5. EVALUASI DAN PENGUJIAN SISTEM

Bab ini akan membahas hasil dari performa program pengenalan rambu lalu lintas Indonesia yang telah dibuat. Selain itu pada bab ini akan membahas pengujian dataset, pengujian *preprocessing* dan pengujian terhadap penelitian sebelumnya. Pengujian dataset berisi pengujian akurasi, kecepatan dan jarak dari model yang di-train menggunakan dataset yang dibuat untuk mengukur performa *baseline* yang dicapai dalam skripsi ini. Pengujian *preprocessing* berisi pengujian akurasi dan kecepatan pada dataset yang telah di *preprocess* untuk mencari tahu pengaruh *preprocessing* terhadap performa model YOLOv4-tiny. Terakhir adalah perbandingan hasil yang dicapai dalam percobaan yang dilakukan dalam skripsi ini terhadap hasil implementasi metode yang digunakan dalam penelitian sebelumnya.

5.1 Hasil Program Pengenalan Rambu Lalu Lintas

Program pengenalan rambu lalu lintas Indonesia dibuat dalam bahasa pemrograman python dengan *frame*work Tensorflow. File weight dari model YOLOv4-tiny yang akan digunakan dalam program ini adalah *weight* yang di-*train* pada dataset berwarna dengan ukuran *input* model 416 x 416. Program dibuat dalam 2 bagian sesuai desain *user interface* pada bab 3 dimana bagian atas akan menampilkan gambar *frame* sedangkan di bagian bawah menampilkan rambu yang berhasil dideteksi. Program juga dibuat untuk bekerja menggunakan *input* rekaman langsung dari *webcam* dan video. Untuk menguji program, *input* yang digunakan adalah video yang diambil dari YouTube.

Cara kerja program ini, pertama program akan membaca dan mengambil 1 *frame* dari *input* rekaman / video. Selanjutnya program dengan menggunakan model YOLOv4-tiny akan melakukan deteksi pada gambar *frame*. Setelah dideteksi, apabila ditemukan rambu dalam gambar maka program akan menggambar *bounding box* dan label pada *frame*. *Output* deteksi berupa kumpulan *class index* rambu akan digunakan program untuk menampilkan atau menyembunyikan rambu pada bagian bawah tampilan program. Tampilan program ketika berhasil mendeteksi rambu dapat dilihat pada gambar 5.1

Gambar 5.1 Program Pengenalan Rambu Lalu Lintas

Sumber : Dokumentasi Pribadi

5.1.1 Pengujian Akurasi Deteksi pada Program

Bagian ini akan menjelaskan proses pengujian akurasi deteksi program pengenalan rambu lalu lintas Indonesia. Pengujian ini akan mencerminkan performa program dalam mendeteksi baik dengan input rekaman langsung dari kamera ataupun dengan input video. Dalam pengujian ini akan menggunakan 3 buah input video dan 2 buah input rekaman langsung kamera. Pengujian dilakukan menggunakan video dan input kamera langsung karena terdapat perbedaan signifikan ketika program dijalankan menggunakan video input dengan program yang dijalankan menggunakan kamera dengan simulasi keadaan riil. Metrik yang digunakan untuk menghitung akruasi deteksi pada program adalah metrik *recall*. Metrik ini menghitung jumlah deteksi benar terhadap total jumlah deteksi salah dan tidak tedeteksi. Dalam kasus ini terdeteksi salah dan tidak terdeteksi dianggap sebagai False Negative (TN), sedangkan terdeteksi benar dianggao True Positive (TP). Rumus *recall* dapat dilihat pada persamaan 2.5.

Video yang digunakan sebagai input program ada 3 yakni 1 video yang dikumpulkan secara mandiri di Surabaya saat malam hari dan 2 Video yang merupakan cuplikan dari video yang diambil dari kanal YouTube “Niko_Channel” dengan judul “Tunjungan ROMANSA - Wajah

Baru Jalan Tunjungan SURABAYA" dan "Sidoarjo ke SURABAYA naik Motor.. Jalannya Banyak LUBANG ???" (Niko_ Channel, 2021, 2021). Selain itu pengujian ini juga menggunakan 2 simulasi dengan input kamera yang dijalankan dari dalam mobil. Simulasi ini digunakan untuk menggambarkan situasi ketika program digunakan secara riil.

Untuk pengujian menggunakan video dilakukan menggunakan komputer dengan CPU Intel I3-8100 dan RAM 8GB. Sedangkan untuk pengujian simulasi rekaman langsung kamera menggunakan laptop dengan CPU Intel I5-11400H dan RAM 8GB dengan status *on battery.* Karena perbedaan antara komputer dengan laptop untuk menjalankan program, performa program di komputer lebih tinggi hinga mencapai 19 – 20 FPS sedangkan di laptop hanya mampu mencapai kecepatan deteksi 6 FPS. Hal ini menyebabkan proses deteksi di laptop menjadi patah-patah karena adanya lompatan dalam frame yang diambil karena kecepatan deteksi yang rendah. Hasil perhitungan akurasi deteksi program dapat dilihat di tabel 5.1.

Tabel 5.1

Hasil Pengujian Correct Detection Rate Program

Rambu	Kemunculan	Terdeteksi Benar	Mis-klasifikasi	Tidak Terdeteksi	Recall
Input Video 1 – 20 FPS – Video Siang Hari					
Dilarang Parkir	25	19	1	5	76.00%
Dilarang Berhenti	29	15	7	7	51.72%
Batas Kecapatan Maksimum 40km	13	6	2	5	46.15%
Dilarang Masuk	1	1	-	-	100.00%
Dilarang Mendahului	2	2	-	-	100.00%
Total	**70**	**43**	**10**	**17**	**61.43%**
Input Video 2 – 20 FPS – Video Siang Hari					
Dilarang Parkir	15	13	1	1	86.67%
Dilarang Berhenti	14	10	2	2	71.43%
Batas Kecapatan Maksimum 40km	4	3	-	1	75.00%
Dilarang Masuk	2	2	-	-	100.00%
Total	**35**	**28**	**3**	**4**	**80.00%**
Input Video 3 – 20 FPS – Video Malam Hari					
Dilarang Parkir	3	3	-	-	100.00%
Dilarang Berhenti	11	8	-	3	72.73%

Rambu	Kemunculan	Terdeteksi Benar	Mis-klasifikasi	Tidak Terdeteksi	Recall
Dilarang Putar Balik	2	2	-	-	100.00%
Dilarang Putar Balik dan Belok Kanan	3	1	2	-	33.33%
Dilarang Belok Kanan	1	1	-	-	100.00%
Dilarang Belok Kiri	2	2	-	-	100.00%
Batas Kecapatan Maksimum 40km	2	2	-	-	100.00%
Belok Kiri ikuti Isyarat Lampu	2	1	-	1	50.00%
Belok Kiri Langsung	1	1	-	-	100.00%
Dilarang Masuk	6	4	-	2	66.67%
Total	**33**	**25**	**2**	**6**	**75.76%**
Input Rekaman Langsung 1 – 6 FPS – Simulasi dari Dalam Mobil					
Dilarang Parkir	12	10	-	2	83.33%
Dilarang Berhenti	21	12	-	9	57.14%
Dilarang Putar Balik	2	2	-	-	100.00%
Dilarang Putar Balik dan Belok Kanan	3	3	-	-	100.00%
Dilarang Belok Kanan	2	2	-	-	100.00%
Batas Kecapatan Maksimum 40km	8	8	-	-	100.00%
Belok Kiri ikuti Isyarat Lampu	3	1	-	2	33.33%
Belok Kiri Langsung	2	1	-	1	50.00%
Dilarang Masuk	4	1	-	3	25.00%
Total	**57**	**40**	**-**	**17**	**70.18%**
Input Rekaman Langsung 2 – 6 FPS – Simulasi dari Dalam Mobil					
Dilarang Parkir	40	27	3	14	67.50%
Dilarang Berhenti	32	25	-	7	78.13%
Dilarang Putar Balik	2	2	-	-	100.00%
Dilarang Putar Balik dan Belok Kanan	2	2	-	-	100.00%
Dilarang Belok Kanan	1	1	-	-	100.00%
Batas Kecapatan Maksimum 40km	17	16	-	1	94.12%
Belok Kiri ikuti Isyarat Lampu	3	2	-	1	66.67%

Rambu	Kemunculan	Terdeteksi Benar	Mis-klasifikasi	Tidak Terdeteksi	Recall
Belok Kiri Langsung	1	1	-	-	100.00%
Dilarang Masuk	7	5	-	2	71.43%
Total	**105**	**81**	**3**	**25**	**77.14%**

Berdasarkan tabel 5.1, program pengenalan rambu lalu lintas Indonesia dengan metode YOLOv4-tiny berhasil mencapai skor *recall* rata-rata 72.9% dari 5 pengujian. Dari tabel diatas juga dapat dilihat jika masih ditemukan kasus misklasifikasi rambu dan kasus rambu tidak terdeteksi sama sekali, terutama pada rambu dilarang parkir dan dilarang berhenti. Selain 2 rambu tersebut, model mampu mengenali rambu dengan cukup baik. Kategori rambu dengan correct detection rate terendah adalah rambu dilarang masuk pada pengujian program dengan input rekaman langsung 1 dengan skor 25%. Berdasarkan observasi, diduga pada pengujian program dengan input rekaman langsung 1, frame yang dideteksi oleh program mengalami blur karena kecepatan deteksi (pengambilan gambar) yang rendah sehingga model sulit melakukan deteksi.

5.1.2 Pengujian Perbandingan Kecepatan Deteksi Program terhadap Manusia

Pengujian ini akan mengukur selisih waktu antara saat manusia mendeteksi rambu pada video terhadap waktu program mendeteksi rambu dalam video dengan benar, untuk mencari tahu performa program jika dibandingkan dengan mata manusia. Cara kerjanya pengujian ini adalah dengan mencatat waktu ketika rambu berhasil dilihat oleh manusia pada video, dan mencata waktu program berhasil mendeteksi dengan benar. Kemudian, waktu program akan dikurangi oleh waktu manusia dan menghasilkan selisih waktu dalam satuan detik. Jika program berhasil mendeteksi rambu sebelum manusia, maka selisih waktu akan negatif, sebaliknya jika manusia mendeteksi rambu sebelum program maka selisih waktu akan positif.

Waktu yang akan dicatat adalah waktu dari 10 rambu yang terdeteksi dengan benar dari masing-masing video input. Video input yang akan digunakan adalah lima video yang digunakan pada pengujian tingkat deteksi program. 10 rambu yang terdeteksi benar. Perlu dicata bahwa dalam pengujian ini, manusia secara sadar dan aktif mencari rambu dalam video sehingga tidak menggambarkan situasi nyata dimana saat berkendara manusia bisa saja tidak memerhatikan rambu lalu lintas. Untuk rata-rata selisih waktu dalam video dapat melihat tabel 5.2.

Tabel 5.2

Hasil Uji Selisih Waktu Deteksi Manusia dan Program

Input	Rata-rata Selisih Waktu Deteksi
Video 1	1.1654 Detik
Video 2	1.8982 Detik
Video 3	1.5871 Detik
Rekaman Langsung 1	0.7992* (1.5984 Detik)
Rekaman Langsung 2	0.5274* (1.0548 Detik)
Rata-rata	1.461
*Untuk rata-rata selisih waktu deteksi rekaman langsung nilai yang didapat perlu dikalikan 2 karena video memiliki nilai FPS yang lebih rendah	

Berdasarkan Tabel 5.2 dapat dilihat jika secara rata-rata dibutuhkan waktu 1.461 detik bagi program untuk mendeteksi rambu setelah rambu dikenali oleh mata manusia sebelumnya. Selain itu ditemukan pada beberapa kasus pada pengujian dimana program berhasil mendeteksi rambu yang belum sepenuhnya tampak bagi manusia, dan dihitung sebagai rambu yang berhasil dideteksi terlebih dahulu oleh program dengan selisih waktu negatif. Namun secara umum, mata manusia mampu mendeteksi rambu lalu lintas terlebih dahulu jika manusia tersebut secara aktif mencari rambu lalu lintas.

5.2 Pengujian Dataset Berwarna

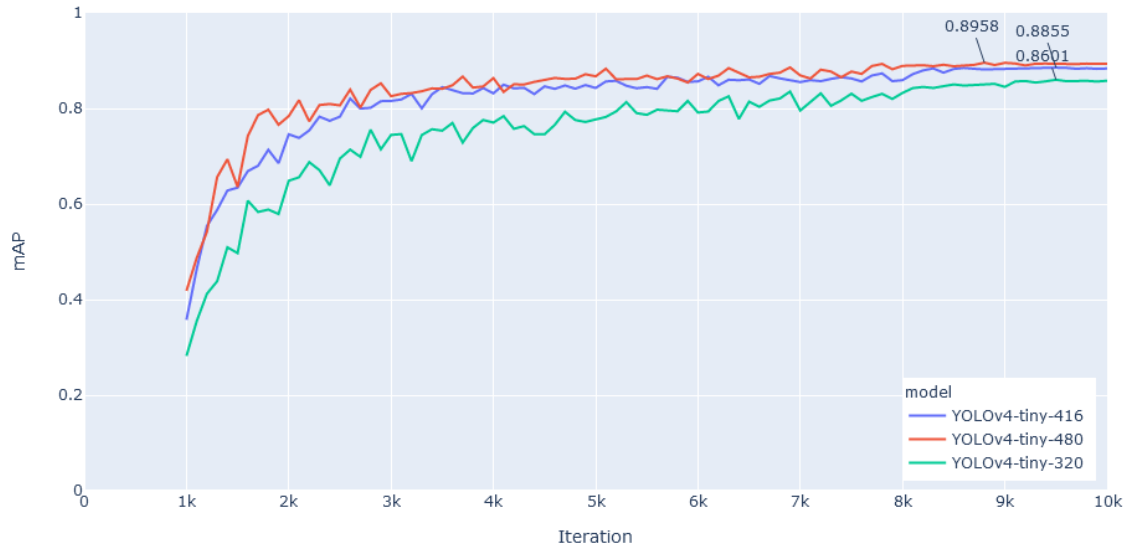
Bagian ini akan menjelaskan proses pengujian yang dilakukan menggunakan dataset berwarna tanpa *preprocessing* yang dikumpulkan dalam skripsi ini. Hal ini bertujuan untuk menghasilkan skor performa yang dapat dibandingkan dengan penelitian-penelitian lain. Untuk pengujian yang akan dilakukan ada 3 jenis yaitu pengujian akurasi, pengujian kecepatan deteksi dan pengujian jarak deteksi.

Pengujian akurasi akan diukur menggunakan metrik mAP@50 yang diujikan pada dataset test setelah proses *training* model YOLOv4-tiny selesai dijalankan. Pengujian kecepatan deteksi akan diukur menggunakan metrik *frames per second* (FPS) yang menunjukkan jumlah gambar yang dapat diproses (melakukan deteksi dan pengenalan pada gambar) oleh model YOLOv4-tiny dalam 1 detik. Terakhir adalah pengujian jarak deteksi, pengujian ini akan diukur dengan cara menghitung persentase rambu yang berhasil dikenali pada jarak tertentu.

Model YOLOv4-tiny yang akan diuji ada 3 varian. Perbedaan antara ketiga varian ini terletak pada ukuran input yang digunakan. Ukuran input ini akan menentukan bagaimana model YOLOv4-tiny akan melakukan *downsizing* pada gambar yang akan diproses. Varian input yang digunakan adalah 416 (416 x 416 pixel), 320, dan 480. Pengujian dilakukan menggunakan 3 varian input ini untuk mengetahui pengaruh input terhadap akurasi dan kecepatan deteksi dari model YOLOv4-tiny.

5.2.1 Pengujian Akurasi

Pengujian akurasi dilakukan untuk melihat kemampuan deteksi dari model YOLOv4-tiny dalam mendeteksi dan mengenali rambu. Akurasi deteksi diukur dengan nilai metrik mAP@50 yang dihasilkan oleh *weight* terbaik dari masing-masing variasi model yang digunakan dalam pengujian dataset berwarna. *Weight* terbaik didapat selama proses *training* dimana dilakukan perhitungan mAP@50 setiap iterasi kelipatan 100 setelah iterasi ke 1000. Untuk melakukan pengujian, *weight* terbaik dari setiap variasi *input* model akan dimuat dan diuji terhadap dataset *test*. Pengujian akurasi dilakukan dalam *notebook training* yang dijalankan pada Google Colab, pengujian menggunakan *script* "detector" milik *framework* Darknet.



Gambar 5.2 Chart mAP Training Dataset Berwarna

Gambar 5.2 menunjukkan pertumbuhan mAP@50 dari iterasi 1000 hingga 1000. Jika melihat pertumbuhan dari ketiga model, secara umum YOLOv4-tiny mampu mencapai mAP@50 tertinggi kurang lebih di iterasi ke 9000.

Tabel 5.3

Hasil Pengujian Akurasi dari Model YOLOv4-tiny

Model	mAP@50
YOLOv4-tiny 416 (416 x 416)	0.8855 / 88.55
YOLOv4-tiny 480	0.8958 / 89.58
YOLOv4-tiny 320	0.8601 / 86.01

Tabel 5.3 menunjukkan hasil mAP@50 yang dicapai oleh setiap variasi YOLOv4-tiny yang di-*train* menggunakan dataset berwarna. Model dengan mAP@50 terbaik adalah YOLOv4-tiny dengan ukuran input 480 yang berhasil mencapai skor 89.58 %. Berdasarkan pengujian ini dapat disimpulkan bahwa ukuran input dapat mempengaruhi hasil mAP@50 yang dicapai oleh model. Semakin besar ukuran input yang digunakan maka semakin tinggi skor mAP@50 yang diperoleh, begitu juga sebaliknya.

5.2.2 Pengujian Kecepatan

Pengujian kecepatan dilakukan untuk mencari tahu kapabilitas model YOLOv4-tiny untuk melakukan deteksi dan pengenalan rambu lalu lintas secara *real-time*. Pengujian ini dilakukan dengan menghitung rata-rata *frames per second* (FPS) yang dihasilkan setelah program selesai melakukan pengenalan pada video. Pengujian kecepatan dilakukan dengan menggunakan *script* "detect_video.py" yang dapat ditemukan dalam *repository* tensorflow-yolov4-tflite (The AI Guy, 2020). Pengujian kecepatan ini dilakukan pada komputer yang sama dengan spesifikasi CPU i3 8100 dan RAM 8GB agar hasil dapat dikomparasikan satu sama lain. Pengujian hanya menggunakan CPU agar lebih menggambarkan performa yang diharapkan dari implementasi model YOLOv4-tiny pada *small computer* atau *embedded device*.

Video yang digunakan sebagai *input* merupakan video yang digunakan saat pengujian akurasi deteksi program yakni video dengan judul "Tunjungan ROMANSA - Wajah Baru Jalan Tunjungan SURABAYA" dari kanal YouTube Niko_ Channel (Niko_ Channel, 2021). Video ini dipilih karena video menggambarkan situasi yang direkam dari kendaraan dan juga jalan yang direkam belum muncul pada dataset. Pengujian ini dilakukan dengan 3 variasi model dimana setiap model akan dijalankan sebanyak 3x untuk mengetahui konsistensi FPS yang dihasilkan selama pengujian kecepatan.

Tabel 5.4

Hasil Pengujian Kecepatan dari Model YOLOv4-tiny

Model	Rata - rata FPS Pengujian ke -			Rata - rata FPS dari 3 Pengujian
	1	2	3	
YOLOv4-tiny 416	19.07	19.55	19.6	19.41
YOLOv4-tiny 480	15.71	15.89	15.82	15.81
YOLOv4-tiny 320	29.88	29.91	30.26	30.02

Dari hasil yang diperoleh pada tabel 5.4, dapat dilihat jika model YOLOv4-tiny bisa melakukan deteksi dalam program / *script* dengan nilai FPS antara 15 hingga 30 FPS. Dari pengujian ini juga bisa dilihat jika ukuran *input* mempengaruhi kecepatan deteksi dari model YOLOv4-tiny. Semakin besar ukuran *input* yang digunakan maka kecepatan deteksi akan semakin pelan, begitu juga sebaliknya.

Tabel 5.5

Hasil Akurasi dan Kecepatan Deteksi YOLOv4-tiny dengan Dataset Berwarna

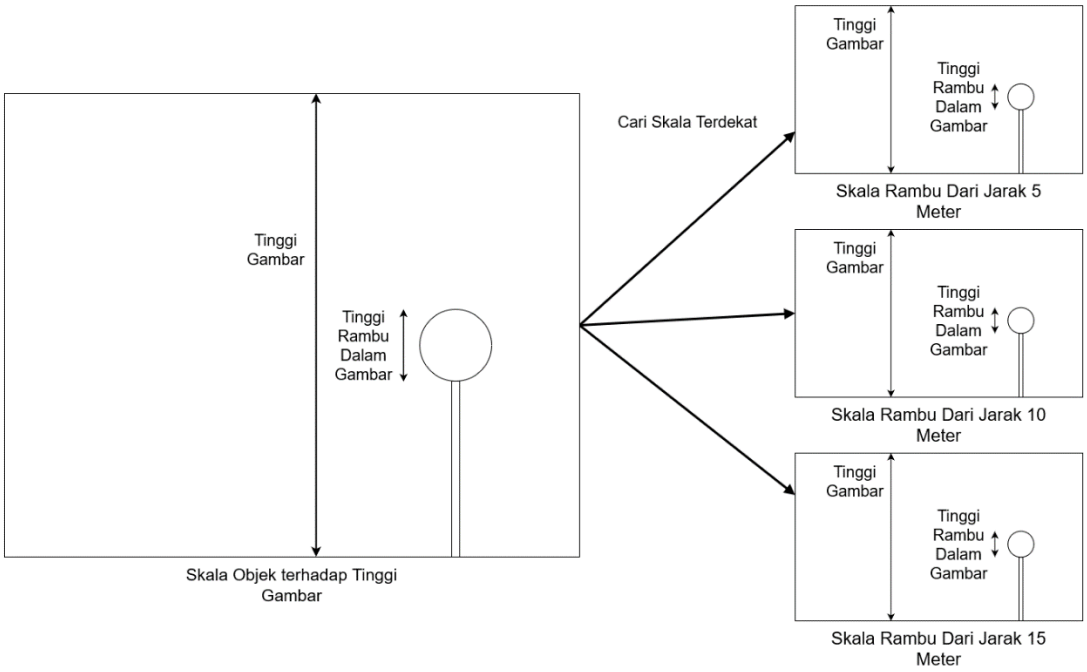
Model	mAP@50	FPS
YOLOv4-tiny 416 (416 x 416)	88.55	19.41
YOLOv4-tiny 480	**89.58**	15.81
YOLOv4-tiny 320	86.01	**30.02**

Berdasarkan hasil tabel 5.5, dapat dilihat korelasi antara mAP@50 dan FPS yang dicapai oleh ketiga variasi model yang diuji pada bagian ini. Karena model dengan mAP@50 terbaik dan model dengan kecepatan deteksi terbaik berbeda, model terbaik tidak dapat ditentukan dengan mudah. Namun karena perfomra mAP@50 dan FPS yang dicapai model YOLOv4-tiny dengan input 416 cukup seimbang dibandingkan terhadap 2 model lainnya, model ini dipilih sebagai model baseline yang digunakan pada program pengenalan rambu lalu lintas Indonesia.

5.2.3 Pengujian Jarak

Pengujian jarak dilakukan untuk mengetahui pada jarak berapa rambu dapat dikenali oleh model YOLOv4-tiny dengan dimensi *input* 416. Pengujian dilakukan dengan menghitung

skala tinggi rambu dalam gambar terhadap tinggi gambar. Agar rambu dapat dikategorisasikan perlu menghitung skala tolok ukur dari rambu lingkaran dan persegi panjang di jarak 5, 10 dan 15 meter. Hal ini dilakukan dengan berasumsi jika semua rambu berbentuk lingkaran memiliki diameter yang sama, dan berasumsi jika semua rambu berbentuk persegi panjang memiliki dimensi yang sama.



Gambar 5.3 Ilustrasi Pengukuran Jarak Rambu

Sumber : Dokumentasi Pribadi

Gambar 5.3 menggambarkan proses pengujian jarak dari rambu yang berhasil dideteksi oleh model YOLOv4-tiny. Skala tinggi rambu dalam gambar yang berhasil dideteksi dikomparasi terhadap skala tolok ukur untuk rambu dengan bentuk yang sama pada jarak 5,10 dan 15 meter. Skala tolok ukur ini adalah rata-rata skala rambu lingkaran atau persegi panjang pada jarak 5, 10 atau 15 meter.

Tabel 5.6

Skala Tolok Ukur Pengujian Jarak

Bentuk Rambu & Jarak Pengambilan	**Rata-rata Skala Tinggi Rambu dalam Gambar**
Lingkaran 5 Meter	0.112255 / 11.22 %
Lingkaran 10 Meter	0.058368 / 5.83 %

Lingkaran 15 Meter	0.039257 / 3.92 %
Persegi Panjang 5 Meter	0.057886 / 5.79 %
Persegi Panjang 10 Meter	0.032211 / 3.22 %
Persegi Panjang 15 Meter	0.023711 / 2.37 %

Dengan menggunakan rasio yang ditemukan pada tabel 5.6 dapat dilakukan kategorisasi jarak rambu dengan melakukan pencarian selisih terdekat antara skala tolok ukur untuk bentuk yang terdeteksi terhadap skala tinggi rambu yang dideteksi. Apabila ditemukan bahwa jarak terdekat adalah dengan rasio tolok ukur untuk 5 meter maka rambu tersebut diklasifikasi sebagai rambu yang terdeteksi dari jarak 5 meter. Pengujian ini dilakukan pada semua gambar dalam dataset test. Tabel 5.7 akan menunjukkan jumlah kemunculan rambu pada jarak 5,10 dan 15 meter dalam dataset *test*.

Tabel 5.7

Distribusi Kemunculan Rambu Dataset Test pada Jarak 5,10 dan 15 Meter

Nama Rambu	**Jumlah Kemunculan Dalam Dataset Test**	**Jarak**		
		5 Meter	**10 Meter**	**15 Meter**
Dilarang Parkir	51	7	18	26
Dilarang Berhenti	56	11	26	19
Dilarang Putar Balik	45	11	25	9
Dilarang Putar Balik dan Belok Kanan	42	6	24	12
Dilarang Belok Kanan	45	10	21	14
Dilarang Belok Kiri	37	5	19	13
Batas Kecapatan Maksimum 40km	48	8	30	10
Belok Kiri ikuti Isyarat Lampu	43	19	23	1
Belok Kiri Langsung	37	17	15	5
Dilarang Masuk	30	2	7	21

Dilarang Mendahului	42	10	9	23

Distribusi jarak untuk semua jenis rambu tidak seimbang karena pada saat pembuatan dataset test tidak mempertimbangkan jarak rambu. Untuk mengukur performa pengujian jarak akan dilakukan penghitungan akurasi jarak deteksi dengan membagi jumlah terdeteksi dengan jumlah kemunculan pada jarak 5/10/15 meter.

Tabel 5.8

Pengujian Deteksi Dataset Test pada Jarak 5,10 dan 15 Meter

Nama Rambu	**Rambu Terdeteksi pada Jarak Jumlah Terdeteksi / Total Kemunculan**			**Akurasi**
	5 Meter	**10 Meter**	**15 Meter**	
Dilarang Parkir	7 / 7	14 / 18	13 / 26	66.67 %
Dilarang Berhenti	8 / 11	21 / 26	9 / 19	67.86 %
Dilarang Putar Balik	9 / 11	25 / 25	9 / 9	95.56 %
Dilarang Putar Balik dan Belok Kanan	4 / 6	20 / 24	9 / 12	78.57 %
Dilarang Belok Kanan	5 / 10	15 / 21	11 / 14	68.89 %
Dilarang Belok Kiri	5 / 5	18 / 19	9 / 13	86.49 %
Batas Kecapatan Maksimum 40km	8 / 8	28 / 30	8 / 10	91.67 %
Belok Kiri ikuti Isyarat Lampu	18 / 19	23 / 23	1 / 1	97.67 %
Belok Kiri Langsung	17 / 17	15 / 15	3 / 5	94.59 %
Dilarang Masuk	2 / 2	2 / 7	12 / 21	53.33 %
Dilarang Mendahului	10 / 10	9 / 9	16 / 23	83.33 %
Total Akurasi	87.74%	87.56%	65.36%	**80.42%**

Berdasarkan hasil pada tabel 5.8 jika rambu dapat dideteksi oleh model dengan cukup baik. Hampir semua jenis rambu memiliki akurasi deteksi per rambu lebih dari 75%, dengan

akurasi keseluruhan mencapai 80.42 %. Namun ditemukan pada beberapa kombinasi rambu dan jarak tertentu, rambu sulit dideteksi. Salah satu rambu yang sulit dideteksi adalah rambu dilarang parkir pada jarak 15 meter dimana model hanya mampu mendeteksi 5 dari 15 kemunculan. Selain itu pada rambu dilarang belok kanan pada semua jarak yang diuji, performanya kurang baik. Mengingat dataset yang digunakan tidak seimbang, hasil pengujian ini tidak menggambarkan performa deteksi aktual model YOLOv4-tiny dengan input 416 x416.

Hasil yang ditemukan pada pengujian jarak ini tidak dapat dibandingkan dengan akurasi deteksi model pada subbab 5.2.1 karena *threshold Intersection Over Union* (IoU) yang digunakan pada pengujian ini lebih tinggi yakni 0.8 dengan minimal *confidence* 0.7.

Gambar 5.4 Cuplikan Video Hasil Deteksi pada Video Malam Hari
Sumber : Dokumentasi Pribadi

Pada gambar 5.4, gambar a menunjukkan rambu dengan pencahayaan yang cukup untuk dilakukan deteksi. Rambu pada gambar a berhasil dideteksi sebagai rambu batas kecepatan maksimum pada *frame* berikutnya. Gambar b menunjukkan rambu dengan pencahayaan yang kurang dan tertutupi sehingga model kesulitan melakukan deteksi pada rambu ini. Gambar c menunjukkan misklasifikasi yang dialami saat deteksi video pada malam hari, dimana rambu dilarang belok kanan dimisklasifikasi sebagai rambu diralang belok kiri.

5.3 Pengujian Dataset Preprocessing

Bagian ini akan membahas hasil pengujian akurasi dan kecepatan menggunakan model YOLOv4-tiny yang menggunakan dataset *preprocessing*. Dataset *preprocessing* mengakomodasi 2 macam dataset yakni dataset *grayscale* dan dataset canny. Dataset *grayscale* dan canny dibuat menggunakan dataset berwarna dimana gambarnya diproses menggunakan script python agar berubah menjadi gambar *grayscale* atau canny *edge*.

Gambar 5.5 Contoh Gambar Dataset Grayscale
Sumber : Dokumentasi Pribadi

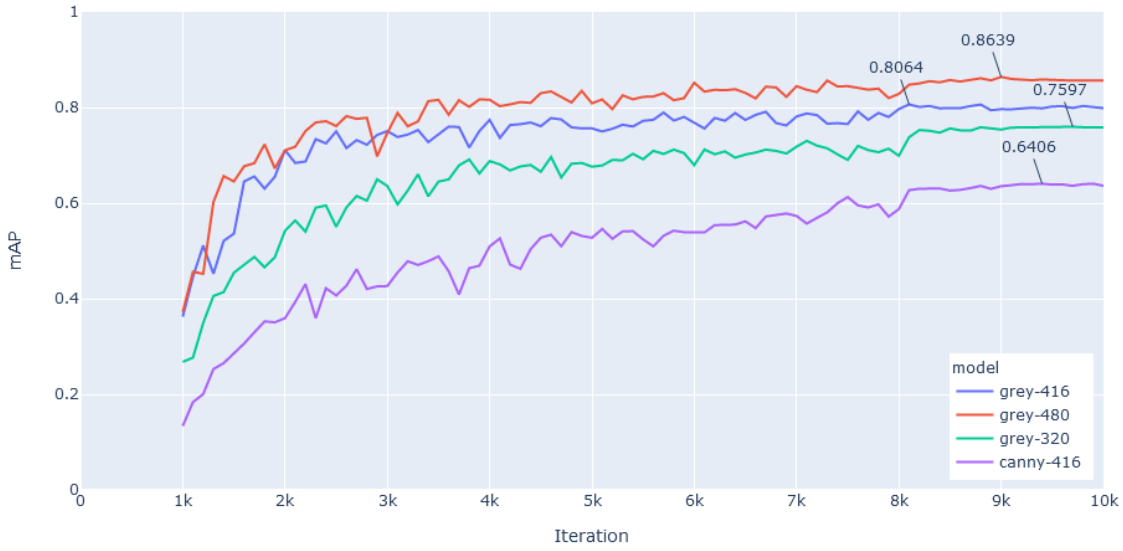
Pengujian pada dataset *preprocessing* ini dilakukan untuk mencari tahu pengaruh gambar *input* teradap performa akurasi dan kecepatan deteksi model YOLOv4-tiny. Dalam pengujian dataset *preprocessing*, dataset *grayscale* akan di-*train* menggunakan 3 variasi ukuran input model yakni 416, 480 dan 320 sedangkan dataset canny hanya akan di-*train* pada ukuran input model 416. Gambar 5.5 menunjukkan tampilan gambar dataset setelah diubah menjadi *grayscale*, dapat dilihat jika gambar masih memiliki detail dalam gambar yang cukup jelas.

Gambar 5.6 Contoh Gambar Dataset Canny

Sumber : Dokumentasi Pribadi

Gambar 5.6 menunjukkan gambar dataset canny setelah diubah untuk menunjukkan fitur canny *edge*. Gambar 5.6 merupakan gambar yang sama dengan gambar 5.5 namun pada gambar 5.6, ditemukan banyak *noise* disekitaran objek dan background sehingga sulit menemukan detail dalam gambar.

5.3.1 Pengujian Akurasi Dataset Preprocessing



Gambar 5.7 Chart mAP Training Dataset Grayscale & Dataset Canny

Pengujian ini dilakukan untuk mencari tahu pengaruh gambar *input* yang digunakan terhadap akurasi deteksi yang dicapai oleh model YOLOv4-tiny. Hasil pengujian dataset *preprocessing* nantinya akan dibandingkan dengan hasil dataset berwarna. Pertumbuhan nilai mAP@50 untuk *training* model YOLOv4-tiny dengan dataset *grayscale* dan dataset canny dapat dilihat pada gambar 5.7.

Berdasarkan hasil yang dicapai dalam pengujian akurasi dataset *preprocessing* dan pengujian akurasi dataset berwarna, dibuat tabel 5.7 yang membandingkan akurasi terbaik yang dicapai oleh setiap kombinasi ukuran *input* model dan dataset.

Tabel 5.9

Perbandingan Akurasi Semua Dataset dan Model

Dataset	Model	mAP@50
Dataset *Original*	YOLOv4-tiny 416	88.55
	YOLOv4-tiny 480	**89.58**
	YOLOv4-tiny 320	86.01
Dataset Grayscale	YOLOv4-tiny 416	80.64
	YOLOv4-tiny 480	86.39
	YOLOv4-tiny 320	75.97
Dataset Canny	YOLOv4-tiny 416	64.06

Dari tabel 5.9 dapat dilihat jika hasil mAP@50 yang dicapai oleh *dataset* berwarna lebih tinggi dibandingkan mAP@50 yang dicapai oleh dataset *preprocessing*. Namun selisih mAP@50 dari dataset berwarna dan dataset *grayscale* tidak terlalu besar yakni antara 0.4 – 2.1%, sehingga cukup sulit untuk menentukan model terbaik tanpa mempertimbangkan performa kecepatan deteksi.

5.3.2 Pengujian Kecepatan Dataset Preprocessing

Pengujian ini dilakukan untuk mencari tahu pengaruh gambar *input* terhadap kecepatan deteksi dari model YOLOv4-tiny. Proses yang dilakukan kurang lebih sama seperti yang dilakukan pada saat pengujian kecepatan untuk dataset berwarna, namun ada sedikit modifikasi yang dilakukan untuk mengakomodasi *input* dataset *grayscale* atau canny. Modifikasi dilakukan dalam *script* pengujian kecepatan untuk mengubah gambar dari 1 *channel* menjadi 3 *channel* dengan menyalin *value* dari *channel* pertama. Perbandingan hasil pengujian kecepatan

yang didapat dari dataset *preprocessing* terhadap dataset berwarna dapat dilihat pada tabel 5.10.

Tabel 5.10

Perbandingan Kecepatan Deteksi Semua Dataset dan Model

Dataset	**Model**	**Rata-rata FPS dari 3x Pengujian**
Dataset Berwarna (Original)	YOLOv4-tiny 416	19.41
	YOLOv4-tiny 480	15.81
	YOLOv4-tiny 320	30.02
Dataset Grayscale	YOLOv4-tiny 416	20.58
	YOLOv4-tiny 480	16.12
	YOLOv4-tiny 320	**30.87**
Dataset Canny	YOLOv4-tiny 416	18.16

Dari tabel 5.10 dapat dilihat jika model yang di-*train* menggunakan dataset *grayscale* memiliki kemampuan deteksi paling cepat dibandingkan dengan model yang di-*train* menggunakan dataset berwarna dan dataset canny. Kecepatan deteksi rata-rata tertinggi berhasil diraih oleh model YOLOv4-tiny dengan dataset grayscale dengan ukuran input 320 x 320 dengan kecepatan deteksi 31.66 FPS Disini juga muncul pola yang sama ketika pengujian kecepatan dataset berwarna yakni ukuran *input* sangat mempengaruhi kecepatan deteksi dari model.

Melihat korelasi antara nilai mAP@50 dengan FPS dari masing-masing model dapat dilihat jika dataset berwarna unggul dari sisi akurasi sedangkan dataset *grayscale* unggul pada sisi kecepatan deteksi. Korelasi ini juga menunjukkan adanya *trade-off* antara kecepata deteksi dengan akurasi, dimana jika akurasi lebih tinggi maka kecepatan deteksi akan lebih pelan. Jika dalam implementasi model YOLOv4-tiny memprioritaskan akurasi, maka dapat menggunakan model yang di-*train* pada dataset berwarna dengan ukuran *input* yang agak besar. Sedangkan jika kecepatan deteksi menjadi prioritas dalam implementasi, dapat menggunakan model yang di-*train* pada dataset *grayscale* dengan ukuran *input* yang kecil.

5.4 Perbandingan Penelitian Sebelumnya

Skripsi ini juga membandingkan akurasi dan kecepatan deteksi dari metode YOLOv4-tiny terhadap metode yang digunakan oleh penelitian sebelumnya yang menggunakan metode

Mask R-CNN (Tabernik & Skocaj, 2019) dan Faster R-CNN (Pon et al., 2018). Namun karena adanya perbedaan dataset dan spesifikasi komputer yang digunakan dalam penelitian sebelumnya, nilai yang ditemukan tidak dapat dibandingkan dengan setara. Oleh karena itu dibuat implementasi metode yang digunakan oleh penelitian sebelumnya pada dataset dan komputer yang sama. metode yang akan diimplementasi adalah Mask R-CNN dan Faster R-CNN.

Metode Mask R-CNN akan diimplementasi menggunakan file *training* dan implementasi dalam *repository* Matterport (Matterport, 2017). Metode Faster R-CNN akan diimplementasi menggunakan *frame*work Detectron2 (Meta Research, 2020). Mengingat Implementasi yang dilakukan tidak sepenuhnya mencerminkan metode yang digunakan dalam penelitian sebelumnya. Hasil yang didapat dari implementasi hanya memberikan gambaran akurasi dan kecepatan metode yang digunakan dalam penelitian sebelumnya.

Implementasi Mask R-CNN akan dibuat menggunakan *backbone* restnet50 dan konfigurasi yang diesuaikan dari penelitian sebelumnya. Implementasi Mask R-CNN ini akan menggunakan pre*train*ed *weight* resnet50 yang di-*train* pada data ImageNet dan ukuran *input* 448 x 448 (salah satu kelipatan 64 terdekat dari 416). Implementasi yang dilakukan akan melakukan *training* dengan melakukan freeze (membekukan node pada layer CNN) pada semua layer kecuali layer Head sebanyak 80 epoch, yang kemudian dilanjutkan dengan *training* pada all layer sebanyak 20 epoch.

Implementasi Faster R-CNN dibuat menggunakan *backbone* yang sama seperti yang digunakan dalam penelitian sebelumnya yakni resnet50. Untuk *training* digunakan pre*train*ed weight resnet50 ImageNet seperti yang metode Mask R-CNN. Implementasi ini akan menggunakan *input* 739 x 416. Ukuran *input* yang diperkecil akan tetap mempertahankan aspek rasio *original* dari gambar. Berikut adalah perbadingan hasil akurasi yang dicapai oleh metode Mask R-CNN, Faster R-CNN dan YOLOv4-tiny dengan dataset berwarna.

Tabel 5.11

Perbandingan Akurasi Terhadap Penelitian Sebelumnya

Model	**Ukuran Input Model**	**mAP@50**
YOLOv4-tiny	416 x 416	88.55
	480 x 480	**89.57**
	320 x 320	86.01
Mask R-CNN (Backbone Resnet50) 80 Freeze except Head + 20 All Layer	448 x 448	70.0

Faster R-CNN (Backbone Resnet50)	739 x 416	82.32

Dari tabel 5.11 dapat dilihat jika implementasi metode YOLOv4-tiny dapat bersaing dengan metode Faster R-CNN yang digunakan dalam penelitian sebelumnya. Untuk metode YOLOv4-tiny dengan ukuran *input* 480 x 480 dapat mencapai akurasi terbaik debgan mAP@50 89.57%, melebihi hasil implementasi dari metode yang digunakan dalam penelitian sebelumnya yakni Mask R-CNN dan Faster R-CNN. Namun harus diingat jika perbandingan yang dilakukan tidak sempurna karena adanya perbedaan dataset, dan implementasi yang tidak sempurna karena kurangnya informasi dari penelitian sebelumnya. Implementasi ini hanya menggambarkan perbedaan dari hasil yang didapat dari metode-metode yang digunakan tanpa optimisasi.

Tabel 5.12

Perbandingan Kecepatan Deteksi Terhadap Penelitian Sebelumnya

Model	**Ukuran Input Model**	**FPS**
YOLOv4-tiny	416 x 416	19.41
	480 x 480	15.81
	320 x 320	**30.02**
Mask R-CNN (Backbone Resnet50) 80 Freeze except Head + 20 All Layer	448 x 448	0.73
Faster R-CNN (Backbone Resnet50)	739 x 416	0.85

Dari perbandingan kecepatan deteksi dalam tabel 5.12, dapat dilihat jika untuk implementasi metode YOLOv4-tiny memiliki keunggulan yang sangat signifikan. Semua hasil kecepatan deteksi dilakukan pada komputer yang sama dengan hanya menggunakan CPU sehingga hasil kecepatan deteksi antar model dapat dibandingkan satu sama lain. Hasil yang dicapai metode Mask R-CNN dan Faster R-CNN memiliki kecepatan deteksi yang sangat rendah karena penggunaan *backbone* yang memiliki ukuran jauh lebih besar dibandingkan *backbone* metode YOLOv4-tiny. Hasil ini menggambarkan perbandingan implementasi metode tanpa melakukan optimisasi.

5.5 Modifikasi YOLOv4-tiny

Bagian ini akan menjelaskan mengenai hasil mAP@50 yang dicapai dari model YOLOv4-tiny dengan modifikasi pada konfigurasi yang digunakan untuk *training* dan implementasi model. Model yang akan dimodifikasi berbasis konfigurasi YOLOv4-tiny dengan *input* 416. Modifikasi hanya akan dilakukan pada *layer convolutional* pertama dengan modifikasi pada *size*, *stride* dan *pad* yang digunakan. Modifikasi pada pad dipengaruhi oleh jumlah size yang digunakan.

Dalam pengujian ini akan menggunakan dataset yang dibatasi pada 5 rambu saja yakni rambu dilarang parkir, dilarang berhenti, dilarang putar balik, dilarang belok kanan dan belok kiri ikuti isyarat lampu. Penggunaan dataset yang dibatasi ini bertujuan untuk mempercepat proses *training* untuk melihat pengaruh modifikasi pada mAP@50 yang dicapai. Untuk total keseluruhan gambar yang digunakan adalah 129 gambar. Kombinasi modifikasi dan hasil mAP@50 yang dicapai dalam pengujian ini dapat dilihat pada tabel 5.13.

Tabel 5.13

Detail Percobaan Modifikasi Konfigurasi Layer Convolution Pertama YOLOv4-tiny

Percobaan	Modifikasi	Filters	Size	Stride	Pad	mAP@50
0 (Original)	-	32	3	2	1	89.88
1	Filters	**16**	3	2	1	82.47
2		**64**	3	2	1	81.84
3	Size & Pad	32	**5**	2	**2**	83.86
4		32	**7**	2	**3**	82.28
5	Stride	32	3	**1**	1	93.6
6		32	3	**3**	1	85.34

Berdasarkan Hasil dari tabel 5.13 ditemukan bahwa sebagian besar modifikasi pada parameter convolution layer pertama YOLOv4-tiny tidak meningkatkan tingkat akurasi dari model YOLOv4-tiny, namun mengubah ukuran *stride* dari 2 menjadi 1 meningkatkan akurasi deteksi sebesar 3.7%. Kombinasi *value* parameter terbaik untuk *layer convolution* pertama YOLOv4-tiny adalah *Filters* = 32, *Size* = 3, *Stride* = 1, dan *Pad* = 1. Untuk membuktikan peningkatan akurasi ini, akan dilakukan *training* ulang menggunakan kombinasi value parameter terbaik pada dataset original yang memiliki 11 macam rambu. Hasil *training* dataset original dengan parameter terbaik dapat dilihat di tabel 5.14.

Tabel 5.14

Komparasi Hasil Tuning Layer Convolution Pertama YOLOv4-tiny

Model & Parameter	**mAP@50**
YOLOv4-tiny 416 (*Original*) *Filters* = 32, *Size* = 3, *Stride* = 2, *Size* = 1	88.55
YOLOv4-tiny 416 (Konfigurasi Setelah Tuning) *Filters* = 32, *Size* = 3, *Stride* = 1, *Size* = 1	90.61

Tabel 5.14 menunjukkan bahwa melakukan *tunning* pada *layer convolution* pertama pada arsitektur YOLOv4-tiny dapat meningkatkan akurasi deteksi dari 88.55 menjadi 90.61. Dengan mengubah *stride* dari 2 menjadi 1, *layer convolution* pertama tidak akan melakukan downsizing atau penurunan dimensi gambar sehingga data yang diproses oleh model akan lebih banyak.

Berdasarkan rangkaian pengujian yang telah dilakukan, metode YOLOv4-tiny mampu melakukan deteksi dan pengenalan pada rambu lalu lintas Indonesia. Melihat hasil pengujian pada dataset, model YOLOv4-tiny *baseline* yang menggunakan ukuran input 416 x 416 mampu melakukan pengenalan rambu dengan dengan akurat dan cepat dengan skor mAP@50 88.55% dan 19.4 FPS. Selain itu dengan sedikit modifikasi pada ukuran input dan dataset yang digunakan dapat meningkatkan performa metode YOLOv4-tiny hingga berhasil mencapai akurasi 89.57% dan kecepatan deteksi 30.87 FPS. Setelah dilakukan *tuning* parameter, metode YOLOv4-tiny juga berhasil meningkatkan mAP@50 hingga mencapai akurasi 90.61%.

Selain itu berdasarkan hasil pengujian implementasi metode yang digunakan oleh penelitian sebelumnya oleh Tabernik & Skocaj serta Pon, et al., yang menggunakan Mask R-CNN dan Faster R-CNN. Metode YOLOv4-tiny mampu unggul baik dari segi akurasi deteksi dan kecepatan deteksi. Meskipun implementasi metode penelitian sebelumnya tidak dilakukan optimisasi, namun model *baseline* YOLOv4-tiny tetap mampu unggul baik secara akurasi dan kecepatan deteksi. Yang paling menonjol dalam pengujian terhadap metode yang digunakan dalam penelitian sebelumnya adalah dari segi kecepatan deteksi, dimana kedua metode sebelumnya yang berbasis R-CNN yang bekerja secara 2 tahap (2 *stage detection*) memiliki kecepatan deteksi yang sangat lambat. Mengingat penelitian sebelumnya tidak mengejar kecepatan deteksi yang tinggi karena tujuan penelitian yang berbeda, jika dilakukan optimisasi pada implementasi metode penelitian sebelumnya mungkin akurasi deteksi dari model YOLOv4-tiny, Faster R-CNN dan Mask R-CNN akan sebanding dengan selisih 1 – 2%. Namun kecepatan

deteksi dari metode Mask R-CNN dan Faster R-CNN tidak akan dapat mengalahkan kecepatan deteksi dair YOLOv4-tiny.

Program pengenalan rambu lalu lintas Indonesia yang dibuat dengan menggunakan model *baseline* YOLOv4-tiny dengan ukuran input 416 x 416 untuk melalukan pengenalan secara *real-time* mampu melakukan deteksi pada sebagian besar rambu dengan akurasi yang diukur dengan *metric recall* mencapai rata-rata skor 72.9% dari 5 pengujian. Dalam uji coba program pada video, masih ditemukan situasi dimana rambu yang dideteksi mengalami misklasifikasi yang kemudian dikoreksi menjadi pengenalan rambu yang benar seiring rambu menjadi lebih dekat. Selain itu juga masih ditemukan rambu yang tidak terdeteksi oleh program sama sekali. Dari sisi kecepatan deteksi program, program mampu mencapai kecepatan deteksi di kurang lebih 19 FPS jika dijalankan pada komputer. Namun ketika program dijalankan pada simulasi situasi riil didalam mobil dengan device yang menggunakan baterai seperti Laptop kecepatan deteksi yang didapat oleh program menurun hingga kurang lebih 6 FPS. Mengingat implementasi program hanya menggunakan CPU karena ditujukan untuk menggambarkan situasi kapabilitas komputasi pada *mobile device*, *embedded device* dan *small computer*. Secara garis besar, program dapat digunakan untuk mendeteksi rambu dari dalam kendaraan roda 4 namun belum sempurna karena kecepatan deteksi yang rendah ketika dijalankan dalam situasi riil.