Pengenalan Rambu Lalu Lintas di Indonesia Secara Real-time Menggunakan YOLOv4-tiny

Gregorius Nicholas Goenawan, Alvin Nathaniel Tjondrowiguno, Liliana
Program Studi Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) - 8417658
E-mail: nic.goena@gmail.com, alvin.nathaniel@petra.ac.id, lilian@petra.ac.id

ABSTRAK

Konsentrasi sangat dibutuhkan saat mengemudi. Pengemudi yang kehilangan konsentrasi memiliki kecepatan reaksi yang lebih lambat dan berpotensi lebih tinggi untuk melanggar rambu lalu lintas. Pelanggaran rambu merupakan tindak pidana dengan sanksi yang memberatkan. Selain itu pelanggaran rambu mengganggu kenyamanan dan membahayakan pengendara lain. Oleh karena itu, diperlukan sebuah sistem yang mampu mendeteksi rambu dengan akurat dan cepat sehingga mampu menghimbau pengemudi. Penelitian [18] menggunakan metode Mask R-CNN yang berbasis *convolutional neural network* dan berhasil mencapai mAP@50 melebihi 95%. Tetapi peneliti menggunakan rambu lalu lintas Swedia dan Slovenia yang berbeda dengan rambu lalu lintas Indonesia. Selain itu, peneliti tidak melakukan evaluasi kecepatan deteksi.

Dalam skripsi ini akan menggunakan metode berbasis YOLOv4-tiny untuk mendeteksi rambu lalu lintas Indonesia. Rambu yang akan dideteksi adalah 9 rambu larangan dan 2 rambu perintah. Dataset yang digunakan merupakan dataset yang dikumpulkan secara mandiri.

Metode YOLOv4-tiny dengan input berukuran 416 x 416 berhasil mencapai mAP@50 88.55% dengan kecepatan deteksi 19.41 FPS. Dengan modifkasi pada ukuran input dan dataset, YOLOv4-tiny dapat mencapai skor mAP@50 hingga 89.57% dan kecepatan deteksi hingga 30.02 FPS. Metode YOLOv4-tiny mampu mendeteksi rambu dari jarak 5 sampai 15 meter dengan akurasi 80.42%. Program pengenalan rambu lalu lintas menggunakan YOLOv4-tiny mampu mencapai rata-rata *recall* 72.9%

Kata Kunci: deteksi, rambu lalu lintas Indonesia, *convolutional neural network*, yolov4-tiny, kecepatan deteksi

ABSTRACT

Concentration are crucial when driving. Drivers who lose their concentration tend to have a slower reaction time, and a higher possibility of violating traffic signs. Traffic signs violation is considered a criminal act with harsh penalties. In addition, traffic sign violations interferes with comfort and endanger other road users. Therefore, we need a system that is able to detect signs accurately and quickly which can inform driver in advance. A research on traffic signs detection on Swedish and Slovenian traffic signs use Mask R-CNN model which based on convolutional neural networks [18]. These method was capable of achieving a mAP@50 score that exceeds 95%. However, the research did not evaluate on the detection speed of such methods.

In this research, YOLOv4-tiny is used to detect Indonesian traffic signs. Dataset used in this research are independently collected, which consist of nine prohibition signs and two command signs. The YOLOv4-tiny method with input size of 416 x 416 is able to achieve mAP@50 score of 88.55% with detection speed of 19.41 FPS. With modification to input size and dataset, YOLOv4-tiny are able to achieve mAP@50 score up to 89.58% and detection speed up to 30.87 FPS. YOLOv4-tiny are also able to detect road signs from distance of around 5 to 15 meters with 80.42 % accuracy. Indonesian traffic sign recognition program made by utilizing the YOLOv4-tiny model achieve average recall of 72.9%.

***Keywords**: detection, Indonesian traffic sign, convolutional neural network, yolov4-tiny, detection speed*

1. PENDAHULUAN

Seorang pengemudi diharuskan untuk berkonsentrasi pada jalan, rambu lalu lintas dan kendaraan lain saat sedang berkendara, namun terkadang akan muncul situasi dimana pengemudi dapat kehilangan konsentrasinya saat mengemudi. Pengemudi yang kehilangan konsentrasi memiliki kecepatan reaksi yang lebih lambat terhadap perubahan lingkungan sekitar sehingga menimbulkan kemungkinan lebih tinggi untuk melakukan pelanggaran rambu lalu lintas. Pelanggaran rambu lalu lintas dapat menggangu kenyaman dan keamanan pengguna jalan lain, serta dapat menyebabkan terjadinya kecelakaan lalu lintas.

Hilangnya konsentrasi pengemudi dapat disebabkan oleh berbagai hal seperti gangguan *visual*, *auditory* dan *cognitive*. Berdasarkan survei yang dilakukan pada lebih dari 100 orang, 91 orang menyatakan bahwa mereka dapat mengemudi. Dari 91 orang tersebut, sebanyak 74 orang pernah mengalami gangguan konsentrasi yang menyebabkan pelanggaran rambu lalu lintas secara tidak disengaja. Dari 74 orang tersebut, 31 orang memiliki pengalaman mengemudi dibawah 4 tahun, 23 orang memiliki pengalaman mengemudi antara 4 sampai 9 tahun dan 20 orang memiliki pengalaman mengemudi lebih dari 9 tahun. Teralihkannya konsentrasi pengemudi adalah masalah yang dapat dialami oleh semua pengemudi termasuk pengemudi yang memiliki pengalaman mengemudi, oleh karena itu diperlukan sebuah sistem yang dapat membantu pengemudi dengan mendeteksi dan mengenali rambu lalu lintas secara *real-time*.

Penelitian deteksi rambu lalu lintas merupakan topik penelitian yang sudah banyak dilakukan. Banyak penelitian telah dilakukan menggunakan algoritma *machine learning* dan *computer vision* [4, 5, 7, 14, 17, 20]. Selain itu metode berbasis *deep learning* berbasis *convolutional neural network* (CNN) juga banyak digunakan untuk melakukan deteksi rambu lalu lintas [6, 8, 9, 12, 13, 18, 21, 22, 24]. Metode berbasis CNN termasuk dalam 2 kategori yakni 2 stage dan single stage. Metode 2 stage melakukan deteksi terlebih dahulu kemudian klasifikasi dilakukan pada area gambar yang dipercaya memiliki objek, sedangkan metode *single stage* melakukan deteksi dan klasifikasi objek secara bersamaan [23]. Contoh model yang menggunakan metode 2 stage adalah Mask R-CNN [3], Faster R-

CNN [16], FPN [10] dan R-FCN [2], sedangkan model yang menggunakan metode *single stage* seperti YOLO [15] dan SSD [11].

Metode - metode berbasis CNN berhasil menghasilkan performa akurasi deteksi yang tinggi pada deteksi rambu lalu lintas [13, 18]. Selain itu metode berbasis CNN dapat melakukan deteksi dengan kecepatan yang tinggi. Namun performa kecepatan tinggi ini didapat jika menggunakan *graphics processing unit* (GPU) untuk mempercepat proses perkalian *convolution matrix*, sehingga beberapa metode berbasis CNN sulit diimplementasikan pada *small computer / embedded device*. Oleh karena itu dalam penelitian ini akan menggunakan metode YOLOv4-tiny [19] untuk melakukan deteksi rambu lalu lintas secara *end-to-end*. YOLOv4-tiny dipilih karena performa deteksi yang lebih tinggi dibandingakn model varian tiny lainnya dengan kecepatan deteksi yang sangat tinggi.

2. TINJAUAN STUDI

Penelitian akan melakukan tinjauan studi pada penelitian topik terkait yang telah dilakukan sebelumnya. Berikut adalah penelitian-penelitian yang melakukan deteksi pada rambu lalu lintas:

2.1 Deep Learning for Large-Scale Traffic-Sign Detection and Recognition

Tabernik & Skočaj [18] memanfaatkan metode Mask R-CNN secara *end-to-end* untuk melakukan deteksi dan pengenalan pada rambu lalu lintas. Varian Mask R-CNN yang digunakan menggunakan ResNet-50 *architecture*. Dataset yang digunakan adalah Swedish Traffic-sign Dataset (STDS) dan DFG Traffic Dataset (DFG). Skor mAP@50 tertinggi yang berhasil dicapai adalah 95.2% pada dataset STDS dan 96.7% pada dataset DFG. Kelemahan penelitian ini adalah pada kecepatan deteksi Mask R-CNN yang rendah karena penggunaan arsitektur ResNet-50.

2.2 A Hierarchical Deep Architecture and Mini-Batch Selection Method For Joint Traffic Sign and Light Detection

Pon, et al. [13] memanfaatkan metode Faster R-CNN dengan arsitektur ResNet-50 yang dimodifikasi untuk melakukan deteksi dan klasifikasi rambu dan lampu lalu lintas. Modifikasi yang dilakukan adalah menggunakan *hierarchical object detection* dimana objek akan di klasifikasi kedalam superclass terlebih dahulu yakni rambu atau lampu lalu lintas, kemudian akan diklasifikasi lanjut kedalam subclass. Dataset yang digunakan ada 2 yakni Bosch Dataset untuk lampu lalu lintas dan Tsinghua-Tencent 100k Dataset untuk rambu lalu lintas. mAP@50 tertinggi yang berhasil dicapai untuk deteksi gabungan rambu dan lampu lalu lintas adalah 38%. Sedangkan untuk deteksi rambu lalu lintas saja berhasil mencapai skor mAP@50 40%. Kecepatan deteksi yang berhasil dicapai adalah 0.015 detik per gambar, performa ini dicapai dengan GPU NVIDIA GeForce GTX 1080 Ti. Kelemahan penelitian ini adalah pada kecepatan deteksi arsitektur ResNet-50 yang rendah.

2.3 Real-Time Detection and Recognition of Road Traffic Signs using MSER and Random Forests

Kuang, et al. [7] memanfaatkan metode Maximally Stable Extremal Regions (MSER) & Random Forest untuk melakukan deteksi dan klasifikasi pada rambu lalu lintas. Metode dipecah menjadi 2 bagian yakni detection stage dan recognition stage. Untuk *detection stage*, MSER digunakan untuk mencari *region of interest* (ROI) dalam gambar. Dalam *recognition stage* dibuat sebuah feature descriptor dengan nama HSV-HOG-LBP yang diadopsi bersamaan dengan random forest untuk mengklasifikasi rambu yang terdeteksi. Metode ini berhasil mencapai *recall* tertinggi di 95.38% dengan kecepatan deteksi 258 ms atau 0.258 detik. Kelemahan penelitian ini adalah kecepatan deteksi yang belum optimal, dan penggunaan metode *machine learning* yang memerlukan banyak optimisasi agar berjalan optimal.

2.4 Detecting Small Chinese Traffic Signs via Improved YOLOv3 Method

Zhang, et al. [22] memanfaatkan YOLOv3 + K-Means untuk melakukan deteksi pada rambu lalu lintas Tiongkok. Arsitektur YOLOv3 yang digunakan berbasis Darknet-53 dengan ukuran input 416 x 416. Dataset yang digunakan adalah dataset CSUST Chinese Traffic Sign Detection Benchmark (CCTSDB). K-Means digunakan untuk melakukan *re-clustering* pada *training set* untuk meningkatkan kecepatan deteksi dari model YOLOv3. Skor mAP@50 tertinggi yang dicapai adalah 98.8% dengan kecepatan deteksi 43.8 FPS pada GPU NVIDIA GeForce GTX 1080. Kelemahan penelitian ini adalah kecepatan deteksi YOLOv3 yang belum optimal jika dibandingkan dengan model YOLO varian tiny.

3. DATASET

Dataset yang digunakan dalam penelitian ini adalah dataset yang dikumpulakn secara mandiri. Dataset yang digunakan terdiri atas gambar *frame* yang diambil dijalan yang mengandung rambu lalu lintas yang akan dideteksi. Spesifikasi gambar yang digunakan adalah 1920 x 1080. Rambu yang digunakan dalam dataset ini ada 9 macam yang terdiri dari 7 rambu larangan dan 2 rambu perintah. Gambar *frame* dalam dataset didapat dari video rekaman jalan kota Surabaya yang direkam dari dalam mobil. Video rekaman kemudian diproses untuk ekstrak *frame*, jumlah *frame* yang di ekstrak adalah 1 dari setiap 6 *frame*. Hal ini dilakukan untuk meningkatkan variasi letak, bentuk dan ukuran rambu dalam gambar dalam dataset.

Saat mengekstrak *frame* dilakukan preprocessing untuk meningkatkan brightness gambar sebanyak 30. Setelah semua *frame* dari video selesai di ekstrak, penulis memilih secara manual gambar *frame* yang terkumpul dan memasukkan gambar terpilih kedalam dataset dengan memerhatikan jumlah kemunculan masing-masing rambu yang digunakan dalam dataset. Target jumlah kemunculan minimal untuk setiap macam rambu adalah 100 kemunculan.

Tabel 1. Rambu dan Jumlah Kemunculan dalam Dataset

Nama Rambu	Jumlah Kemunculan
Dilarang Parkir	205
Dilarang Berhenti	225
Dilarang Putar Balik	182
Dilarang Putar Balik dan Belok Kanan	171
Dilarang Belok Kanan	182
Dilarang Belok Kiri	150
Batas Kecapatan Maksimum 40km	195
Belok Kiri ikuti Isyarat Lampu	169
Belok Kiri Langsung	148
Dilarang Masuk	123
Dilarang Mendahului	169

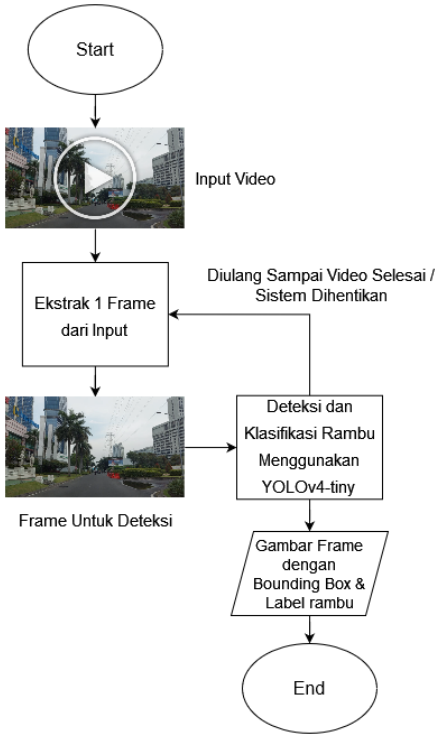
Tabel 1 menyebutkan 11 macam rambu yang digunakan serta jumlah kemunculan masing-masing rambu dalam dataset. Total

gambar yang terkumpul dalam dataset adalah 1367 gambar. Setelah terkumpul menjadi dataset, gambar dianotasi menggunakan format YOLO. Dataset kemudian akan dibagi kedalam kategori *train* dan *test*. Pembagian dataset kedalam kategori *train* dan *test* dilakukan secara random dengan memerhatikan kemunculan rambu dalam dataset. Rasio yang digunakan untuk membagi dataset adalah 75:25 jumlah kemunculan, *train* 75 dan *test* 25. Setelah dibagi menjadi *train* dan *test*, dataset *train* terdiri atas 1018 gambar dan dataset *test* terdiri atas 349 gambar.

Dalam penelitian ini juga dibuat 2 dataset preprocessing yakni dataset *grayscale* dan dataset *canny edge*. Dataset *grayscale* dan canny, dibuat berdasarkan dataset yang telah dikumpulkan sebelumnya namun dengan gambar yang dimodifikasi menjadi *grayscale* atau *canny edge*. Dataset preprocessing ini dibuat untuk mencari tahu performa model dengan dataset input yang berbeda.

4. METODE

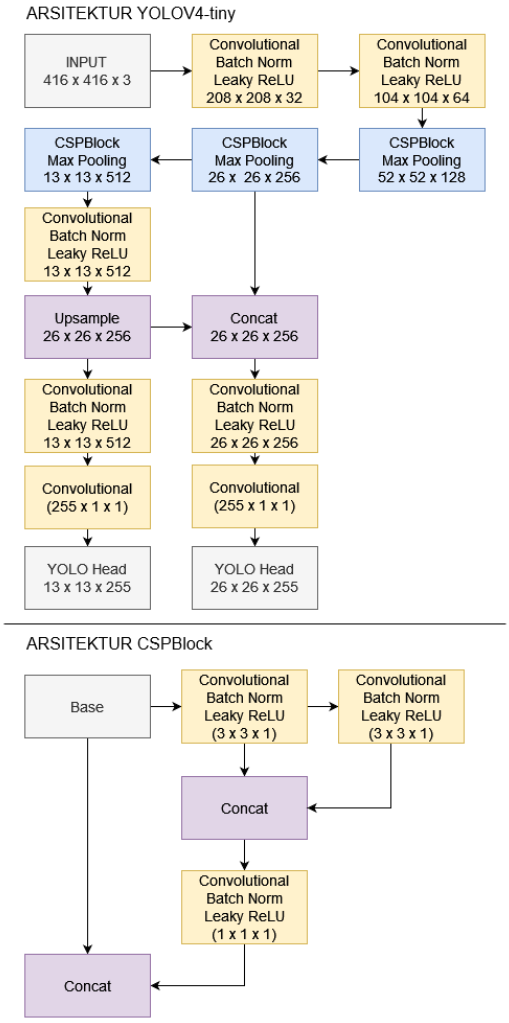
Metode yang digunakan dalam penelitian ini adalah YOLOv4-tiny [19] dengan ukuran input 416 x 416. YOLOv4-tiny digunakan secara end-to-end untuk melakukan deteksi dan klasifikasi rambu dalam gambar. YOLOv4-tiny digunakan secara end-to-end karena performa akurasi yang mampu dicapai dapat bersaing dengan algoritma state-of-the-art berukuran kecil lainnya, serta kecepatan deteksi YOLOv4-tiny yang sangat cepat.



Gambar 1. Alur Sistem Pengenalan Rambu

Gambar 1 menunjukkan alur sistem pengenalan rambu lalu lintas menggunakan YOLOv4-tiny. Sistem akan menerima input berupa video atau rekaman langsung jalanan. Sistem kemudian akan mengekstrak 1 *frame* dari input. *Frame* yang diekstrak kemudian di proses oleh model YOLOv4-tiny untuk melakukan deteksi dan klasifikasi rambu dalam gambar. Output sistem adalah rambu yang telah diberi *bounding box* dan label rambu yang terdeteksi.

YOLOv4-tiny merupakan model CNN yang merupakan varian tiny dari model YOLOv4 [1, 19] yang didesain agar mudah diimplementasi pada *low-end device*. YOLO sendiri adalah metode object detection yang melakukan deteksi dalam 1 tahap [23]. Hal ini membedakan model-model YOLO dengan model CNN lain seperti Mask R-CNN dan Faster R-CNN yang melakukan deteksi dalam 2 tahap [23]. Karena YOLO melakukan deteksi dan klasifikasi objek secara bersamaan, YOLO cenderung memiliki kecepatan deteksi yang lebih tinggi dibandingkan metode CNN yang melakukan deteksi dalam 2 tahap.



Gambar 2. Arsitektur YOLOv4-tiny dan CSPBlock

Gambar 2 menunjukkan arsitektur yang digunakan oleh YOLOv4-tiny. YOLOv4-tiny terdiri dari 5 komponen yakni; *convolutional*, CSPBlock, *upsample*, *concat*, dan YOLO *head*. Ukuran input yang digunakan dapat disesuaikan namun ukuran yang digunakan harus bilangan bulat kelipatan 32.

5. PENGUJIAN SISTEM

Pengujian yang akan dilakukan ada 4 pengujian pokok yakni pengujian program, pengujian pada dataset berwarna, pengujian pada dataset *preprocessing* dan pengujian jarak deteksi. Selain itu akan dilakukan perbandingan terhadap hasil implementasi metodeyang digunakan oleh penelitian sebelumnya [13, 18]. Terakhir akan dilakukan pengujian jarak deteksi model terhadap dataset *test*.

Pada pengujian program akan melakukan uji pada program pengenalan rambu lalu lintas Indonesia yang menggunakan model baseline yakni YOLOv4-tiny dengan ukuran input 416 x 416. dalam pengujian ini akan dihitung *recall* atau *true positivity rate* untuk mencari tahu performa deteksi program ketika dijalankan pada video dan rekaman langsung. *Recall* dihitung dengan membagi jumlah deteksi benar terhadap total kemunculan rambu dalam gambar.

$$recall = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (1)$$

Persamaan 1 menunjukkan rumus yang digunakan dalam menghitung *recall*. TP adalah jumlah rambu yang terdeteksi benar, sedangkan FN adalah jumlah rambu yang tidak terdeteksi ditambah jumlah rambu terdeteksi namun salah klasifikasi (Misklasifikasi).

Pada pengujian dataset berwarna dan preprocesisng, yang akan diukur adalah akurasi dan kecepatan deteksi rambu lalu lintas Indonesia pada berbagai varian model YOLOv4-tiny. Metrik evaluasi yang digunakan untuk menguji akurasi adalah mAP@50 (mAP) atau rata-rata *average precision* (AP) pada semua class dalam dataset dengan batas *threshold intersection over union* (IoU) 50%.

$$\text{mAP} = \frac{1}{n}\sum_{i=1}^{n} AP_i \quad (2)$$

Persamaan 2 menunjukkan perhitungan mAP dimana n menggambarkan jumlah class dan AP_i adalah hasil *average precision* dari *class* dengan *index* i.

Untuk menguji kecepatan deteksi menggunakan evaluation metric rata-rata *frames per second* (FPS) yang didapat dengan menguji model pada video. Pengujian dijalankan sebanyak 3 kali untuk setiap model untuk mencari rata-rata FPS yang dihasilkan, agar dapat mengetahui konsistensi performa model. Untuk pengujian FPS dilakukan pada *environment* berbasis CPU Intel I3-8100 dan RAM 8GB.

$$\text{FPS} = 1 \,/\, (end\ time - start\ time) \quad (3)$$

$$\text{Mean } FPS = \frac{\sum_{i=1}^{n} FPS_i}{n} \quad (4)$$

Persamaan 3 menunjukkan cara untuk menghitung FPS ketika mendeteksi sebuah *frame*. *Start time* adalah *checkpoint* waktu ketika sistem akan mulai membaca gambar, sedangkan *end time* adalah *checkpoint* waktu ketika sistem selesai melakukan deteksi pada gambar. Persamaan 4 menunjukkan cara untuk menghitung rata-rata FPS yang didapat jika melakukan deteksi pada input video. Notasi n menggambarkan total jumlah *frame* yang diproses oleh sistem dimana total jumlah FPS dari *frame* pertama hingga *frame* ke n akan dijumlah dan dibagi dengan n.

5.1 Pengujian Program

Pengujian ini dilakukan untuk mengetahui kemampuan program saat dijalankan dengan menggunakan input video dan rekaman langsung. Pengujian dilakukan menggunakan input rekaman langsung untuk lebih menggambarkan dan mensimulasikan ketika program digunakan dalam situasi riil.

Pengujian yang dilakukan ada 2 macam yakni pengujian *recall* dan pengujian performa program dibandingkan dengan manusia. Pengujian *recall* akan menggambarkan performa program mendeteksi secara benar saat sedang dijalankan. Pengujian performa program akan menggambarkan kemampuan program mendeteksi jika dibandingkan dengan manusia.

Untuk mengumpulkan data pengujian, program akan dijalankan 3 kali menggunakan video berbeda dan 2 kali menggunakan input rekaman langsung. Output dari menjalankan program sebanyak 5 kali adalah video output dari masing-masing percobaan. Video ini kemudian akan digunakan untuk 2 pengujian yang akan dilakukan dibawah ini

Pengujian *recall* / *true positivity rate* dilakukan dengan menghitung jumlah rambu yang terdeteksi benar di video dibagi denan jumlah rambu yang tidak terdeteksi ditambah jumlah rambu yang terdeteksi misklasifikasi. Dari masing-masing video dicari jumlah rambu terdeteksi benar, rambu terdeteksi salah dan rambu tidak terdeteksi untuk masing-masing rambu dalam dataset. Setelah ditemukan jumlah untuk masing-masing rambu, akan dijumlah menjadi total kemunculan rambu, total rambu terdeteksi benar, total rambu terdeteksi salah dan total rambu terdeteksi misklasifikasi per video. Setelah itu dilakukan perhitungan untuk mencari tahu skor *recall* dari masing-masing video.

Tabel 2. Hasil Perhitungan Recall atau True Posivity Rate dari 5 Video Pengujian Program

Input	Deteksi Benar	Mis-klasifikasi	Tidak Terdeteksi	Recall
Video 1	43	10	17	0.6143
Video 2	28	3	4	0.8000
Video 3	25	2	6	0.7576
Rekaman Langsung 1	40	0	17	0.7018
Rekaman Langsung 2	81	3	25	0.7714

Tabel 2 menunjukkan hasil pengujian *recall* dengan input yang berbeda-beda. Dari hasil pengujian *recall* pada 5 video, didapat rata-rata *recall* dengan skor 0.729 atau 72.9 %. Skor ini menunjukkan bahwa program mambu mendeteksi secara benar 72.9 % dari rambu yang ditemukan ketika menjalankan program. Berdasarkan observasi dari video yang digunakan dalam pengujian, ditemukan program yang menggunakan input rekaman langsung memiliki gambar yang patah-patah karena rendahnya kecepatan program yang menyebabkan muncul blur pada frame dalam video.

Tabel 3. Tabel Selisih Performa Program terhadap Manusia

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 hasil Tabel 3, dapat dilihat jika 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

Pengujian dataset berwarna berisi 2 pengujian yakni pengujian akurasi deteksi pada dataset *test* dan pengujian kecepatan deteksi pada video *test*. Pengujain akurasi deteksi akan diukur menggunakan metrik mAP@50 sedangkan pengujian kecepatan deteksi akan menggunakan metrik FPS.

Model yang akan diuji pada bagian ini adalah model YOLOv4-tiny dengan 3 variasi ukuran input yang di *train* menggunakan dataset berwarna yang dibuat dalam penelitian ini. Variasi ukuran input yang akan diuji adalah 416 (416 x 416), 480 dan 320. Pengujian dilakukan menggunakan 3 variasi input, untuk mencari tahu pengaruh ukuran input terhadap akurasi dan kecepatan deteksi dari model YOLOv4-tiny.

Tabel 4. Hasil Akurasi dan Kecepatan Deteksi YOLOv4-tiny pada Dataset Berwarna

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

Tabel 4 menunjukkan performa akurasi dan kecepatan deteksi dari ketiga model YOLOv4-tiny yang memiliki ukuran input berbeda. Dari ketiga model, model dengan ukuran input 480 x 480 mampu mencapai performa akurasi tertinggi dengan skor mAP@50 89.58, sedangkan model dengan ukuran input 320 x 320 mampu mencapai performa kecepatan deteksi tertinggi dengan nilai FPS 30.02 FPS. Karena model yang meraih akurasi tertinggi dan model yang meraih kecepatan deteksi tertinggi berbeda, agak sulit memilih model terbaik. Oleh karena itu dipilih model YOLOv4-tiny dengan ukuran input 416 x 416 dipilih sebagai model *baseline* karena memiliki nilai akurasi yang cukup baik dan kecepatan deteksi yang tergolong mendekati *real-time*. Model *baseline* dipilih untuk digunakan dalam program pengenalan rambu lalu lintas Indonesia di pengujian sebelumnya

5.3 Pengujian Dataset Preprocessing

Sama seperti bagian 5.2, bagian ini akan berisi 2 pengujian yakni pengujian akurasi dan pengujian kecepatan deteksi pada video *test*. Metrik yang digunakan untuk mengukur akurasi adalah mAP@50 sedangkan metrik untuk mengukur kecepatan deteksi adalah FPS.

Pada pengujian ini akan melakukan *preprocessing* berupa mengubah gambar dari dataset original menjadi *grayscale* dan canny edge. Hal ini dilakukan untuk mencari tahu apakah ada keuntungan dalam melakukan *preprocessing* terhadap akurasi dan kecepatan deteksi model. Pada dataset *grayscale* akan menggunakan 3 variasi input yakni 416, 480 dan 320 sedangkan dataset canny hanya akan diuji pada ukuran 416. Dataset canny hanya diuji pada ukuran 416 karena ditemukan perbedaan yang sangat jauh antara akurasi dataset canny terhadap akurasi yang lain.

Tabel 5 menunjukkan hasil akurasi dan kecepatan deteksi yang dicapai oleh model YOLOv4-tiny yang di *train* menggunakan macam-macam variasi input. Secara umum model yang di *train* menggunakan dataset berwarna memiliki akurasi yang lebih tinggi, sedangkan model yang di-*train* menggunakan dataset *grayscale* memiliki kecepatan deteksi yang lebih tinggi. Model dengan akurasi terbaik adalah model yang di-*train* pada dataset berwarna dengan ukuran input 480 x 480 dan model dengan kecepatan deteksi tertinggi diraih oleh model yang di-*train* pada dataset *grayscale* dengan ukuran input 320 x 320.

Tabel 5. Perbandingan Akurasi dan Kecepatan Deteksi Model YOLOv4-tiny pada Dataset Grayscale dan Canny

Dataset	Ukuran Input	mAP@50	FPS
Dataset Berwarna	416 x 416	88.55	19.41
	480 x 480	**89.58**	15.81
	320 x 320	86.01	30.02
Dataset *Grayscale*	416 x 416	80.64	20.58
	480 x 480	86.39	16.12
	320 x 320	75.97	**30.87**
Dataset Canny	416 x 416	64.06	18.16

5.4 Perbandingan terhadap Penelitian Sebelumnya

Pengujian ini akan membandingkan performa akurasi dan keccepatan deteksi yang didapat dari implementasi model yang digunakan pada paper [13, 18]. Model yang akan diimplementasi adalah Mask R-CNN [18] dan Faster R-CNN [13]. Model Mask R-CNN dan Faster R-CNN akan di-*train* menggunakan dataset berwarna yang digunakan pada saat *train* YOLOv4-tiny.

Implementasi Mask R-CNN [3] akan menggunakan *backbone* ResNet-50 seperti yang digunakana pada penelitian [18]. Ukuran input akan diubah menjadi 448 (salah satu kelipatan 64 terdekat dari 416). *Training* dilakukan secara *transfer learning* dari *pretrained weight* ImageNet. Parameter *training* yang digunakan adalah *learning rate* = 0.0025, *weight decay* 0.0001, *momentum* = 0.9. *Epoch training* yang dipakai adalah 80 *epoch* pada *head* ditambah 20 *epoch* pada *all layer* (80:20).

Implementasi Faster R-CNN [16] akan menggunakan *backbone* yang sama dengan Mask R-CNN yakni ResNet50. Ukuran input yang digunakan yakni 739 x 416. Parameter *training* yang akan digunakan adalah parameter *default* dari konfigurasi Faster R-CNN pada *framework* Detectron2.

Tabel 6. Perbandingan Akurasi dan Kecepatan Deteksi YOLOv4-tiny terhadap Metode dari Penelitian Sebelumnya

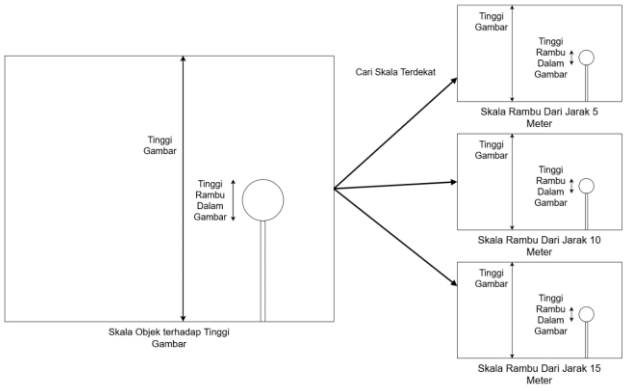
Model	Ukuran Input Model	mAP@50	FPS
YOLOv4-tiny	416 x 416	88.55	19.41
	480 x 480	**89.57**	15.81
	320 x 320	86.01	**30.02**
Mask R-CNN	448 x 448	70.0	0.73
Faster R-CNN	739 x 416	82.32	0.85

Tabel 6 menunjukkan perbandingan akurasi dan kecepatan deteksi model YOLOv4-tiny terhadap hasil implelementasi metode yang digunakan oleh penelitian sebelumnya yakni Mask R-CNN dan Faster R-CNN [13, 18]. Metode YOLOv4-tiny berhasil mencapai tingkat akurasi yang lebih tinggi dibandingkan dengan metode yang digunakan pada penelitian sebelumnya. Selain itu metode YOLOv4-tiny juga memiliki kecepatan deteksi jauh lebih tinggi hingga lebih dari 15x lipat dibandingkan kecapatan deteksi yang berhasil dicapai oleh metode Mask R-CNN dan Faster R-CNN.

5.5 Pengujian Jarak Deteksi

Pengujian ini dilakukan untuk mengukur kemampuan metode YOLOv4-tiny dalam mendeteksi rambu pada jarak tertentu. Model yang akan digunakan untuk menguji jarak adalah model *baseline* YOLOv4-tiny dengan ukuran input 416 x 416.

Untuk mengelompokkan rambu dalam dataset *test*, akan dilakukan klasifikasi dengan menghitung skala tinggi rambu dalam gambar terhadap tinggi gambar. Agar rambu dapat dikategorisasikan perlu menghitung skala tolok ukur dari rambu berbentuk lingkaran dan persegi panjang pada 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 3. Illustrasi Kategorisasi Rambu

Gambar 3 mengilustrasikan 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 7. Hasil Pengujian Jarak Deteksi Metode YOLOv4-tiny

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

Tabel 7 menampilkan jumlah rambu yang berhasil dideteksi benar terhadap jumlah rambu sebenarnya dalam dataset untuk rambu dan jarak tertentu. Total *recall* / *true positivity rate* keseluruhan dari pengujian jarak adalah 80.24%. Namun dilihat berdasarkan hasil pengujian jarak, model mampu mendeteksi dengan cukup baik di jarak kurang lebih 5 hingga 10 meter, sedangkan pada jarak kurang lebih 15 meter, model kesulitan mengenali rambu.

6. KESIMPULAN

Penelitian ini melakukan deteksi pada rambu lalu lintas Indonesia menggunakan metode YOLOv4-tiny secara *end-to-end*. Berdasarkan hasil pengujian program pengenalan rambu lalu lintas Indonesia, ditemukan bahwa program yang memuat model *baseline* YOLOv4-tiny dengan ukuran input 416 x 416 mampu mendeteksi dengan tingkat akurasi rata-rata sebesar 72.9%. Namun program memiliki kelemahan dimana program memiliki rata-rata 1.461 detik lebih lambat dibandingkan manusia, dengan catatan manusia yang dijadikan pembanding secara sadar dan aktif mencari rambu-rambu. Selain itu untuk penggunaan secara riil time, terdapat penurunan kecepatan deteksi yang cukup drastis jika program dijalankan pada laptop dengan status *on battery*. Kecepetan deteksi turun dari kurang lebih 19 FPS menjadi kurang lebih 6 FPS. Penurunan kecepatan deteksi menyebabkan frame yang dideteksi oleh program menjadi *blur* dan program berjalan dengan patah-patah.

Berdasarkan hasil pengujian pada dataset, dapat disimpulkan jika metode YOLOv4-tiny mampu mendeteksi semua jenis rambu yang digunakan dalam dataset. Metode YOLOv4-tiny mampu mencapai hasil akurasi yang setara dengan metode *state-of-the-art.* Model YOLOv4-tiny dengan ukuran input 480 x 480 berhasil mencapai skor mAP@50 89.58%. Dari segi kecepatan deteksi, YOLOv4-tiny berhasil mencapai kecepatan deteksi hingga lebih dari 30 FPS pada model dengan ukuran input 320 x 320 ketika dijalankan menggunakan CPU. Selain itu ditemukan bahwa pemberlakuan preprocessing untuk mengubah gambar dataset menjadi *grayscale* dapat meningkatkan kecepatan deteksi meskipun mengalami pengurangan pada akurasi deteksi. Dalam pengujian jarak deteksi rambu, metode YOLOv4-tiny mampu mendeteksi dengan akurasi 80.24%, dimana performa terbaik ketika mendeteksi rambu dari jarak dekat hingga 10 meter. Setelah 10 meter ditemukan penurunan kemampuan deteksi rambu.

7. SARAN

Dibawah ini adalah saran yang dapat dilakukan untuk mengembangkan penelitian lebih lanjut:

1. Menambah jumlah gambar dalam dataset dan gunakan rambu dengan posisi, jarak, sudut pandang dan kondisi pencahayaan yang berbeda.
2. Menambah kemunculan rambu yang diambil pada malam hari.
3. Melakukan implementasi dan optimisasi pada *small computer* dan *embedded device*.
4. Mengembangkan program pengenalan rambu agar dapat dipergunakan pada kendaraan tanpa pengemudi.

8. REFERENSI

[1] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*. DOI: 10.48550/arXiv.2004.10934

[2] Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. *Advances in neural information processing systems*, *29*. DOI: 10.48550/arXiv.1605.06409

[3] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961-2969). DOI: 10.48550/arXiv.1703.06870

[4] Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., & Igel, C. (2013, August). Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *The 2013 international joint conference on neural networks (IJCNN)* (pp. 1-8). IEEE. DOI: 10.1109/IJCNN.2013.6706807

[5] Huang, Z., Yu, Y., Gu, J., & Liu, H. (2016). An efficient method for traffic sign recognition based on extreme learning machine. *IEEE transactions on cybernetics*, *47*(4), 920-933. DOI: 10.1109/TCYB.2016.2533424

[6] Karaduman, M., & Eren, H. (2017, October). Deep learning based traffic direction sign detection and determining driving style. In *2017 International Conference on Computer Science and Engineering (UBMK)* (pp. 1046-1050). IEEE. DOI: 10.1109/UBMK.2017.8093453

[7] Kuang, X., Fu, W., & Yang, L. (2018). Real-Time Detection and Recognition of Road Traffic Signs using MSER and Random Forests. *International Journal of Online and Biomedical Engineering (iJOE)*, *14*(03), pp. 34–51. DOI: 10.3991/ijoe.v14i03.7925

[8] Li, D., Zhao, D., Chen, Y., & Zhang, Q. (2018, July). Deepsign: Deep learning based traffic sign recognition. In *2018 international joint conference on neural networks (IJCNN)* (pp. 1-6). IEEE. DOI: 10.1109/IJCNN.2018.8489623

[9] Lin, C., Li, L., Luo, W., Wang, K. C., & Guo, J. (2019). Transfer learning based traffic sign recognition using inception-v3 model. *Periodica Polytechnica Transportation Engineering*, *47*(3), 242-250. DOI: 10.3311/PPtr.11480

[10] Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2117-2125). DOI: 10.48550/arXiv.1612.03144

[11] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham. DOI: 10.48550/arXiv.1512.02325

[12] Liu, Z., Qi, M., Shen, C., Fang, Y., & Zhao, X. (2021). Cascade saccade machine learning network with hierarchical classes for traffic sign detection. *Sustainable Cities and Society*, *67*, 102700. DOI: 10.1016/j.scs.2020.102700

[13] Pon, A., Adrienko, O., Harakeh, A., & Waslander, S. L. (2018, May). A hierarchical deep architecture and mini-batch selection method for joint traffic sign and light detection. In *2018 15th Conference on Computer and Robot Vision (CRV)* (pp. 102-109). IEEE. DOI: 10.1109/CRV.2018.00024.

[14] Ramík, D. M., Sabourin, C., Moreno, R., & Madani, K. (2014). A machine learning based intelligent vision system for autonomous object detection and recognition. *Applied intelligence*, *40*(2), 358-375. DOI: 10.1007/s10489-013-0461-5

[15] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788). DOI: 10.1109/CVPR.2016.91

[16] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, *28*. DOI: 10.48550/arXiv.1506.01497

[17] Swathi, M., & Suresh, K. V. (2017, February). Automatic traffic sign detection and recognition: A review. In *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)* (pp. 1-6). IEEE. DOI: 10.1109/ICAMMAET.2017.8186650

[18] Tabernik, D., & Skočaj, D. (2019). Deep learning for large-scale traffic-sign detection and recognition. *IEEE transactions on intelligent transportation systems*, *21*(4), 1427-1440. DOI: 10.1109/TITS.2019.2913588

[19] Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (2021). Scaled-yolov4: Scaling cross stage partial network. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition* (pp. 13029-13038). DOI: 10.1109/cvpr46437.2021.01283

[20] Yang, Y., Luo, H., Xu, H., & Wu, F. (2015). Towards real-time traffic sign detection and classification. *IEEE Transactions on Intelligent transportation systems*, *17*(7), 2022-2031. DOI: 10.1109/TITS.2015.2482461

[21] Zang, D., Wei, Z., Bao, M., Cheng, J., Zhang, D., Tang, K., & Li, X. (2018). Deep learning–based traffic sign recognition for unmanned autonomous vehicles. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, *232*(5), 497-505. DOI: 10.1177/0959651818758865

[22] Zhang, B., Wang, G., Wang, H., Xu, C., Li, Y., & Xu, L. (2021). Detecting small chinese traffic signs via improved yolov3 method. *Mathematical Problems in Engineering*, *2021*. DOI: 10.1155/2021/8826593

[23] Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2019). Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, *30*(11), 3212-3232. DOI: 10.1109/TNNLS.2018.2876865

[24] Zhu, Z., Liang, D., Zhang, S., Huang, X., Li, B., & Hu, S. (2016). Traffic-sign detection and classification in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2110-2118). DOI: 10.1109/cvpr.2016.232