Neural Network*
(Sumber: (Putra, 2020) halaman 144)

Dalam hal ini deep learning dapat menangani masalah yang lebih rumit seperti *computer vision* (kemampuan mesin untuk melihat objek dalam gambar) menggunakan *artificial neural network* (ANN) (Russell dan Norvig, 2021). Secara sederhana, *deep learning* meniru cara kerja otak

manusia melalui jaringan neuron (*neural network*) yang arsitekturnya sangat beragam. Salah satu algoritma *deep learning* adalah YOLO.

2. YOLOv4

   a. Pendahuluan

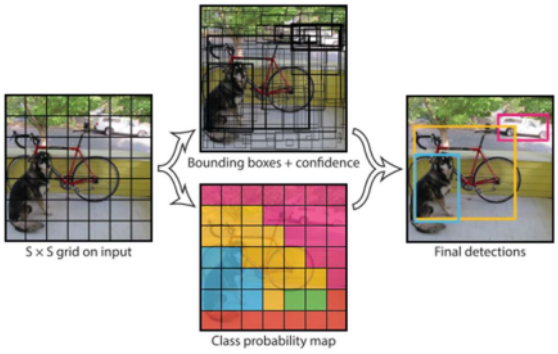
      YOLO (*You Only Look Once*) pertama kali diciptakan oleh Joseph Redmon pada tahun 2015. YOLO adalah sistem deteksi objek secara *real time* berdasarkan CNN (*Convolutional Neural Network*) (Redmon dkk., 2016). Kemudian YOLOv2 dan YOLOv3 dikembangkan oleh Joseph Redmon dan Ali Farhadi. YOLOv2 rilis pada konferensi CVPR 2017 yang telah ditingkatkan akurasi dan kecepatan algoritmanya (Redmon dan Farhadi, 2017). Kemudian pada April 2018, keduanya merilis YOLOv3 yang memiliki *performance* atau kinerja yang semakin meningkat pada deteksi objek (Redmon dan Farhadi, 2018).

      Setelah keduanya memutuskan untuk berhenti melakukan penelitian *computer vision*, YOLOv4 selanjutnya dikembangkan oleh 3 orang *author* yaitu Alexey Bochkovskiy, Chien-Yao Wang dan Hong-Yuan Mark Liao. YOLOv4 rilis pada 24 April 2020 ini merupakan peningkatan penting dari YOLOv3, implementasi arsitektur baru di *Backbone* dan modifikasi di *Neck* telah meningkatkan mAP (*mean Average Precision*) sebesar 10% dan jumlah FPS (*Frame per Second*) sebesar 12% (Bochkovskiy dkk., 2020).

b. Ide Dasar

Menurut (Redmon dkk., 2016), Ide dasar dari algoritma YOLO adalah sebagai berikut :

> YOLO membagi gambar atau video yang di *input* menjadi S X S *grid*. Jika titik tengah koordinat pada GT (*Ground Truth*) suatu objek jatuh ke dalam *grid*, maka *grid* tersebut bertanggung jawab untuk mendeteksi suatu objek. YOLO memprediksi *bounding box* dari objek yang berada di semua *grid*, kemungkinan lokasi dan probabilitas dari semua *class* pada satu waktu.
>
> Dengan demikian YOLO menyelesaikan semua masalah dalam satu kali proses saja. Ide dasar YOLOv1 seperti pada Gambar 2.2.



Gambar 2. 2 Ide dasar YOLOv1
(Sumber: (Redmon dkk., 2016))

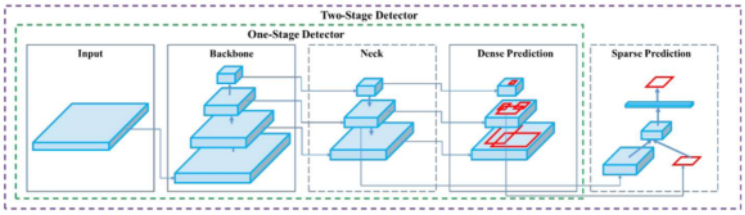
c. Detektor Satu Tahap

Detector Modern umumnya dibagi menjadi 2 yakni detektor satu tahap (*One Stage Detektor*) dan detektor dua tahap (*Two Stage Detector*). Menurut (Bochkovskiy dkk, 2020), YOLO termasuk kategori detektor satu tahap dengan penjelasan sebagai berikut :

> YOLO merupakan detektor satu tahap, di mana algoritma ini mampu mendeteksi objek tanpa perlu langkah awal (deteksi dan klasifikasi). Keuntungan penggunaan detektor satu tahap adalah

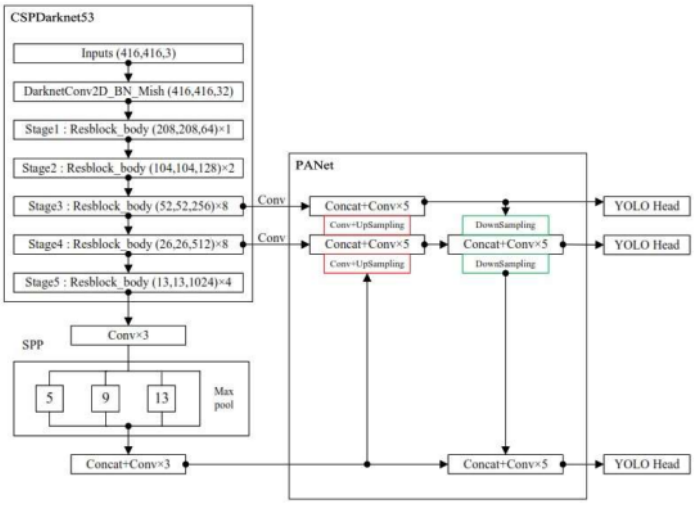
kemampuan prediksinya sangat cepat sehingga memungkinkan penggunaan waktu nyata.

Dalam detektor satu tahap terdapat 3 komponen penting, yakni *backbone* (blok tulang punggung*)*, *neck* (blok leher) dan *dense prediction* (blok prediksi padat atau kepala) seperti pada Gambar 2.3. Blok tulang punggung bertujuan untuk mengekstrak fitur gambar, kemudian blok leher digunakan untuk memperoleh fitur yang lebih efektif (menghasilkan piramida fitur) dan blok kepala berguna untuk memprediksi kelas (klasifikasi).



Gambar 2. 3 Arsitektur deteksi objek
(Sumber: (Bochkovskiy dkk., 2020))

d. Arsitektur YOLOv4



Gambar 2. 4 Arsitektur YOLOv4

YOLOv4 menggunakan CSPDarknet53 sebagai *backbone*, kemudian pada *Neck* menggunakan SPP + PAN dan *Head* menggunakan YOLOv3 untuk *Dense Prediction* (Bochkovskiy dkk., 2020). Arsitektur YOLOv4 seperti pada Gambar 2.4.

1) *Backbone* (blok tulang punggung)

YOLOv4 menggunakan CSPDarknet53 sebagai *Backbone* karena dapat memisahkan fitur konteks yang paling signifikan dan hampir tidak ada pengurangan kecepatan operasi jaringan *network* (Bochkovskiy dkk., 2020).

Menurut (Wang dkk., 2020), *CSP (Cross Stage Partial connections)* adalah ide membagi lapisan *layer* menjadi 2 bagian. Satu lapisan yang akan melalui blok konvolusi dan satunya tidak. Kemudian akan dibandingkan hasilnya. Kemudian (Redmon dan Farhadi, 2018) menyatakan Darknet53 adalah *Backbone* yang digunakan dalam YOLOv3. Angka 53 mengacu pada banyaknya layer konvolusi pada jaringan tersebut.

Menurut (Chen dkk., 2020), blok kepala YOLOv4 adalah sebagai berikut :

> *Darknet* berisi 5 modul residu dan CSPDarknet53 menggunakan konvolusi 1 × 1 untuk membagi lapisan saluran peta fitur *input* menjadi dua sebelum ke setiap jaringan residu, dan menambahkan CSP setelah setiap modul residu besar.

2) *Neck* (blok leher)

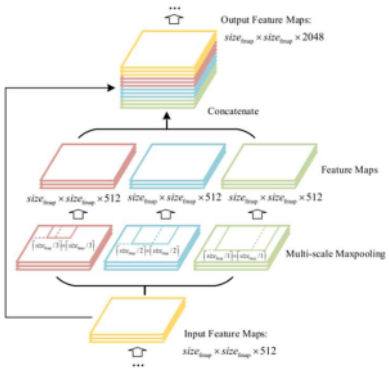
Leher YOLOv4 menggunakan SPP (*Spatial Pyramid Pooling*) dan *Path Aggregation Network* (PAN) yang dimodifikasi untuk mengumpulkan informasi agar mendapatkan akurasi yang lebih tinggi (Bochkovskiy dkk., 2020), blok leher seperti pada Gambar 2.5.

Menurut (Bochkovskiy dkk, 2020), proses yang terjadi pada blok leher YOLOv4 adalah sebagai berikut :

> SSP menggunakan 4 ukuran kernel geser yang berbeda yakni (1 × 1), (5 × 5), (9 × 9) dan (13 × 13) untuk mengonvolusikan kandidat gambar yang kemudian menerapkan *Multi-scale Maxpooling* untuk mendapatkan dimensi yang sama dari peta fitur.

Kemudian (Chen dkk., 2020), menjabarkan SPP dan PANet sebagai berikut :

> SPP memungkinkan ukuran spasial dari setiap kandidat peta untuk dipertahankan, dan kemudian menghubungkan peta fitur dengan ukuran inti yang berbeda sebagai *output*, dengan *output* berupa peta fitur ukuran tetap. PANet mengusulkan ROIPooling (*Region of Interest Pooling*) yang lebih fleksibel yang dapat mengekstrak dan mengintegrasikan fitur pada berbagai skala.



Gambar 2. 5 *Neck* (blok leher)

3) *Head* (blok kepala)

Peta fitur dari keluaran skala yang berbeda oleh PANet disambung, kemudian setelah melakukan operasi konvolusi, 3 kepala skala yang berbeda dapat diperoleh (Chen dkk., 2020).

Ukuran masukkan kepala saat pelatihan (416 × 416) adalah sebagai berikut [(52 × 52 * 3 * (4 + 1 +(kelas=4))), (26 × 26 × 3 * (4 + 1 + (kelas=4))), (13 × 13 × 3 * (4 + 1 + (kelas=4)))]. Penjelasan (Redmon dan Farhadi, 2018) tentang ukuran kepala saat pelatihan adalah sebagai berikut :

> 4 adalah nilai koordinat kotak kalibrasi, 1 adalah tingkat kepercayaan kotak kalibrasi, kelas adalah jumlah kategori, setiap kepala berisi 3 kotak pembatas, dan 3 kepala skala yang berbeda dapat digunakan untuk mendeteksi objek dari ukuran yang berbeda.

Dengan demikian proses pada kepala YOLOv4 sama dengan YOLOv3. Penjelasan (Bochkovskiy dkk., 2020), Proses pada blok kepala YOLOv4 sebagai berikut :

> jaringan mendeteksi koordinat kotak pembatas (*x,y,w,h*) serta skor kepercayaan untuk suatu kelas dengan membuang semua kotak prediksi yang skor kepercayaannya berada di bawah nilai ambang batas tertentu (melakukan proses *Non-Maximum Supression* dengan pertimbangan *Intersection Over Union*).

e. *Framework Darknet*

Framework sendiri adalah sebuah kerangka kerja yang digunakan untuk mempermudah para *developer software* dalam membuat dan mengembangkan aplikasi. *Framework* berisikan dasar perintah dan fungsi yang umumnya digunakan untuk membangun sebuah *software*

sehingga diharapkan *software* tersebut dapat dibangun dengan lebih cepat dengan struktur yang rapi.

Kemudian (Redmon dkk., 2016), mendefinisikan *framework Darknet* sebagai berikut

> *Darknet* adalah *framework neural network* yang bersifat *open source* yang ditulis dalam Bahasa C dan CUDA. *Darknet* sangat cepat, proses penginstalan yang mudah dan mendukung komputasi dengan CPU maupun GPU. *Darknet* memiliki banyak fitur-fitur yang telah dikembangkan untuk proses *machine learning*, seperti *object detection* dan *classification*.

3. Tesseract-OCR

Menurut (Mohammad dkk., 2014), *Optical character recognition* adalah metode untuk mengubah gambar huruf menjadi karakter ASCII yang dimengerti oleh komputer. Gambar huruf tersebut dapat berupa dokumen hasil *scan*, halaman web hasil *print screen*, foto, dan lain-lain

Sejak 2006, *Tesseract* dikembangkan oleh Google dengan lisensi Apache 2.0 yang berarti *library Optical Character Recognition* (OCR) tersebut bebas digunakan (*open source*) (Smith, 2007). Untuk dapat membaca teks dalam pelat nomor secara akurat, model *tesseract* harus dilatih terlebih dahulu menggunakan jenis *font* yang digunakan pada pelat nomor.

4. Masker bedah

Menurut (Cohen dan Birkner, 2012), Masker adalah perlindungan pernafasan yang digunakan untuk melindungi pemakai dari menghirup zat berbahaya atau racun di udara.

Menurut (Kemenkes, 2011), masker bedah di definisikan sebagai berikut :

> Masker bedah adalah masker yang biasa digunakan oleh petugas kesehatan di pelayanan kesehatan. Masker bedah terbuat dari bahan sintetik yang dapat memberikan perlindungan dari tetesan partikel berukuran besar (>5 µm) yang dapat disebarkan melalui batuk atau bersin ke orang yang berada di dekat pasien (kurang dari 1 meter).

Sejak abad ke20, masker bedah tidak hanya digunakan saat operasi, namun juga digunakan oleh petugas kesehatan dan orang sakit untuk mencegah penyebaran infeksi ke orang lain (MacIntyre dan Chughtai, 2015). Contoh masker bedah seperti pada Gambar 2.6 berikut :

Gambar 2. 6 Masker bedah

B. Kajian Pustaka

Kajian Pustaka adalah kegiatan yang meliputi mencari, membaca, dan menelaah laporan - laporan penelitian dan bahan Pustaka yang memuat teori - teori yang relevan dengan penelitian yang akan dilakukan. Penelitian terdahulu yang menjadi referensi dalam penelitian ini adalah sebagai berikut.

1. Penelitian yang dilakukan oleh (Maliye dkk., 2021) "*Mask and Helmet Detection in Two-Wheelers using YOLOv3 and Canny Edge Detection*". Penelitian ini bertujuan untuk mendeteksi pengendara sepeda motor yang tidak menggunakan helm atau masker menggunakan YOLOv3 untuk

selanjutnya dilakukan pengenalan pelat nomor sepeda motor menggunakan deteksi tepi *canny*. Telah berhasil mendeteksi objek dengan akurasi deteksi sepeda motor=95,92%, helm=90,84% dan masker=93,36%. Perbedaan dengan penelitian yang dilakukan adalah, jika sebelumnya menggunakan YOLOv3 maka di sini menggunakan YOLOv4 dan identifikasi pelat dilakukan menggunakan Tesseract-OCR

2. Penelitian yang dilakukan oleh (Sung dkk., 2020)" *Real-time Automatic License Plate Recognition System using YOLOv4*". Penelitian ini bertujuan untuk mendeteksi pelat kendaraan Korea serta deteksi karakter pelat menggunakan YOLOv4. YOLOv4 mampu mendeteksi pelat dengan baik yang sebelumnya tidak terdeteksi oleh YOLOv3 atau SSD (*Single Shot MultiBox Detector*) dengan akurasi deteksi pelat = 94,25%. Perbedaan dengan penelitian yang dilakukan adalah, jika penelitian sebelumnya menggunakan YOLOv4 untuk mendeteksi karakter pelat maka di sini menggunakan Tesseract-OCR.
3. Penelitian yang dilakukan oleh (Loey dkk., 2021) *"Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection*". Penelitian ini bertujuan untuk mendeteksi masker menggunakan YOLOv2 dengan ResNet-50 sebagai *backbone* menggunakan *Adam Optimizer.* Penelitian berhasil mendeteksi masker dengan mAP=81%. Perbedaan dengan penelitian sebelumnya menggunakan YOLOv2 untuk mendeteksi masker maka di sini menggunakan YOLOv4 dengan *backbone CSPDarknet53 (Mish Activation)*.

4. Penelitian yang dilakukan oleh (Wu dkk., 2019) "*Helmet detection based on improved YOLO V3 deep model*". Penelitian ini bertujuan untuk mendeteksi helm pelindung kepala pekerja menggunakan YOLOv3 dengan *backbone DenseNet.* Penelitian berhasil mendeteksi helm pelindung pekerja dan meningkatkan mAP sebesar 2,44% dari 95,15 menjadi 97,59%. Perbedaan dengan penelitian sebelumnya menggunakan YOLOv3 dengan memodifikasi *backbone* menggunakan *DenseNet* untuk mendeteksi helm pelindung pekerja maka di sini menggunakan YOLOv4 untuk mendeteksi helm pada pengendara sepeda motor.
5. Penelitian yang dilakukan oleh (Albert dkk., 2020) "Deteksi Helm pada Pengguna Sepeda Motor dengan Metode *Convolutional Neural Network*". Penelitian ini menggunakan YOLOv3 untuk mendeteksi sepeda motor kemudian mencari lokasi helm dengan menambahkan persentase tinggi 60% dari tinggi sepeda motor. Perbedaan dengan penelitian sebelumnya menggunakan YOLOv3 untuk mendeteksi sepeda motor dan menambahkan tinggi 60% untuk mendapatkan helm maka di sini sepeda motor dan helm di deteksi menggunakan YOLOv4

BAB III

ANALISA DAN DESAIN SISTEM

Dalam bab ini akan dijelaskan tentang analisa sistem, desain sistem dan desain antar muka.

A. Analisa Sistem

1. Analisa sistem lama

   Dalam mendeteksi pelanggaran lalu lintas (tilang elektronik) pihak terkait mengimplementasikan teknologi untuk mencatat pelanggaran tersebut yakni dengan menggunakan sistem ETLE (*Electronic Traffic Law Enforcement*), namun sistem tersebut belum mampu mendeteksi pelanggaran masker.

   Pada penelitian (Maliye dkk., 2021) "*Mask and Helmet Detection in Two-Wheelers using YOLOv3 and Canny Edge Detection*" telah mengimplementasikan YOLOv3 untuk mendeteksi pelanggaran Helm dan Masker dengan deteksi citra plat menggunakan deteksi tepi canny, Dengan masukkan berupa video, sistem mendapatkan citra plat terkait namun belum mampu mengidentifikasi karakter nomornya.

   Kemudian pada penelitian (Albert dkk., 2020) "Deteksi Helm pada Pengguna Sepeda Motor dengan Metode *Convolutional Neural Network*" telah mengimplementasikan YOLOv3 untuk mendeteksi sepeda motor kemudian mencari lokasi helm dengan menambahkan persentase tinggi 60% dari tinggi sepeda motor. Dengan masukkan berupa video, sistem berhasil mendapatkan citra helm terkait..

2. Analisa sistem yang diusulkan

Pada subbab ini akan menjelaskan tentang gambaran umum dari sistem yang akan dibangun berdasarkan :

a. Analisa kebutuhan fungsi

Berdasarkan analisa sistem lama di atas, penelitian ini bertujuan untuk mendeteksi pelanggaran helm atau masker pada citra pengendara sepeda motor, kemudian mendapatkan hasil identifikasi karakter plat secara otomatis jika terjadi pelanggaran. Untuk deteksi objeknya, sistem ini menggunakan *transfer learning* YOLOv4. YOLOv4 dipilih karena merupakan peningkatan penting dari YOLOv3. Sementara identifikasi karakter plat memanfaatkan teknologi Tesseract-OCR dengan melakukan pelatihan model untuk meningkatkan akurasinya.

b. Analisa kebutuhan data

1) Data *input*

Pada penelitian ini menggunakan 600 gambar yang terdiri dari motor, helm, masker dan pelat nomor dengan masing-masing gambar sejumlah 150.

a) Sepeda Motor dan Helm

Data sepeda motor dan helm diperoleh dari *google images* dengan memperhatikan kualitas gambar serta keberagaman objek seperti pada Gambar 3.1.

Gambar 3. 1 Data sepeda motor dan helm
(Sumber: *google images*)

b) Masker

Data masker diperoleh dari situs *kaggle (Face Mask Detection* yang berisi 853 citra masker dengan 3 *class* (dengan masker, tanpa masker dan penggunaan masker yang tidak sesuai). Kemudian dipilih 150 citra masker yang memiliki *class* dengan masker (Kaggle, 2020). Data citra masker seperti pada Gambar 3.2.

Gambar 3. 2 Data citra masker
(Sumber: (Kaggle, 2020))

c) Pelat Nomor

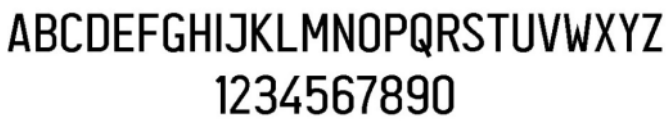
Data Pelat Nomor diperoleh dari *website kaggle (Indonesian Plate Number)*. *Dataset* ini berisi pelat nomor area

Cirebon yang kemudian dipilih 150 (Kaggle, 2018). Data citra plat seperti pada Gambar 3.3.

Gambar 3. 3 Data citra plat nomor
(Sumber: (Kaggle, 2018))

d) *Font* Pelat Nomor

Jenis *font* pelat nomor yang digunakan untuk *training* model *tesseract* diperoleh dari situs *dafont* (dafont, 2020). Data *font* pelat nomor seperti pada Gambar 3.4.



Gambar 3. 4 *Font* pelat nomor

2) Data Uji

Data uji diperoleh melalui kamera *smartphone* dengan total 9 data seperti pada Gambar 3.5. Data tersebut diambil dari 3 pengendara dengan setiap pengendara menggunakan 3 kondisi sebagai berikut :

a) Pengendara motor memakai helm dan masker

b) Pengendara motor hanya memakai helm

c) Pengendara motor hanya memakai masker

(a) (b) (c)

Gambar 3. 5 Data uji

3. Analisa kebutuhan perangkat

   a. Kebutuhan Hardware

      1) PC dengan Core i5, 4GB RAM dan Harddisk 1TB

      2) Windows 10

      3) Nvidia Geforce MX150

   b. Kebutuhan Software

      1) CMake 3.21.3

      2) OpenCV 4.5.3

      3) CUDA dan CUDNN 11.4

      4) Tesseract v5.0.0-alpha.20210811

      5) Anaconda Navigator 2.1.2

      6) Python 3.7.0

      7) Tensorflow-GPU 2.3

      8) LabelImg 1.8.1

      9) Serak-trainer 0.4

B. Desain Sistem (arsitektur)

1. Desain deteksi objek

   a. *Preprocessing* data

      1) Mengumpulkan citra sesuai paparan dalam kebutuhan data.

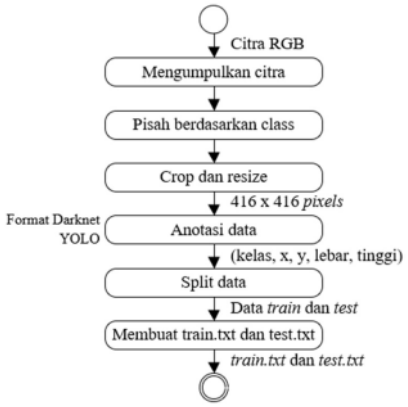
      2) Memisahkan citra tersebut berdasarkan kelas.

      3) Melakukan *crop* dan *resize* agar data menjadi satu ukuran.

      4) Menganotasikan setiap citra.

      5) Membagi data menjadi data *training* dan *testing*

      6) Membuat train.txt dan test.txt yang berisi pemetaan lokasi citra



Gambar 3. 6 Desain *preprocessing* data

   b. Pelatihan YOLOv4

      1) Melakukan *cloning framework darknet* dari *github* pengembang.

      2) Melakukan konfigurasi *file .cfg*

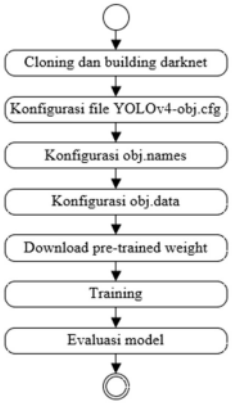
      3) Mengonfigurasi *obj.names* yang berisi nama kelas

      4) Mengonfigurasi *obj.data* yang berisi pemetaan lokasi data.

      5) *Mendownload pre-trained* model (yolov4.conv.137)

6) Proses *training*

7) Evaluasi model



Gambar 3. 7 Desain *training* YOLOv4

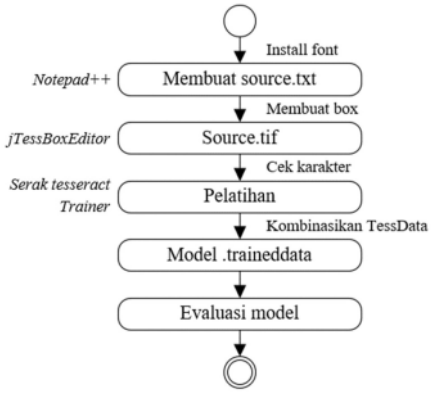
2. Desain pengenalan karakter

   a. Membuat *source.txt*

   b. Membuat *box* setiap karakter

   c. Proses *training*

   d. Evaluasi model



Gambar 3. 8 Desain pengenalan karakter

3. Desain inti

   a. Konfigurasi sistem

      1) Melakukan *cloning* program inti

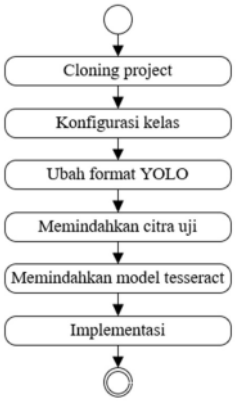
      2) Konfigurasi kelas

      3) Mengubah format model YOLO

      4) Memindahkan model Tesseract

      5) Memindahkan citra uji

      6) Implementasi



Gambar 3. 9 Konfigurasi sistem

   b. Desain testing

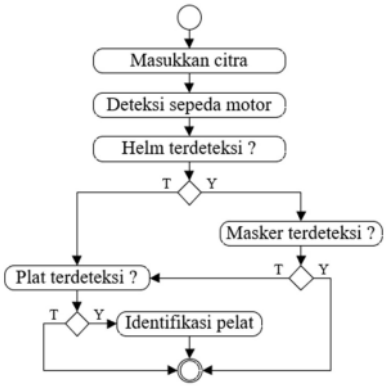
      1) Memasukkan citra uji

      2) Mendeteksi objek sepeda motor.

      3) Mendeteksi objek helm.

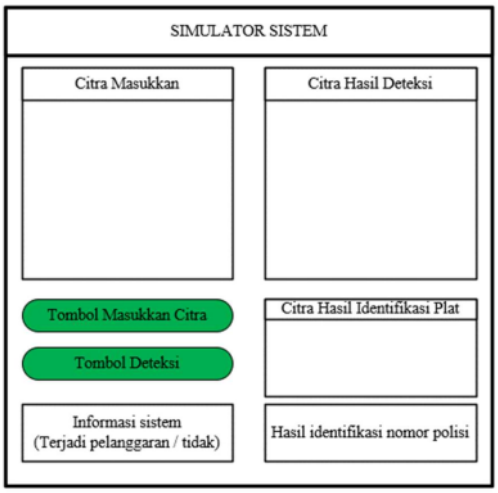
      4) Jika helm terdeteksi maka lanjut mendeteksi masker, jika tidak maka mendeteksi pelat dan melakukan identifikasi kemudian selesai.

5) Jika masker terdeteksi maka selesai, jika tidak maka mendeteksi pelat dan melakukan rekognisi kemudian selesai.



Gambar 3. 10 Desain testing

C. Desain Antar Muka



Gambar 3. 11 Tampilan aplikasi

Gambar 3.11 merupakan tampilan antar muka dari simulator sistem yang dibuat. Tahapan pada tampilan yang digunakan adalah sebagai berikut :

1. Pengguna menekan tombol masukkan citra, kemudian pengguna memilih citra yang ingin dimasukkan sehingga citra tersebut otomatis di tampilkan pada *frame* citra masukkan.
2. Pengguna menekan tombol deteksi, sehingga citra masukkan di deteksi oleh sistem dan hasil deteksi tersebut otomatis di tampilkan pada *frame* citra hasil deteksi. Jika tidak terjadi pelanggaran, sistem menampilkan informasi tidak ada pelanggaran pada informasi sistem. Namun jika terjadi pelanggaran, sistem otomatis menampilkan citra hasil identifikasi plat pada *frame* citra hasil identifikasi plat, menampilkan hasil identifikasi pengenalan nomor polisi pada hasil identifikasi nomor polisi dan menampilkan informasi terjadi pelanggaran pada informasi sistem.

BAB IV

IMPLEMENTASI DAN HASIL

Dalam bab ini akan dijelaskan tentang implementasi lembar kerja, keterkaitan lembar kerja, implementasi program, pengujian sistem, hasil dan evaluasi hasil.

A. Implementasi Lembar Kerja

Dalam penelitian ini menggunakan YOLOv4 sebagai *object detector* kemudian memanfaatkan teknologi Tesseract-OCR untuk pengenalan *plat*. sehingga sistem yang dibangun memerlukan proses sebagai berikut :

1. Lembar kerja deteksi objek

   a. *Preprocessing* data

      Tahapan *preprocessing* terhadap data citra dilakukan dengan melakukan pengumpulan data, memisahkan citra berdasarkan *class*, pengubahan ukuran *pixel* pada citra (*resize*) dan melakukan pemotongan, menganotasikan citra pada keseluruhan citra dan membagi data tersebut.

      1) Pengumpulan data

         a) Sepeda motor dan helm

            Data sepeda motor dan helm yang umum digunakan oleh masyarakat Indonesia diperoleh dari *google images* yang terdiri atas 150 citra sepeda motor dan 150 citra helm. Dengan ketentuan memilih helm *open face* (bagian kaca bening dan batok menutupi telinga).

b) Masker

Data Masker terdiri atas 2 jenis kelas yakni bermasker dan tidak dengan jumlah 850 citra yang kemudian dipilih 150 citra yang menggunakan masker.

c) Plat

Data plat berisi 500 citra plat dengan kode kota E yakni Cirebon, yang kemudian dipilih 150 citra plat.

2) Memisahkan data berdasarkan kelas

Proses memisahkan data berdasarkan kelas berguna untuk mempermudah pengelompokan data. Pada proses ini, peneliti membuat satu folder utama yang berfungsi menaruh data berdasarkan kelas.

3) Crop dan resize

Proses pemotongan dan penskalaan dilakukan untuk membuat data citra menjadi satu ukuran yakni 416 x 416. Proses ini dilakukan karena YOLOv4 menerapkan *multiscale-detection.* Sehingga saat pelatihan, ukuran citra tidak boleh terlalu besar. Ukuran 416 x 416 *pixels* digunakan agar tidak menghilangkan karakteristik sampel setelah diubah ukurannya.

4) Anotasi data

Proses anotasi dilakukan pada seluruh data yang berjumlah 600 menggunakan aplikasi LabelImg yang bertujuan untuk memberi *bounding box* dan *class* yang sesuai agar citra dapat dikenali sistem

seperti pada Gambar 4.1. Setiap citra yang sudah teranotasi akan menghasilkan *file .txt* dengan format YOLO yang berisi (*class*, x, y, w, h) seperti pada Tabel 4.1.

Gambar 4. 1 Contoh anotasi citra

Tabel 4. 1 Contoh format anotasi

Keterangan	**Nilai**
Kelas objek	3
Posisi x	0.499412
Posisi y	0.494048
Lebar	0.434118
Tinggi	0.926190

5) Split data

Setelah semua citra telah berlabel kemudian membagi data menjadi data *train* dan *test*. Folder *train* berfungsi untuk menaruh data yang akan diproses pada proses pembelajaran, sedangkan folder *test* berfungsi untuk memvalidasi data pada proses *training*. Jumlah seluruh data adalah 600 kemudian dibagi menjadi 568 untuk pelatihan dan 32 untuk *test* dengan perincian seperti pada tabel 4.2 berikut :

Tabel 4. 2 Perincian pembagian data

No.	Kelas	Jumlah	Latih	Uji
1	Helm	150	138	12
2	Masker	150	137	13
3	Sepeda Motor	150	138	12
4	Plat	150	137	13
Total		600	568	32

6) Membuat file *train.txt* dan *test.txt* yang berisi pemetaan lokasi citra.

Dalam mempermudah pekerjaan tersebut, generate_train.py digunakan untuk menghasilkan train.txt, sementara untuk menghasilkan test.txt perlu mengubah *string train* menjadi *test*.

generate_train.py

```
1.  import os
2.  image_files = []
3.  os.chdir("train")
4.  for filename in os.listdir(os.getcwd()):
5.      if filename.endswith(".jpg"):
6.          image_files.append("data/obj/train/" +
    filename)
7.  os.chdir("..")
8.  with open("train.txt", "w") as outfile:
9.      for image in image_files:
10.         outfile.write(image)
11.         outfile.write("\n")
12.     outfile.close()
13. os.chdir("..")
```

b. Pelatihan YOLOv4

Setelah tahap terakhir data *preprocessing*, citra terbagi menjadi data *training* dan *testing*. Selanjutnya melakukan proses berikut sehingga proses *training* menghasilkan model.

1) Melakukan *cloning framework darknet*

Framework darknet tersedia pada situs *github* pengembang YOLOv4 (https://github.com/AlexeyAB/darknet). Setelah *repository*

tersebut tersedia pada komputer lokal (folder darknet) dengan menggunakan *git clone* / dengan *download*, kemudian membangunnya menggunakan aplikasi CMake dengan tujuan mengintegrasikan *framework* dengan CUDA dan OpenCV pada komputer lokal sehingga *framework* dapat digunakan.

2) Melakukan konfigurasi file *yolov4-obj.cfg*.

File tersebut digunakan untuk melatih *custom dataset*, yang berisi kumpulan parameter. Parameter *yolov4-obj.cfg* yang diubah untuk melakukan proses *training* adalah *batch* 64, *subdivisions* 16, *width* 416, *height* 416, *classes* 4, *max batches* 8000 (2000**classes*), *steps* 6400 dan 7200 (80% dan 90% dari *max batches*) dan *filters* 27 ((*classes* + 5)x3).

yolov4-obj.cfg

```
1. batch=64
2. subdivisions=16
3. width=416
4. height=416
5. classes=4
6. max_batches=8000
7. steps=6400,7200
8. filters=27
```

3) Mengonfigurasi *obj.names*

Obj.names berisi nama kelas yang digunakan (urutan harus sesuai dengan kondisi saat anotasi data).

obj.names

```
1. Helm
2. Masker
3. Motor
4. Plat
```

4) Mengonfigurasi *obj.data*

Obj.data berisi pemetaan lokasi data. Dengan memasukkan file *train.txt*, *test.txt* dan *obj.names* ke dalam folder data. Folder *backup* berfungsi sebagai tempat penyimpanan model hasil *training*.

obj.data

```
1. classes = 4
2. train = data/train.txt
3. valid = data/test.txt
4. names = data/obj.names
5. backup = backup
```

5) *Mendownload pre-trained weight* (yolov4.conv.137)

yolov4.conv.137 merupakan jaringan yang telah dilatih pada *dataset* MS COCO. File tersebut tersedia pada situs *github* pengembang YOLOv4 (https://github.com/AlexeyAB/darknet).

6) Proses *training*

Sesuai file *yolov4-obj.cfg*, *max batches* (iterasi maksimal) 8000 dengan setiap 1000 iterasi akan menghasilkan 1 model sehingga total akan terdapat 8 model. Berikut ini adalah perintah untuk melakukan *training* :

```
darknet.exe detector train data/obj.data yolov4-obj.cfg
yolov4.conv.137
```

7) Evaluasi model

Setelah proses *training* selesai, maka evaluasi dilakukan untuk mengukur mAP atau F1-Score dari model hasil *training*. Selanjutnya model dengan mAP atau F1-Score tertinggi akan diintegrasikan dengan sistem. Perintah evaluasi model seperti berikut :

```
darknet.exe detector map data/obj.data yolov4-obj.cfg backup/yolov4-
obj_1000.weights
```

2. Lembar kerja pengenalan karakter

   a. Membuat *source.txt*

      Pertama komputer *menginstall font* plat nomor sesuai yang telah dipaparkan dalam kebutuhan data. Kemudian membuat source.txt yang berisi angka (0 - 9)*15 = 150 karakter dan alpabet (A – Z)*15 = 390 karakter.

source.txt

```
1. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2. 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

   b. Membuat *box* setiap karakter

      Setelah source.txt terbuat, selanjutnya membuat *box* per karakter menggunakan *box generator* (jTessBoxEditor) dengan memilih *font* yang telah di *install*. Kemudian akan proses tersebut menghasilkan *file .tif* seperti gambar 4.2.

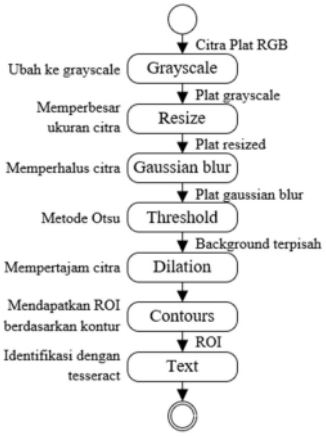
Gambar 4. 2 Contoh *source.tif*

   c. Proses *training*

      Proses *training* menggunakan Serak Trainer dengan memasukkan *file .tif* dan mengonfigurasi jenis *font* yang digunakan. Proses *training* menghasilkan 15 data yang kemudian digabungkan menjadi 1 menjadi *plat.trainneddata*.

d. Evaluasi model

Evaluasi model Tesseract dilakukan untuk mengetahui hasil identifikasi dari model Tesseract yang telah dilatih dengan gambaran seperti pada Gambar 4.3.



Gambar 4. 3 Evaluasi model Tesseract

3. Lembar kerja sistem inti

a. Konfigurasi sistem

1) Melakukan c*loning* program inti

Untuk dapat menggunakan model YOLOv4 menggunakan Tensorflow, maka perlu mengubah format model tersebut (*.weight*) menjadi format yang mendukung Tensorflow (*.pb*). Project yang digunakan untuk proses selanjutnya menggunakan repositori https://github.com/hunglc007/tensorflow-yolov4-tflite.

2) Konfigurasi kelas

Setelah repositori tersebut tersedia pada perangkat lokal maka mengubah konfigurasi kelas yang terdapat dalam core/config.py

dengan mengubahnya menjadi kelas yang digunakan (seperti *obj.names* di atas).

config.py

```
1.  __C.YOLO.CLASSES = "./data/classes/custom.names"
```

3) Mengubah format model YOLO

Untuk mengubah format model, model YOLO terpilih dimasukkan dalam folder data. Sehingga perintahnya akan seperti berikut :

```
python save_model.py --weights ./data/yolov4 -obj_6000.weights --output
./checkpoints/skripsi-6000 --input_size 416 --model yolov4
```

4) Memasukkan citra uji

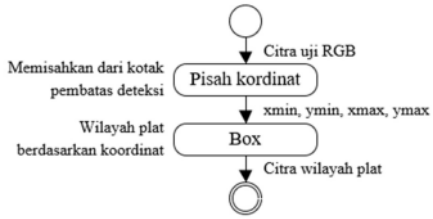
Setelah repositori tersebut tersedia pada perangkat lokal maka selanjutnya memindahkan model terpilih ke dalam folder data/images.

5) Memindahkan model Tesseract

Model pelatihan tesseract harus dipindahkan ke dalam direktori C:/Program Files/Tesseract-OCR/tessdata sehingga dapat dilakukan pemanggilan pada model tersebut.

6) Implementasi

Menambahkan beberapa baris kode ke dalam *project* agar proses testing sistem inti dapat dilakukan yang dibahas dalam implementasi program. Untuk mendapatkan wilayah plat secara otomatis, maka diperlukan proses seperti pada Gambar 4.5 yang selanjutnya di proses seperti pada Gambar 4.3.



Gambar 4. 4 Proses identifikasi karakter plat

b. Testing

Setelah model terpilih telah diubah format menjadi *.pb*, selanjutnya proses testing dapat dilakukan sampai mendapatkan hasil dengan alur sebagai berikut :

1) Memasukkan citra uji

Proses testing dilakukan dengan perintah sebagai berikut :

```
python detect.py --weights ./checkpoints/skripsi-6000 --size 416 --model yolov4 -images ./data/images/test1.jpg --skripsi
```

2) Mendeteksi objek sepeda motor

Setelah menjalankan perintah di atas, sistem akan mendeteksi objek sepeda motor pada citra uji dengan memanfaatkan model yang dipanggil. Proses deteksi seperti yang dipaparkan dalam simulasi algoritma. Setelah objek sepeda motor terdeteksi selanjutnya mendeteksi objek helm.

3) Mendeteksi objek helm

Sistem akan mendeteksi objek helm pada citra uji dengan memanfaatkan model yang dipanggil. Proses deteksi seperti yang

dipaparkan dalam simulasi algoritma. Kemudian sistem akan melakukan pengkondisian.

a. Jika helm terdeteksi maka lanjut mendeteksi masker, jika tidak maka mendeteksi pelat dan melakukan rekognisi kemudian selesai.
b. Jika masker terdeteksi maka selesai, jika tidak maka mendeteksi pelat dan melakukan rekognisi kemudian selesai.

B. Keterkaitan Lembar Kerja

1. Keterkaitan lembar kerja deteksi objek

a. *Preprocessing* data – pelatihan YOLOv4

Preprocessing data dilakukan untuk memproses data citra dengan mengumpulkan data citra sesuai dalam kebutuhan data, kemudian memisahkan data tersebut berdasarkan kelas untuk mempermudah pendataan citra, *crop* dan *resize* dilakukan untuk mendapatkan ukuran citra konsisten 416 x 416 *pixels*, kemudian anotasi dilakukan pada setiap citra dengan format YOLO sehingga setiap citra menghasilkan file.txt yang berisi (kelas, x, y, w, h) yang kemudian dilakukan pembagian data dengan persentase 95% untuk pelatihan dan 5% untuk testing dari 600 data. Terakhir membuat train.txt dan test.txt yang berisi pemetaan lokasi citra *train* dan *test* sehingga citra siap untuk digunakan untuk pelatihan.

Setelah proses *preprocessing* data yang menghasilkan data *train* dan testing, proses selanjutnya adalah melakukan *cloning framework darknet* dari situs *github* pengembang dan membangunnya menggunakan aplikasi CMake agar terintegrasi dengan OpenCV dan CUDA. Kemudian

melakukan konfigurasi file *yolov4-obj.cfg* yang berisi parameter yang akan digunakan dalam pelatihan, kemudian mengonfigurasi *obj.names* dan *obj,data* serta *mendownload pre-trained weight*. Kemudian proses *training* model akan menghasilkan 8 model yang akan dievaluasi berdasarkan mAP dan F1-Score model tersebut sehingga model dengan mAP atau F1-Score tertinggi akan dipilih untuk diintegrasikan dengan sistem inti.

2. Keterkaitan lembar kerja pengenalan karakter

   Untuk melatih model Tesseract diperlukan data latih berupa source.txt yang kemudian memberi *box* untuk setiap karakternya sehingga menghasilkan file *.tif* yang berisi karakter dengan *class* sesuai karakter tersebut sehingga data siap digunakan untuk pelatihan. Pelatihan akan menghasilkan beberapa file yang selanjutnya digabungkan menjadi *plat.traineddata*. Terakhir file tersebut akan diuji menggunakan *script python* untuk mengevaluasi model tersebut.

3. Keterkaitan lembar kerja sistem inti

   a. Konfigurasi sistem – testing

      Konfigurasi sistem dilakukan untuk mengintegrasikan deteksi objek dan pengenalan karakter dengan memanfaatkan model YOLO dan Tesseract yang telah dilatih dengan mengimplementasikannya pada *library* Tensorflow. Sehingga proses testing dapat dilakukan untuk mendapatkan hasil dari sistem yang telah dibuat.

C. Implementasi Program (Development)

Berikut ini adalah tahap implementasi yang telah dilakukan :

1. Membuat detect.py yang berguna mendeteksi objek.

   a. Mendefinisikan *flags*

```
flags.DEFINE_string('framework', 'tf', '(tf, tflite, trt')
flags.DEFINE_string('weights', './checkpoints/yolov4-416', 'path to weights file')
flags.DEFINE_integer('size', 416, 'resize images to')
flags.DEFINE_string('model', 'yolov4', 'yolov3 or yolov4')
flags.DEFINE_list('images', './data/images/motor.jpg', 'path to input image')
flags.DEFINE_string('output', './detections/', 'path to output folder')
flags.DEFINE_float('iou', 0.45, 'iou threshold')
flags.DEFINE_float('score', 0.50, 'score threshold')
flags.DEFINE_boolean('skripsi', False, 'perform skripsi')
```

   b. Menyimpan ukuran masukkan jaringan dan gambar dalam variabel

```
input_size = FLAGS.size
images = FLAGS.images
```

   c. Melakukan pemanggilan terhadap *weight* yang sudah di rubah format

```
saved_model_loaded = tf.saved_model.load(FLAGS.weights, tags=[tag_constants.SERVING])
```

   d. Melakukan *loop* terhadap gambar

```
for count, image_path in enumerate(images, 1):
```

   1) Mendapatkan objek gambar

```
original_image = cv2.imread(image_path)
original_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB)
image_data = cv2.resize(original_image, (input_size, input_size))
image_data = image_data / 255.
image_name = image_path.split('/')[-1]
image_name = image_name.split('.')[0]
images_data = []
for i in range(1):
    images_data.append(image_data)
images_data=np.asarray(images_data).astype(np.float32)
```

   2) Melakukan prediksi dengan memanggil model

```
infer = saved_model_loaded.signatures['serving_default']
batch_data = tf.constant(images_data)
pred_bbox = infer(batch_data)
for key, value in pred_bbox.items():
    boxes = value[:, :, 0:4]
```

```
    pred_conf = value[:, :, 4:]
```

3) Melakukan *Non-Max Supression* pada deteksi

```
boxes, scores, classes, valid_detections = tf.image.combined_non_max_suppression(
        boxes=tf.reshape(boxes, (tf.shape(boxes)[0], -1, 1, 4)),
        scores=tf.reshape(
            pred_conf, (tf.shape(pred_conf)[0], -1, tf.shape(pred_conf)[-1])),
        max_output_size_per_class=50,
        max_total_size=50,
        iou_threshold=FLAGS.iou,
        score_threshold=FLAGS.score
    )
```

4) Melakukan format *bounding box* dari *ymin, xmin, ymax, xmax* menjadi *xmin, ymin, xmax, ymax*.

```
original_h, original_w, _ = original_image.shape
bboxes = utils.format_boxes(boxes.numpy()[0], original_h, original_w)
```

5) Menyimpan deteksi dalam satu variabel

```
pred_bbox = [bboxes, scores.numpy()[0], classes.numpy()[0], valid_detections.numpy()[0]]
```

6) Membaca kelas pada *config*

```
class_names = utils.read_class_names(cfg.YOLO.CLASSES)
allowed_classes = list(class_names.values())
```

7) Melakukan pengecekan jika helm atau masker tidak terdeteksi maka akan melakukan identifikasi plat seperti pada proses 2.

```
counted_classes = count_objects(pred_bbox, by_class = True, allowed_classes=allowed_classes)
if len(counted_classes) == len(allowed_classes):
  print("Objek Motor, Helm, Masker, Plat terdeteksi")
  image = utils.draw_bbox(original_image, pred_bbox, info=True, allowed_classes=allowed_classes, read_plate = False)
  print("Program Selesai")
else:
  # motor
  if allowed_classes[2] in counted_classes.keys():
    print("Motor terdeteksi")
    # helm
    if allowed_classes[0] in counted_classes.keys():
      print("Helm terdeteksi")
      # Masker
      if allowed_classes[1] in counted_classes.keys():
        print("Masker terdeteksi")
        image = utils.draw_bbox(original_image, pred_bbox, info = True, allowed_classes=allowed_classes, read_plate = False)
```

```
        else:
          print("Masker tidak terdeteksi")
          image = utils.draw_bbox(original_image, pred_bbox, info = True, allowed_classes=allowed_classes, read_plate = True)
      else:
        print("Helm tidak terdeteksi")
        image = utils.draw_bbox(original_image, pred_bbox, info = True, allowed_classes=allowed_classes, read_plate = True)
    else:
      print("Motor tidak terdeteksi")
    print("Program Selesai")
```

8) Menyimpan gambar deteksi dan selesai.

```
image = cv2.cvtColor(np.array(image), cv2.COLOR_BGR2RGB)
path = FLAGS.images
cv2.imwrite(FLAGS.output + os.path.basename(path[0]), image)
```

2. *utils.py* yang berguna untuk mengidentifikasi plat

a. Melakukan pengecekan plat.

```
if read_plate and class_name=="Plat":
  height_ratio = int(image_h / 25)
  plate_number = recognize_plate(image, coor)
  if plate_number != None:
    cv2.putText(image, plate_number, (int(coor[0]), int(coor[1]-height_ratio)), cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,0), 2)
```

b. Fungsi identifikasi plat.

```
def recognize_plate(img, coords):
    # memisahkan koordinat dari box
    xmin, ymin, xmax, ymax = coords
    # mendapatkan gambar plat berdasarkan koordinat
    box = img[int(ymin)-5:int(ymax)+5, int(xmin)-5:int(xmax)+5]
    # merubah ke citra grayscale
    gray = cv2.cvtColor(box, cv2.COLOR_RGB2GRAY)
    # melakukan resize
    gray = cv2.resize(gray, None, fx = 3, fy = 3, interpolation = cv2.INTER_CUBIC)
    # memperhalus
    blur = cv2.GaussianBlur(gray, (5,5), 0)
    # thresholding citra menggunakan metode Otsu untuk preprocessing tesseract
    ret, thresh = cv2.threshold(blur, 0, 255, cv2.THRESH_OTSU | cv2.THRESH_BINARY_INV)
    # membuat kernal persegi untuk operasi dilation (morfologi)
    rect_kern = cv2.getStructuringElement(cv2.MORPH_RECT, (5,5))
    # mengaplikasikan dilation
    dilation = cv2.dilate(thresh, rect_kern, iterations = 1)
    # mencari kontur untuk mendapatkan bagian karakter plat
    try:
        contours, hierarchy = cv2.findContours(dilation, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    except:
        ret_img, contours, hierarchy = cv2.findContours(dilation, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    # mengurutkan contours kiri-kanan
    sorted_contours = sorted(contours, key=lambda ctr: cv2.boundingRect(ctr)[0])
    # mengcopy citra grayscale
    im2 = gray.copy()
```

```
    # variable penampung karakter plat
    plate_num = ""
    # melakukan loop pada kontur untuk mendapatkan karakter
    for cnt in sorted_contours:
        x,y,w,h = cv2.boundingRect(cnt)
        height, width = im2.shape
        if height / float(h) > 6: continue
        ratio = h / float(w)
        if ratio < 1: continue
        area = h * w
        if width / float(w) > 15: continue
        if area < 50: continue
        rect = cv2.rectangle(im2, (x,y), (x+w, y+h), (0,255,0),2)
        # ROI
        roi = thresh[y-5:y+h+5, x-5:x+w+5]
        roi = cv2.bitwise_not(roi)
        roi = cv2.medianBlur(roi, 5)
        try:
            text = pytesseract.image_to_string(roi,lang='plat',
config='-c
tessedit_char_whitelist=0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ --psm 8
--oem 1')[0]
            text = text.replace("\n", " ")
            cv2.putText(im2,text,(x+(int(1/2*w)),y-
10),cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0), 2)
            plate_num += text
        except:
            text = None
    if plate_num != None:
        print("Plat Nomor #: ", plate_num)
    cv2.namedWindow('finalImg', cv2.WINDOW_NORMAL)
    cv2.imshow("finalImg",im2)
    cv2.waitKey(0)
    p = os.path.join(os.getcwd(), 'detections')
    cv2.imwrite(os.path.join(p,'bacaplat.jpg'),im2)
    return plate_num
```

D. Pengujian Sistem

Pengujian sistem dilakukan berdasarkan tahapan berikut :

1. Pengujian model deteksi objek

Tabel 4. 3 Pengujian model deteksi objek

Iterasi	mAP	F1-score	Objek	AP	TP	FP
1000	91,91%	0.78	Helm	90.62	20	6
			Maker	88.63	37	15
			Motor	98.32	22	17
			Plat	90.08	21	9
2000	93.38%	0.77	Helm	96.83	20	5
			Masker	87.89	36	21
			Motor	98.33	26	12
			Plat	90.46	21	14
3000	91.95%	0.85	Helm	93.66	19	3
			Masker	90.56	37	11
			Motor	94.92	26	7

Iterasi	mAP	F1-Score	Objek	AP	TP	FP
3000	91.95%	0.85	Plat	88.66	21	7
4000	92.04%	0.83	Helm	93.27	19	2
			Masker	88.34	36	18
			Motor	96.76	26	6
			Plat	89.78	21	6
5000	90.48%	0.85	Helm	88.28	18	2
			Masker	91.44	37	14
			Motor	93.63	26	6
			Plat	88.56	21	4
6000	88.78%	0.86	Helm	80.16	16	1
			Masker	93.96	38	10
			Motor	92.34	26	4
			Plat	88.67	21	7
7000	92.28%	0.85	Helm	87.97	17	3
			Masker	93.13	37	13
			Motor	93.15	26	5
			Plat	94.89	22	6
8000	92.40%	0.85	Helm	88.01	17	3
			Masker	93.20	37	15
			Motor	93.31	26	5
			Plat	95.09	22	3

Tabel 4.3 di atas menunjukkan hasil prediksi yang dilakukan oleh model YOLOv4 dengan melakukan perhitungan *confussion matrix* yang menghasilkan TP, TN, FP dan FN. Kemudian AP (*average precision*) didapatkan dari kurva *precission* dan *recall* yang diringkas menjadi nilai tunggal yang mewakili rata-rata semua presisi. Nilai F1-Score merupakan *harmonic mean* dari *precision* dan *recall* rentang 0. Sampai 1.0 yang berarti jika F1-Score punya skor yang baik mengindikasikan bahwa model tersebut mempunyai *precision* dan *recall* yang baik. Sementara nilai mAP (*mean Average Precision*) menunjukkan tingkat akurasi deteksi objek dan letaknya.

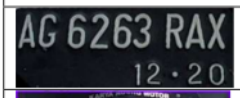
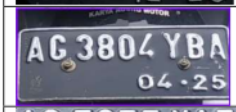
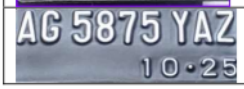
Terlihat bahwasanya model 2 (iterasi 2000) menghasilkan nilai mAP tertinggi dengan mAP sebesar 93.38% sementara nilai F1-Score rendah

yakni 0.77 yang mengindikasikan bahwa model memiliki akurasi deteksi objek yang tinggi namun dalam prediksi objek kelas tersebut kurang baik.

Kemudian untuk F1-Score tertinggi berada pada model 6 (iterasi 6000) dengan nilai F1-Score sebesar 0.86 dan mAP sebesar 88.78% yang mengindikasikan bahwa akurasi deteksi dan prediksi seimbang (baik) sehingga model 6 dipilih untuk di integrasikan dengan sistem inti.

2. Pengujian model pengenalan plat

Tabel 4. 4 Pengujian model pengenalan plat

Nomor plat	Tanpa model		Dengan model		Selisih
	Prediksi	TP	Prediksi	TP	
	A?6?53RAX	6	AG6263RAX	8	2
	A533???BN	2	A638O?YBA	6	4
	A?48?5?AV	4	AG5?75?AX	6	2

Pada tabel 4.4 di atas disajikan perbandingan prediksi TP (*true positif*) yang dilakukan Tesseract tanpa model dan dengan menggunakan model yang telah dilatih, mengindikasikan bahwa pelatihan model Tesseract membantu mengidentifikasi karakter plat sehingga di putuskan bahwa sistem inti menggunakan model Tesseract yang telah dilatih.

3. Pengujian sistem inti

Tabel 4. 5 Pengujian sistem inti

Citra uji	Kondisi	Terdeteksi	*Trust score*	Identifikasi
Pengendara 1	1. Lengkap	Motor	97 %	Tidak
		Helm	99 %	
		Masker	93 %	
		Plat	86 %	

Citra uji	Kondisi	Terdeteksi	*Trust score*	Identifikasi
Pengendara 1	2. Hanya pakai helm	Motor	80 %	Ya
		Helm	99 %	
		Plat	99 %	
	3. Hanya pakai masker	Motor	90 %	Ya
		Masker	99 %	
		Plat	98 %	
Pengendara 2	1. Lengkap	Motor	99 %	Seharusnya tidak
		Helm	68 %	
		Masker tidak terdeteksi		
		Plat	97 %	
	2. Hanya pakai helm	Motor	99 %	Ya
		Helm	54 %	
		Plat	99 %	
	3. Hanya pakai masker	Motor	84 %	Ya
		Masker	99 %	
		Plat	97 %	

Tabel 4. 6 Hasil identifikasi plat

Citra Uji	Kondisi	Plat Asli	Terbaca
Pengendara 1	2. Hanya pakai helm	AG3804YBA	CEO
	3. Hanya bermasker		AG38O4IBE
Pengendara 2	1. Lengkap	AG6225REC	AG6226REC
	2. Hanya pakai helm		AG6Z26RE
	3. Hanya bermasker		AG6ZRC

Pada Tabel 4.5 di atas menunjukkan hasil uji coba yang dilakukan terhadap sistem, sehingga jika terjadi pelanggaran akan mengidentifikasi karakter plat dengan hasil pengenalan plat disajikan pada tabel 4.6 di atas. Terlihat pada pengendara 2 dengan kondisi 1 (lengkap) tidak seharusnya sistem melakukan pengenalan plat, namun sistem tidak dapat mendeteksi objek masker sehingga dilakukan identifikasi karakter plat.

Kemudian berdasarkan Tabel 4.6 di atas, sistem secara otomatis mengidentifikasi karakter plat berdasarkan wilayah prediksi plat yang dilakukan YOLOv4 sehingga mendapatkan karakter plat.

E. Hasil

Metode YOLOv4 diterapkan dalam penelitian yang digunakan untuk mendeteksi plat nomor bagi pelanggar helm dan masker, kemudian memanfaatkan teknologi Tesseract-OCR untuk pengenalan karakter plat. penelitian ini dapat menampilkan pendeteksian objek dan mendapatkan karakter plat secara otomatis. Dalam proses ini pengguna perlu memasukkan citra pengendara sepeda motor, di mana citra tersebut dikelompokkan menjadi 3 kategori kondisi yakni kondisi lengkap, hanya memakai helm dan hanya memakai masker yang kemudian sistem akan melakukan deteksi objek pada citra tersebut, jika terjadi pelanggaran maka dilakukan proses pengenalan plat sehingga sistem otomatis mendapatkan karakter plat sehingga sistem ini memudahkan pengguna. Oleh karena itu sistem yang dibangun telah sesuai dengan rancangan yang dibuat.

F. Evaluasi Hasil

Evaluasi sistem yang dilakukan merupakan hasil dari uji coba sistem yang telah dibuat. Hasil Analisa ini didapat dari kesimpulan semua uji coba yang dijalankan oleh sistem. Kemudian evaluasi hasil coba ini digunakan untuk mengevaluasi tahapan yang dilakukan.

1. Kelebihan sistem
    a. Sistem mampu mendeteksi objek sepeda motor, helm, masker dan plat dalam satu citra secara bersamaan.
    b. Kemudian jika terjadi pelanggaran, sistem mampu mendapatkan karakter pada plat secara otomatis.

c. Kinerja YOLOv4 sebagai *single stage detector* (detektor satu tahap) menghasilkan mAP tertinggi 93.38% (pada model 2) dan F1-Score tertinggi 0.86 (pada model 6) kemudian model pelatihan Tesseract sedikit membantu akurasi pengenalan karakter plat.

2. Kekurangan sistem

a. Sistem belum di implementasikan pada *input* berupa video.

b. Akurasi pengenalan karakter plat perlu ditingkatkan.

BAB V

PENUTUP

Dalam bab ini akan dijelaskan tentang kesimpulan dan saran.

A. Kesimpulan

Berdasarkan hasil dan pembahasan yang telah diuraikan pada bab sebelumnya, maka dapat diambil kesimpulan yaitu :

1. Deteksi objek masker, helm, pelat nomor dan sepeda motor pada pengendara sepeda motor dapat dilakukan menggunakan algoritma YOLOv4 yang menghasilkan prediksi *bounding box* (kotak pembatas) dengan kelas objek dan *trust score* (skor kepercayaan).
2. Identifikasi karakter plat nomor secara otomatis menggunakan Tesseract-OCR dapat dilakukan dengan memisahkan citra plat berdasarkan koordinat *bounding box* hasil prediksi plat YOLOv4 (untuk mendapatkan lokasi plat), kemudian melakukan *preprocessing* dengan mengubah menjadi *grayscale*, melakukan *resize* 3x, memperhalus citra plat (*gaussian blur*), menerapkan metode *otsu thresholding* untuk memisahkan objek karakter dengan *background* plat, mempertajam citra dengan operasi morfologi (dilasi), terakhir mendapatkan ROI (*Region of Interest*) karakter berdasarkan kontur sehingga ROI tersebut digunakan untuk pengenalan karakter.
3. Kinerja metode YOLOv4 untuk mendeteksi masker, helm, pelat nomor dan sepeda motor pada pengendara sepeda motor cukup baik dengan mAP tertinggi 93.38% (model 2) dan F1-Score tertinggi 0.86 (model 6).

B. Saran

Sistem cerdas deteksi pelat nomor untuk pelanggaran helm dan masker masih jauh dari kata sempurna. Oleh karena itu, adapun saran yang dapat penulis berikan untuk penelitian selanjutnya yaitu sebagai berikut:

1. Penambahan variasi dataset sehingga sistem mampu mengidentifikasi objek secara akurat.
2. Penambahan metode pada pengenalan karakter plat seperti *Perspective Transformation* dan *skewness* (kecondongan) agar plat terbaca dengan maksimal.
3. Penerapan secara *real-time* dengan penambahan metode untuk mengidentifikasi pengendara sepeda motor dengan objek motor yang digunakan

SISTEM CERDAS DETEKSI PELAT NOMOR UNTUK PELANGGARAN HELM DAN MASKER

ORIGINALITY REPORT

24%	23%	5%	14%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University Politehnica of Bucharest Student Paper	4%
2	github.com Internet Source	2%
3	docplayer.info Internet Source	1%
4	repository.its.ac.id Internet Source	1%
5	123dok.com Internet Source	1%
6	medium.com Internet Source	1%
7	kimiafarmaapotek.co.id Internet Source	1%
8	www.reddit.com Internet Source	1%
	research-report.umm.ac.id	

9	Internet Source	1 %
10	haiqalmuhamadalfarisi.medium.com Internet Source	1 %
11	Submitted to University of Nottingham Student Paper	<1 %
12	bloblokan.blogspot.com Internet Source	<1 %
13	lp2m.unpkediri.ac.id Internet Source	<1 %
14	ichi.pro Internet Source	<1 %
15	www.coursehero.com Internet Source	<1 %
16	eprints.ums.ac.id Internet Source	<1 %
17	algorit.ma Internet Source	<1 %
18	covid19.hukumonline.com Internet Source	<1 %
19	ykkim.gitbook.io Internet Source	<1 %
20	pythonlang.dev Internet Source	<1 %

21	Submitted to Institut Pemerintahan Dalam Negeri Student Paper	<1 %
22	publication.petra.ac.id Internet Source	<1 %
23	repository.teknokrat.ac.id Internet Source	<1 %
24	Submitted to Universidade Nova De Lisboa Student Paper	<1 %
25	link.springer.com Internet Source	<1 %
26	Submitted to Politeknik Negeri Jember Student Paper	<1 %
27	Submitted to Universitas Atma Jaya Yogyakarta Student Paper	<1 %
28	blog.unnes.ac.id Internet Source	<1 %
29	repository.unimar-amni.ac.id Internet Source	<1 %
30	text-id.123dok.com Internet Source	<1 %
31	Sumaira Manzoor, Eun-Jin Kim, Gun-Gyo In, Tae-Yong Kuc. "Performance Evaluation of YOLOv3 and YOLOv4 Detectors on Elevator	<1 %

	Button Dataset for Mobile Robot", 2021 21st International Conference on Control, Automation and Systems (ICCAS), 2021 Publication	
32	eprints.uny.ac.id Internet Source	<1 %
33	www.mdpi.com Internet Source	<1 %
34	stackoverflow.com Internet Source	<1 %
35	Submitted to Universitas Pamulang Student Paper	<1 %
36	lika-likuartikelku.blogspot.com Internet Source	<1 %
37	Submitted to Montana State University, Bozeman Student Paper	<1 %
38	Submitted to Multimedia University Student Paper	<1 %
39	repo.itera.ac.id Internet Source	<1 %
40	repository.ub.ac.id Internet Source	<1 %
41	repository.usd.ac.id Internet Source	<1 %

42	repository.unpas.ac.id Internet Source	<1 %
43	documents.mx Internet Source	<1 %
44	etheses.uin-malang.ac.id Internet Source	<1 %
45	edukasicovid19.com Internet Source	<1 %
46	mti.binus.ac.id Internet Source	<1 %
47	pt.scribd.com Internet Source	<1 %
48	ugoproto.github.io Internet Source	<1 %
49	eprints.mercubuana-yogya.ac.id Internet Source	<1 %
50	massuqo.wordpress.com Internet Source	<1 %
51	repository.radenintan.ac.id Internet Source	<1 %
52	jurnal.polgan.ac.id Internet Source	<1 %
53	repository.uhn.ac.id Internet Source	<1 %

54	eprints.uad.ac.id Internet Source	<1 %
55	id.scribd.com Internet Source	<1 %
56	johannessimatupang.wordpress.com Internet Source	<1 %
57	media.neliti.com Internet Source	<1 %
58	publications.lib.chalmers.se Internet Source	<1 %
59	rickyrudianto.blogspot.com Internet Source	<1 %
60	www.postingkampus.info Internet Source	<1 %
61	www.radarcilacap.com Internet Source	<1 %
62	digilib.uns.ac.id Internet Source	<1 %
63	docobook.com Internet Source	<1 %
64	repository.isi-ska.ac.id Internet Source	<1 %
65	www.docstoc.com Internet Source	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		