DOI: 10.22146/jnteti.v13i1.8795

Penerapan *Computation Offloading* Pada Sistem Deteksi Pelanggaran Perlintasan Sebidang Berbasis Komputasi Tepi

Rian Putra Pratama[1], Suhono Harso Supangkat[2]

[1] Pusat Riset Mekatronika Cerdas, Badan Riset dan Inovasi Nasional, Bandung, Indonesia
[2] Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, Bandung, Indonesia

[Diserahkan: 28 Juli 2023, Direvisi: 13 November 2023, Diterima: 18 Desember 2023]
Penulis Korespondensi: Rian Putra Pratama (email: rian011@brin.go.id)

INTISARI — Perlintasan sebidang masih menjadi permasalahan di beberapa kota akibat tingkat pelanggaran yang tinggi. Saat ini, pengawasan di perlintasan sebidang masih dilakukan secara konvensional. Permasalahan di perlintasan sebidang makin kompleks dan solusi konvensional tidak lagi efektif, sehingga diperlukan sistem pengawasan video cerdas sebagai solusi. Penerapan sistem pengawasan video cerdas merupakan tugas yang kompleks dan memerlukan perangkat dengan sumber daya komputasi besar. Penelitian ini bertujuan mengoptimalkan sistem dalam melakukan pemrosesan data secara *real-time* dengan melakukan komputasi dekat dengan sumber data dan membagi tugas komputasi ke beberapa perangkat tepi (*edge device*). Sistem pengawasan video cerdas berbasis komputasi tepi (*edge computing*) dengan metode *computation offloading* pada perangkat terbatas diusulkan sebagai solusi pada penelitian ini. Penelitian ini terdiri atas dua tahap pengembangan. Tahap pertama mengembangkan model deteksi objek menggunakan *dataset* perlintasan sebidang di Kota Bandung dan tahap kedua mengembangkan sistem berbasis komputasi tepi dengan menerapkan metode *computation offloading* pada perangkat komputasi terbatas. Metode komputasi tepi memperluas komputasi awan (*cloud computing*) ke tepi jaringan untuk melakukan perhitungan dekat dengan sumber data, sedangkan metode *computation offloading* meningkatkan kinerja komputasi tepi dengan membagi tugas komputasi. Hasil pengujian menunjukkan terjadinya peningkatan kecepatan komputasi sekitar 1,5 kali lebih cepat dengan tingkat akurasi deteksi pelanggaran mencapai 89,4%. Selain itu, terjadi penurunan suhu GPU sebesar 5,50 °C, penggunaan GPU turun 44,05%, penggunaan memori berkurang 301 Mb, dan konsumsi daya menurun 2,28 W. Sistem yang dikembangkan efektif dan efisien dalam mengoptimalkan kinerja sistem deteksi pelanggaran di perlintasan sebidang pada perangkat komputasi terbatas.

KATA KUNCI — *Computation Offloading*, Komputasi Tepi, Sistem Deteksi Pelanggaran, Perlintasan Sebidang, Sistem Pengawasan Video Cerdas.

I. PENDAHULUAN

Perlintasan sebidang adalah pertemuan antara jalur kereta api dan jalan raya [1]. Direktorat Jenderal Perhubungan Darat Republik Indonesia menyebutkan bahwa pengawasan terhadap aspek keamanan dan keselamatan di perlintasan sebidang sangat penting. Permasalahan di perlintasan sebidang umumnya disebabkan oleh tingkat pelanggaran yang cukup tinggi dan kurangnya pengawasan, sehingga meningkatkan risiko terjadinya kecelakaan. Pelanggaran umumnya terjadi saat pengendara melanggar palang pintu perlintasan yang telah ditutup [2]. Permasalahan di perlintasan sebidang makin kompleks dan solusi konvensional tidak lagi efektif dalam mengatasi masalah tersebut. Oleh karena itu, diperlukan pendekatan teknologi sistem pengawasan di perlintasan sebidang sebagai solusi untuk mengurangi pelanggaran lalu lintas dan potensi kecelakaan [3]. Sistem pengawasan cerdas di perlintasan sebidang merupakan bagian dari pengembangan *smart city* dari aspek *safe and secure* [4].

Saat ini, sistem pengawasan video cerdas dengan memanfaatkan visi komputer (*computer vision*) memiliki kemampuan untuk mengintegrasikan beberapa algoritma analisis gambar dan video [5]. Penerapan sistem pengawasan video cerdas dalam pengembangan *smart city* melalui pendekatan visi komputer telah berkembang signifikan selama satu dekade terakhir [6]. Teknologi sistem pengawasan video cerdas diterapkan dalam mengontrol lalu lintas, mendeteksi aktivitas di jalan raya, dan memonitor perilaku pengguna jalan [7]. Penelitian sistem pengawasan video cerdas berupaya menggantikan operator manusia atau sistem konvensional dengan algoritma pemrosesan video yang mampu melakukan tugas secara otonom [8].

State of the art sistem pengawasan video cerdas saat ini difokuskan pada tiga tugas utama, yaitu deteksi, pelacakan, dan pengenalan aktivitas atau pemahaman terhadap perilaku dan situasi tertentu dengan menggunakan jaringan saraf dalam algoritma *deep neural networks* (DNN) [9]. Proses deteksi, pelacakan, dan pengenalan aktivitas menggunakan DNN merupakan tugas yang kompleks dan memerlukan perangkat dengan sumber daya komputasi yang besar [10]. Implementasi DNN di komputasi awan (*cloud computing*) membutuhkan waktu dan beban komputasi yang signifikan. Komputasi awan menerima seluruh data video yang diunggah dari perangkat *internet of things* (IoT) berupa sensor kamera dan melakukan komputasi, sehingga menyebabkan peningkatan biaya transmisi data dalam jaringan, termasuk latensi, *bandwidth*, dan sumber daya komputasi [11].

Dari hal tersebut, muncullah paradigma komputasi tepi (*edge computing*) yang dapat mengatasi permasalahan peningkatan biaya transmisi data dalam jaringan pada komputasi awan [12]. Komputasi tepi melakukan proses komputasi lebih dekat ke sumber data, sehingga latensi komunikasi, *bandwidth*, dan sumber daya dapat dikurangi [13], [14]. Pada dasarnya, komputasi tepi dilakukan untuk memperluas komputasi awan ke tepi jaringan dengan tujuan

melakukan perhitungan komputasi dekat dengan sumber data, yaitu pada perangkat tepi (*edge device*) [15]. Dengan memanfaatkan komputasi tepi, pemrosesan data dapat dilakukan secara lokal di tepi jaringan, sehingga dapat mengurangi beban komputasi dan menjalankan tugas dengan efisien tanpa harus bergantung pada infrastruktur awan (*cloud*).

Tantangan utama dalam penerapan sistem pengawasan video cerdas pada perangkat tepi adalah sumber daya komputasi yang terbatas. Perangkat komputasi terbatas memiliki kekurangan berupa keterbatasan dari sisi kapasitas pemrosesan atau daya komputasi. Keterbatasan ini dapat mencakup berbagai aspek, seperti kecepatan pemrosesan, kapasitas memori, dan kemampuan untuk menangani tugas-tugas yang membutuhkan daya komputasi tinggi. Optimalisasi diperlukan untuk mencapai kondisi ideal untuk mengembangkan solusi yang efektif dan efisien, salah satunya dengan metode *computation offloading* [16]. *Computation offloading* adalah metode untuk meningkatkan kinerja komputasi tepi dengan cara membagi tugas komputasi ke perangkat yang memiliki sumber daya komputasi lebih besar guna mengatasi keterbatasan sumber daya komputasi pada perangkat tepi [17].

Penelitian sistem pengawasan video cerdas di perlintasan sebidang telah dilakukan di berbagai negara. Sebuah penelitian merancang sistem pengawasan perlintasan sebidang di negara bagian New Jersey dan Kota Ashland, Virginia, Amerika Serikat, dengan pendekatan visi komputer menggunakan metode *region of interest* (ROI) dan model deteksi objek *mask region with convolutional neural network* (R-CNN) [18]. Referensi [19] mengembangkan solusi teknologi untuk mengawasi dan membuat keputusan cerdas di perlintasan sebidang menggunakan arsitektur MobileNet dan algoritma (CNN) untuk klasifikasi objek dan deteksi hambatan. Penelitian lain di negara Ceko melakukan deteksi dan klasifikasi di perlintasan sebidang serta peringatan kereta api dengan model DNN YOLOv3 yang diimplementasikan pada perangkat tepi [20].

Penelitian ini mengembangkan sistem pengawasan perlintasan sebidang berbasis komputasi tepi yang dapat mendeteksi pelanggaran dengan mengembangkan model deteksi dan pengenalan objek serta mengoptimalkan kinerja komputasi menggunakan metode *computation offloading* pada perangkat komputasi terbatas. Tahapan pertama melakukan pembentukan model deteksi dan pengenalan objek dengan menggunakan algoritma deteksi objek, mengacu pada penelitian sebelumnya [21], [22]. Tahapan kedua menerapkan hasil pembentukan model pada perangkat dengan sumber daya komputasi terbatas menggunakan metode *computation offloading* dengan tujuan mempercepat waktu inferensi dan mengurangi beban komputasi [23]. Tahapan ketiga melakukan evaluasi dan skenario pengujian untuk mengukur efektivitas dan efisiensi prototipe sistem [24].

II. METODOLOGI

Penelitian ini merujuk pada penelitian sebelumnya yang merancang teknologi sistem pengawasan untuk perlintasan sebidang. Meskipun penelitian sebelumnya menerapkan pendekatan komputasi tepi pada arsitektur sistem yang dikembangkan, penerapannya pada satu perangkat tepi memiliki keterbatasan sumber daya, termasuk memori dan kemampuan pemrosesan komputasi yang terbatas.

Dalam desain arsitektur yang dikembangkan pada penelitian ini, terdapat tiga lapisan, yaitu lapisan perangkat akhir (*end device layer*), lapisan tepi (*edge layer*), dan lapisan layanan awan (*cloud service layer*). Proses pengembangan terdiri atas dua tahap. Tahap pertama melakukan pengembangan model deteksi objek untuk mengenali objek di lingkungan perlintasan sebidang dengan mengumpulkan *dataset* di salah satu perlintasan sebidang di kota Bandung. Tahap kedua melakukan pengembangan sistem berbasis komputasi tepi dengan menerapkan *computation offloading* pada perangkat dengan sumber daya komputasi terbatas, yaitu membagi proses komputasi ke dalam dua perangkat tepi.

Pengujian dalam penelitian ini dilakukan dengan dua skenario untuk membandingkan kinerja komputasi pada perangkat komputasi terbatas. Skenario pertama menggunakan rancangan sistem dari penelitian sebelumnya, sementara skenario kedua menerapkan metode *computation offloading* yang dikembangkan dalam penelitian ini.

A. PEMBENTUKAN MODEL

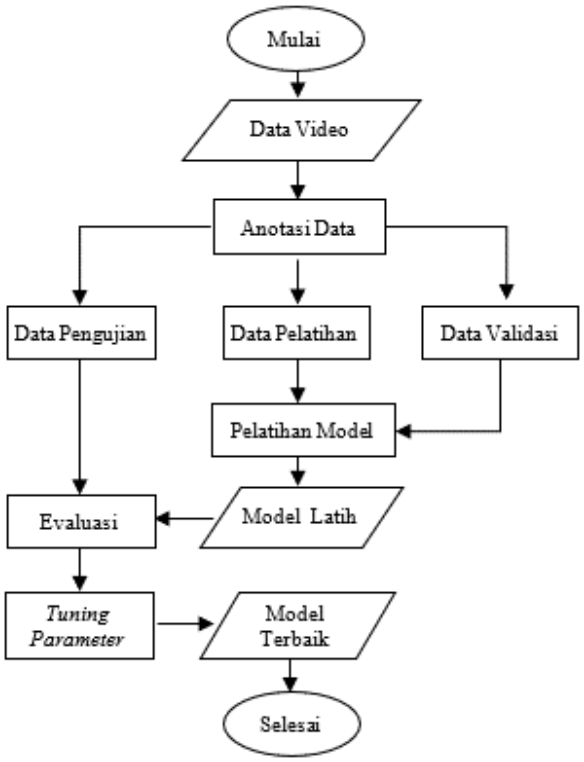
Proses pengembangan sistem dalam penelitian ini dimulai dengan alur proses pembentukan model yang melibatkan langkah-langkah: pengumpulan *dataset*, pelatihan model, dan evaluasi model [25]. Pembentukan model ini bertujuan untuk melakukan deteksi dan pengenalan objek pelanggaran di area perlintasan sebidang yang diimplementasikan di perangkat komputasi terbatas. Proses pembentukan model diawali dengan tahapan persiapan data. Data mentah diubah menjadi data dengan format yang lebih terstruktur dan siap digunakan. Selanjutnya, proses pembagian data dibagi menjadi tiga bagian. Bagian pertama, yaitu data pelatihan, adalah subset data yang digunakan untuk melatih model prediktif. Kedua, data validasi merupakan subset data yang digunakan untuk mengevaluasi kinerja model selama proses pelatihan. Sementara itu, data pengujian adalah subset terpisah dari data yang tidak digunakan selama proses pelatihan dan validasi, yang digunakan untuk menguji kinerja model terlatih terhadap data baru yang tidak dilihat sebelumnya. Tahapan pembentukan model pada penelitian ini dijelaskan sebagai berikut.

1) PENGUMPULAN *DATASET*

Proses pengumpulan *dataset* diawali dengan melakukan perekaman video di salah satu perlintasan sebidang di Kota Bandung. Dari proses tersebut, terkumpul 3.000 citra yang terdiri atas 21.000 anotasi dari enam kelas, dengan rincian jumlah anotasi 8.638 motor, 6.150 mobil, 2.089 truk, 1.288 palang pintu, 1.025 angkot, dan 83 kereta api. Seperti ditunjukkan pada Gambar 1, *dataset* citra terbagi menjadi tiga bagian dengan rasio 7:2:1, yaitu 70% data sebagai data pelatihan, 20% data validasi, dan 10% data pengujian atau 2.200 data pelatihan, 655 data validasi, dan 324 data pengujian. Proses selanjutnya adalah melakukan pengubahan ukuran (*resize*) terhadap citra, dengan ukuran 416 piksel dan 640 piksel, dengan pertimbangan waktu pelatihan model dan berdasarkan beberapa konfigurasi ukuran piksel lainnya.

2) PELATIHAN MODEL

Pada tahapan pelatihan model, *dataset* yang telah dikumpulkan dilatih menggunakan arsitektur CNN CSPDarknet53 pada YOLOv5s menggunakan *library* PyTorch yang terdapat pada *file* custom_yolov5s.yaml [26]. YOLOv5s adalah salah satu algoritma populer dalam pendeteksian objek. Menurut para peneliti di AI Research, arsitektur terpadu YOLOv5s sangat sederhana. Jaringan konvolusi tunggal membuat YOLOv5s langsung mendeteksi objek dengan hanya melewati jaringan saraf sekali [27]. YOLOv5s terdiri atas tiga



Gambar 1. Diagram alir pembentukan model.

bagian, yaitu *backbone*, *neck*, dan *head*. *Backbone* bertindak sebagai pengekstraksi fitur, *neck* bertindak sebagai pengagregasi fitur, dan *head* bertindak untuk melakukan lokalisasi serta klasifikasi pada setiap *bounding box*. *Backbone* mengekstrak fitur menggunakan BottleNeckCSP dan SPP, sedangkan *neck* mengumpulkan fitur-fitur *upsample*. Hasilnya ada di bagian *head*. YOLOv5s akan melakukan klasifikasi objek dengan jaringan konvolusi.

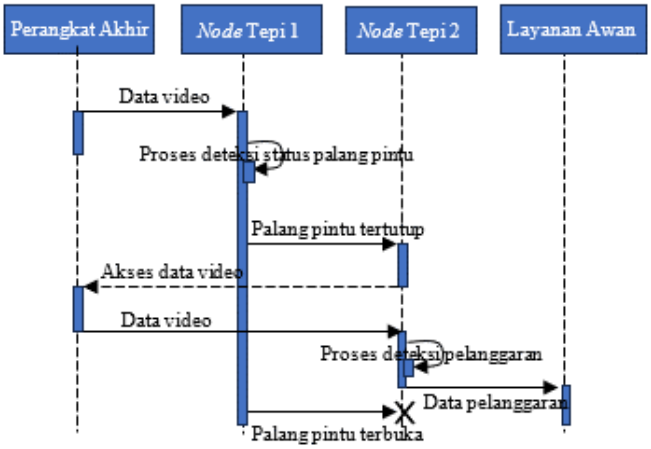
Proses pelatihan dilakukan menggunakan kode Python dengan *file* train.py yang terdapat pada *library* model YOLOv5 menggunakan variasi data ukuran citra, jumlah *epoch*, dan ukuran *batch* [28]. Pelatihan model ini melibatkan konfigurasi dengan variasi ukuran citra 416 piksel dan 640 piksel; jumlah *epoch* 50, 100, dan 150; serta ukuran *batch* 16, 32, dan 64 data. Ukuran citra memengaruhi kemampuan model menangkap detail, jumlah *epoch* menentukan tingkat keberhasilan model belajar dari *dataset*, dan ukuran *batch* memengaruhi efisiensi pelatihan. Eksperimen dilakukan untuk mencari parameter yang paling efektif dan efisien dalam melakukan deteksi objek data pelanggaran di perlintasan sebidang.

3) EVALUASI MODEL

Pada tahapan evaluasi, proses pelatihan model yang telah dilakukan akan menghasilkan nilai presisi, *recall*, dan *mean average precision* (mAP). Metrik mAP digunakan untuk mengukur rata-rata akurasi dari setiap kelas. Jika nilai mAP belum cukup baik, perlu dilakukan perubahan konfigurasi parameter pada proses pelatihan model sampai didapatkan nilai mAP yang baik. Pada penelitian ini, dari hasil beberapa variasi konfigurasi proses pelatihan model, diperoleh model dengan nilai mAP yang paling optimal, yaitu dengan konfigurasi parameter ukuran citra 640 dengan ukuran *batch* 32, dan jumlah *epoch* 100. Dari hasil pelatihan model seperti pada Tabel I, diperoleh dua model yang paling optimal dengan masing-masing nilai mAP@0,5 model 1 sebesar 0,993 dan mAP@0,5 model 2 sebesar 0,852. Metrik mAP@0,5 adalah metrik yang mengukur kinerja deteksi objek pada tingkat

TABEL I
HASIL PELATIHAN MODEL

Parameter	Model 1	Model 2
Objek	palang pintu perlintasan	motor, mobil, truk angkot, dan kereta api
Konfigurasi parameter	*img* 640 *batch* 32 *epoch* 100	*img* 640 *batch* 32 *epoch* 100
Presisi	0,988	0,808
Recall	0,965	0,751
mAP	0,993	0,852



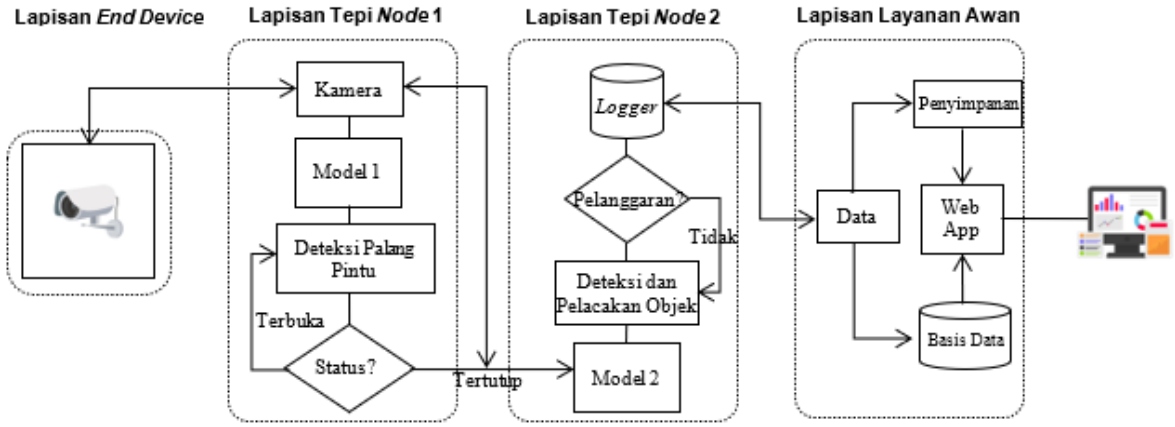
Gambar 2. *Sequence diagram* pengembangan sistem.

kepercayaan 0,5, yang menunjukkan keberhasilan sistem dalam mendeteksi objek dengan nilai kepercayaan prediksi 0,5 atau lebih tinggi [29].

Model 1 terdiri atas satu kelas objek palang pintu perlintasan sebidang, yang berfungsi untuk mengenali pergerakan objek palang pintu perlintasan sebidang. Model 2 terdiri atas lima kelas, yaitu motor, mobil, truk, angkot, dan kereta api. Model 2 berfungsi untuk mendeteksi objek-objek yang melintasi perlintasan sebidang. Hasil pelatihan model ini menunjukkan bahwa model memiliki kinerja yang baik untuk digunakan dalam deteksi objek palang pintu dan objek kendaraan di lingkungan perlintasan sebidang.

B. PENGEMBANGAN SISTEM

Alur pengembangan sistem deteksi dan pengenalan objek pelanggaran di perlintasan sebidang pada penelitian ini ditunjukkan oleh *sequence diagram* pada Gambar 2. *Sequence diagram* menjelaskan proses untuk mendeteksi objek palang pintu perlintasan sebidang. Jika palang pintu perlintasan terdeteksi sedang tertutup, *node* 1 akan mengirimkan pesan status objek palang pintu ke *node* 2. *Node* 2 menerima pesan tersebut dan melakukan akses ke perangkat akhir untuk menerima data video. Selanjutnya, pada *node* 2, data video digunakan untuk melakukan deteksi objek terhadap setiap pelanggaran yang terjadi. Hasil deteksi tersebut dicatat dan dikirimkan ke *data logger* atau layanan awan selama pesan status objek palang pintu dari *node* 1 masih menunjukkan palang tertutup. Namun, jika *node* 1 mengirimkan pesan status objek palang pintu terbuka, proses pada *node* 2 akan berhenti. Proses komputasi pada sistem ini menggunakan protokol komunikasi *message queuing telemetry transport* (MQTT) sebagai media komunikasi data antar lapisan, MQTT menggunakan model komunikasi *publish* dan *subscribe* yang memungkinkan perangkat berlangganan pada topik tertentu dan menerima pesan yang diterbitkan pada topik tersebut, memfasilitasi komunikasi terdistribusi, dan efektif untuk



Gambar 3. Alur proses *computation offloading*.

mengirimkan media, seperti gambar atau video, kepada perangkat yang berlangganan [30].

1) LAPISAN TEPI *NODE* 1

Lapisan tepi *node* 1 menggunakan Raspberry Pi 4 untuk menerima *real-time streaming protocol* (RTSP) dari data video di perlintasan sebidang seperti pada Gambar 3. Proses komputasi pada lapisan ini melibatkan model deteksi objek palang pintu. Algoritma sistem mendeteksi objek palang pintu, mengenali pergerakan objek, dan melacaknya dengan *bounding box*. Ketika palang pintu bergerak menutup perlintasan, sistem melakukan pembaruan koordinat dari *centroid point*. Jika *centroid point* melewati garis linier yang ditentukan, sistem menandai situasi perlintasan sebidang tertutup. Setelah situasi terdeteksi, perangkat tepi *node* 1 berkomunikasi dengan *node* 2 untuk melanjutkan proses komputasi.

Dalam proses komunikasi, perangkat tepi *node* 1 berperan sebagai klien *message queuing telemetry transport* (MQTT) (*publisher*) melalui jaringan lokal dan melakukan *offloading* ke perangkat tepi *node* 2. Proses dimulai dengan koneksi ke *broker* melalui alamat IP dan *port*. Jika terkoneksi, sistem membaca *frame* dari data video. Jika palang pintu terdeteksi tertutup, setiap *frame* dikonversi ke tipe data *byte* menggunakan library OpenCV *imencode*, lalu di-*publish* ke *broker* dengan topik dan data yang di-*encode*. Proses ini berulang selama koneksi ke *broker* tetap terhubung.

2) LAPISAN TEPI *NODE* 2

Pada lapisan tepi *node* 2, digunakan NVIDIA Jetson Nano sebagai perangkat tepi [31]. Prosesnya diperlihatkan pada Gambar 3. Perangkat tepi *node* 2 menerima pesan dari *edge node* 1 yang menunjukkan bahwa palang pintu perlintasan sedang tertutup, mengindikasikan bahwa kereta api akan melintas. Saat palang pintu terbuka, status perangkat tepi *node* 2 menjadi *idle*, sehingga proses komputasi pada lapisan ini hanya aktif saat menerima pesan dari perangkat tepi *node* 1. Tugasnya melibatkan multideteksi objek kendaraan dan pelacakan pergerakan kendaraan. Area pelanggaran ditentukan oleh empat titik koordinat yang membentuk *polygon* dan pelanggaran terdeteksi jika *centroid point* kendaraan berada di dalam area tersebut. Sistem mencatat setiap pelanggaran dan mengirim data dalam format JSON ke lapisan awan melalui Rest API.

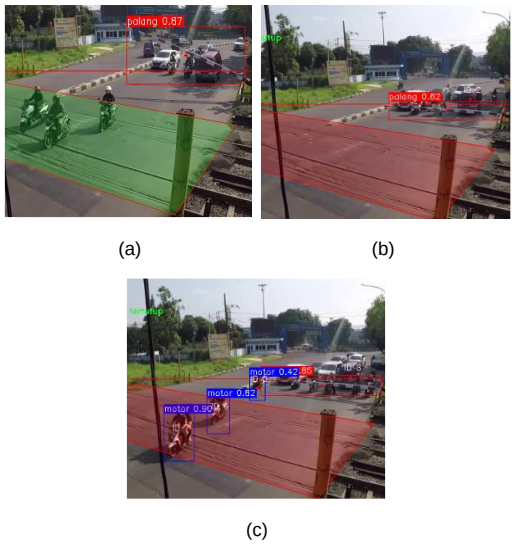
Proses komunikasi pada perangkat tepi *node* 2 dimulai dengan melakukan koneksi ke *broker*. Setelah berhasil terhubung, perangkat tepi dari *node* 2 menerima data yang telah dikirim oleh *publisher*, lalu data yang diterima dikonversi kembali ke bentuk data semula. Selanjutnya, dilakukan proses *decoding* untuk menampilkan gambar. Data yang diterima dalam bentuk *byte* dikonversi dengan fungsi *np.frombuffer* menjadi *ndarray* dan selanjutnya dikonversi kembali ke data semula menggunakan fungsi *cv2.imdecode* untuk melakukan *decode data* menjadi gambar. Terhadap data gambar yang diterima dilakukan proses deteksi objek untuk mengidentifikasi pelanggaran. Hasil deteksi objek (data pelanggaran) disimpan dalam format *array* JSON dan siap untuk dikirimkan ke lapisan awan melalui *application programming interface* (API). Alur komunikasi ini memungkinkan perangkat tepi *node* 2 untuk berkomunikasi dengan *broker* dan mengirimkan data pelanggaran yang telah terdeteksi ke lapisan awan melalui API.

3) LAPISAN LAYANAN AWAN

Platform yang digunakan pada lapisan layanan awan adalah platform awan Heroku. Tahapan proses pada lapisan layanan awan ditunjukkan pada Gambar 3. Tugas lapisan layanan awan adalah menerima setiap data pelanggaran dalam bentuk API yang dicatat oleh sistem pada *data logger*. Dalam rentang waktu tertentu, seluruh data pelanggaran di *logger* akan dikirim dan disimpan di platform penyimpanan awan. Data-data pelanggaran selanjutnya dapat dianalisis untuk keperluan pengawasan tingkat pelanggaran di perlintasan sebidang. Informasi ini akan digunakan untuk pengambilan keputusan terkait dengan proses pengawasan dan pengendalian potensi kecelakaan serta pelanggaran lalu lintas di perlintasan sebidang. Selain itu, data tersebut juga digunakan untuk perencanaan pembangunan perlintasan tidak sebidang [32].

C. IMPLEMENTASI SISTEM

Implementasi penelitian ini menggunakan beberapa perangkat yang mendukung penerapan teknologi komputasi tepi. *Node* tepi 1 menggunakan perangkat Raspberry Pi 4 dengan spesifikasi CPU 64-bit Quad-Core Cortex-A72 1,5 GHz dan memori 4 GB, sedangkan *node* tepi 2 menggunakan perangkat NVIDIA Jetson Nano dengan spesifikasi prosesor Quad-Core ARM Cortex A57 dan memori 4 GB 128-core Maxwell. Sementara itu, server awan yang digunakan adalah platform awan Heroku. Implementasi pengembangan sistem pada penelitian ini dilakukan dengan menggunakan data video situasi kondisi perlintasan di JPL 156 km 152 Stasiun Andir Kota Bandung. Proses pengambilan data video ini menggunakan kamera USB dengan resolusi 1.028 × 780 yang ditempatkan di sudut samping perlintasan sebidang pada sebuah tripod dengan ketinggian sekitar 3 m agar kamera dapat merekam seluruh area perlintasan sebidang.



(a) (b)

(c)

Gambar 4. Implementasi sistem, (a) kondisi palang pintu terbuka, (b) kondisi palang pintu tertutup, (c) kondisi ketika sistem mendeteksi pelanggaran.

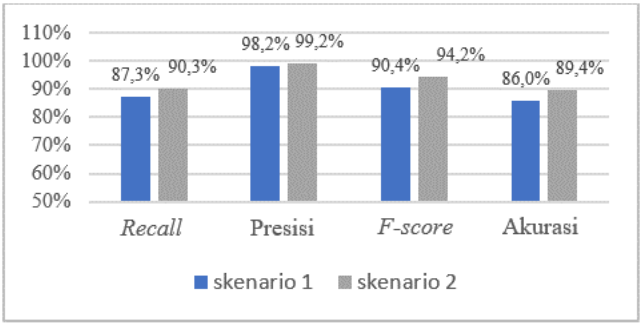
Gambar 4(a) memperlihatkan situasi ketika palang pintu perlintasan sebidang masih dalam kondisi terbuka. Sistem berhasil mendeteksi objek palang pintu perlintasan, yang ditandai dengan *bounding box* dan posisi *centroid point* serta area pelanggaran masih berwarna hijau. Warna hijau ini menandakan bahwa kendaraan yang melintas tidak terdeteksi melakukan pelanggaran. Selanjutnya, pada Gambar 4(b), palang pintu perlintasan sebidang tertutup. Sistem melakukan deteksi pergerakan objek palang pintu melalui *centroid point*. Jika *centroid point* di bawah garis, status palang pintu menjadi tertutup serta area pelanggaran berubah menjadi warna merah, yang artinya setiap objek kendaraan yang melintas akan terdeteksi melakukan pelanggaran. Terakhir, Gambar 4(c) memperlihatkan situasi ketika terjadi pelanggaran. Sistem berhasil mendeteksi dan mengindentifikasi objek pelanggaran pada area pelanggaran pada saat palang pintu perlintasan sebidang tertutup.

III. HASIL DAN PEMBAHASAN

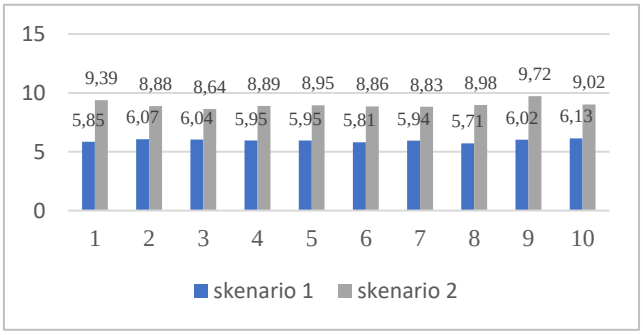
Pengujian dalam penelitian ini dilakukan dengan dua skenario. Tujuannya adalah untuk melakukan perbandingan kinerja komputasi pada setiap skenario, sehingga hasil dari pengembangan sistem dapat diketahui, dapat meningkatkan kinerja komputasi pada perangkat komputasi terbatas dalam aspek akurasi, kecepatan komputasi, dan kinerja komputasi atau tidak [33]. Skenario 1 merujuk pada rancangan sistem yang telah dilakukan pada penelitian sebelumnya [20], sedangkan skenario 2 menggunakan pengembangan sistem pada penelitian ini, dengan menerapkan metode *computation offloading*.

A. HASIL PENGUJIAN AKURASI

Pengujian akurasi dilakukan menggunakan data video situasi perlintasan sebidang. Jumlah data video yang digunakan pada pengujian adalah sepuluh data video, dengan variasi waktu dan kondisi cuaca yang berbeda-beda. Dari dua skenario pengujian akurasi, hasil *confusion matrix* digunakan untuk mengukur tingkat akurasi model deteksi pelanggaran di perlintasan sebidang. *True positive* (TP) terjadi ketika sistem berhasil mendeteksi objek yang melakukan pelanggaran dan sistem berhasil mencatatnya sebagai pelanggaran. *False positive* (FP) terjadi ketika sistem keliru mendeteksi objek sebagai pelanggaran, sedangkan objek tersebut sebenarnya



Gambar 5. Grafik perbandingan akurasi model deteksi pelanggaran.



Gambar 6. Grafik perbandingan rata-rata kecepatan komputasi.

tidak melakukan pelanggaran. Sementara itu, *false negative* (FN) adalah kondisi ketika objek yang seharusnya terdeteksi sebagai pelanggaran tidak tercatat sebagai pelanggaran oleh sistem.

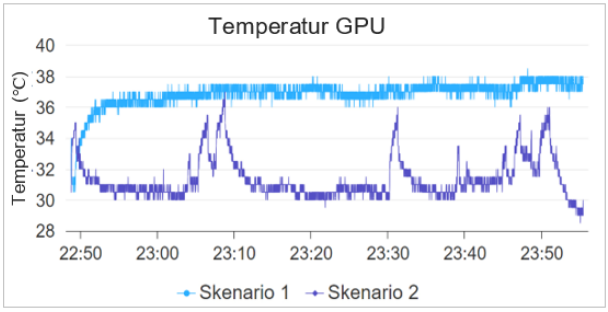
Gambar 5 memperlihatkan grafik perbandingan nilai akurasi model deteksi pelanggaran. Terlihat bahwa skenario 2 memperoleh hasil yang lebih baik daripada skenario 1. Nilai *recall* pada skenario 2 lebih baik 3%, nilai presisi pada skenario 2 lebih baik 1%, nilai *F-score* skenario 2 lebih baik 2,8 %, dan tingkat akurasi pengujian skenario 2 lebih baik sekitar 3,40%. Model deteksi objek pelanggaran yang diimplementasikan mencapai tingkat akurasi yang tinggi, yaitu sebesar 89,4%. Hal ini mengindikasikan bahwa sistem deteksi berhasil mengenali dan mengklasifikasikan objek pelanggaran dengan baik serta berhasil mencatatnya sebagai pelanggaran.

B. HASIL PENGUJIAN KECEPATAN INFERENSI

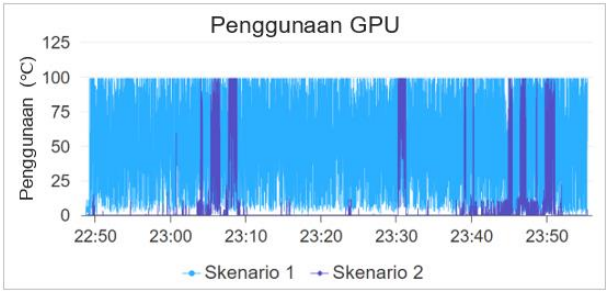
Kecepatan inferensi merujuk pada kecepatan sebuah sistem atau model untuk menghasilkan hasil prediksi atau keluaran setelah menerima masukan. Pengujian kecepatan inferensi dengan sepuluh kali pengujian menggunakan sepuluh data video menghasilkan rata-rata kecepatan sebesar 5,95 fps pada skenario 1 dan 8,95 fps pada skenario 2. Berdasarkan data tersebut, perbandingan hasil pengujian digambarkan pada Gambar 6. Grafik hasil perbandingan rata-rata kecepatan inferensi memperlihatkan bahwa skenario 2 menghasilkan rata-rata kecepatan inferensi lebih tinggi dibandingkan dengan skenario 1.

C. HASIL PENGUJIAN KINERJA KOMPUTASI

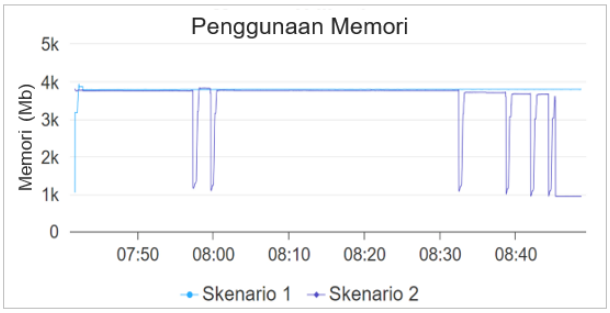
Kinerja komputasi mengacu pada kemampuan perangkat keras dalam melaksanakan tugas-tugas yang diberikan dengan efisiensi dan efektivitas yang tinggi [34]. Gambar 7 memperlihatkan kinerja *graphics processing unit* (GPU) berupa temperatur. Terlihat bahwa proses komputasi pada saat proses deteksi pelanggaran dijalankan pada perangkat NVIDIA Jetson Nano memengaruhi tingkat temperatur GPU pada perangkat. Dari grafik tersebut, terlihat bahwa GPU pada skenario 2 mempunyai nilai temperatur yang lebih rendah



Gambar 7. Grafik hasil perbandingan temperatur GPU.



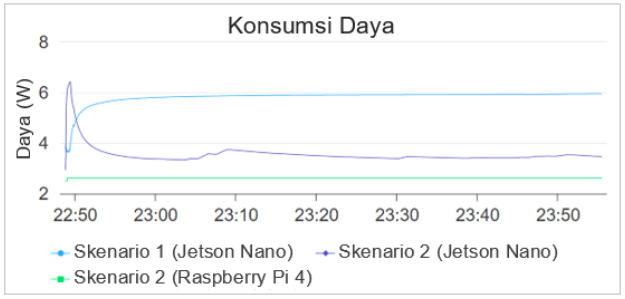
Gambar 8. Grafik hasil perbandingan penggunaan GPU.



Gambar 9. Grafik hasil perbandingan penggunaan memori.

dibandingkan dengan temperatur GPU pada skenario 1. Hal ini terjadi karena proses komputasi dilakukan hanya pada saat pintu perlintasan tertutup. Rata-rata temperatur GPU adalah 36,85 °C pada skenario 1 dan 31,34 °C pada skenario 2 atau terjadi penurunan rata-rata 5,50 °C. Gambar 8 memperlihatkan grafik hasil pengujian dari penggunaan GPU. Diperoleh nilai rata-rata penggunaan GPU sebesar 50,58% pada skenario 1 dan 6,53% pada skenario 2 atau ada penurunan sekitar 44,05%. Proses komputasi pada skenario 2 lebih efektif karena penggunaan GPU hanya terjadi pada kondisi perlintasan sebidang tertutup. Pada Gambar 9 ditampilkan data penggunaan memori. Hasil pengujian penggunaan memori menunjukkan bahwa pada skenario 1 digunakan memori rata-rata 3.790 Mb dan pada skenario 2 digunakan memori rata-rata 3.489 Mb. Dengan kata lain, terjadi penurunan penggunaan memori sekitar 301 Mb. Sementara itu, grafik perbandingan konsumsi daya diperlihatkan pada Gambar 10. Rata-rata konsumsi daya pada skenario 1 dengan perangkat Jetson Nano adalah 5,83 W, sedangkan pada skenario 2 dengan perangkat Jetson Nano adalah 3,55 W. Terjadi penurunan rata-rata konsumsi daya sebesar 2,28 W pada perangkat Jetson Nano. Pada skenario 2 dengan perangkat Raspberry Pi 4 yang diterapkan sebagai *node* 1, rata-rata konsumsi dayanya sebesar 2,61 W, lebih kecil daripada pada Jetson Nano. Hal ini terjadi karena proses komputasi pada perangkat Raspberry Pi 4 hanya menggunakan CPU, tidak menggunakan GPU.

Hasil pengujian pada Tabel II menunjukkan bahwa penerapan *computation offloading* pada prototipe sistem



Gambar 10. Grafik hasil perbandingan konsumsi daya.

TABEL II
PERBANDINGAN HASIL PENGUJIAN

Pengujian	Skenario 1	Skenario 2	Delta
Akurasi	86,0%	89,4%	+3,4%
Kecepatan inferensi	5,95 fps	8,95 fps	+3 fps
Temperatur GPU	36,85 °C	31,34 °C	-5,50 °C
Penggunaan GPU	50,58 %	6,53 %	-44,05 %
Penggunaan memori	3.790 Mb	3.489 Mb	-301 Mb
Konsumsi daya	5,83 W	3,55 W	-2,28 W

Ket.: Delta = skenario 2 – skenario 1

deteksi pelanggaran di perlintasan sebidang berbasis komputasi tepi memberikan hasil yang lebih baik, tampak dari adanya peningkatan kinerja komputasi. Penerapan *computation offloading* berhasil meningkatkan kinerja komputasi pada perangkat komputasi terbatas Jetson Nano. Rata-rata kecepatan komputasi meningkat sekitar 1,5 kali, yang menunjukkan efisiensi strategi pembagian beban komputasi. Akurasi model deteksi objek pelanggaran yang diimplementasikan mencapai 89,4%, yang tergolong tinggi. Hal ini mengindikasikan bahwa sistem deteksi berhasil mengenali dan mengklasifikasikan objek pelanggaran dengan baik. Selain meningkatkan kecepatan, penerapan *computation offloading* berhasil mengurangi beban proses komputasi pada perangkat Jetson Nano. Penurunan temperatur GPU, tingkat penggunaan GPU, tingkat penggunaan memori, dan konsumsi daya memberikan gambaran efisiensi penggunaan sumber daya. Sistem yang dikembangkan pada penelitian ini membuktikan efektivitas dan efisiensi penerapan *computation offloading* dalam mengoptimalkan kinerja sistem deteksi pelanggaran di perlintasan sebidang pada perangkat komputasi terbatas.

IV. KESIMPULAN

Prototipe sistem deteksi pelanggaran di perlintasan sebidang berbasis komputasi tepi dengan menerapkan *computation offloading* yang dikembangkan pada penelitian ini telah berhasil meningkatkan kinerja komputasi dengan membagi beban komputasi. Dari pengembangan yang dilakukan, penerapan *computation offloading* berhasil meningkatkan rata-rata kecepatan komputasi sekitar 1,5 kali lebih cepat pada perangkat komputasi terbatas Jetson Nano. Selain itu, model deteksi objek pelanggaran juga memiliki tingkat akurasi mencapai 89,4%. Selain meningkatkan kecepatan, penerapan *computation offloading* juga berhasil mengurangi beban proses komputasi pada perangkat komputasi terbatas Jetson Nano. Hal ini terlihat dari penurunan temperatur GPU sekitar 5,50 °C, penurunan penggunaan GPU sebesar 44,05%, penurunan penggunaan memori sebesar 301 Mb, dan penurunan konsumsi daya sebesar 2,28 W. Data menunjukkan bahwa penerapan *computation offloading* pada sistem deteksi pelanggaran perlintasan sebidang berbasis komputasi tepi memberikan hasil yang baik. Peningkatan akurasi dan kecepatan inferensi yang signifikan serta pengurangan

temperatur GPU, penggunaan GPU, penggunaan memori, dan konsumsi daya menunjukkan efisiensi penggunaan sumber daya pada perangkat komputasi terbatas.

KONFLIK KEPENTINGAN

Penulis menyatakan bahwa pada penelitian yang ditulis dalam artikel berjudul "Penerapan *Computation Offloading* Pada Sistem Deteksi Pelanggaran Perlintasan Sebidang Berbasis Komputasi Tepi" tidak terdapat konflik kepentingan.

KONTRIBUSI PENULIS

Konseptualisasi dan metodologi, Rian Putra Pratama; perangkat lunak, Rian Putra Pratama; akuisisi data, Rian Putra Pratama; analisis data, Rian Putra Pratama; penulisan—penyusunan draf asli, Rian Putra Pratama; supervisi dan penelaahan, Suhono Harso Supangkat.

UCAPAN TERIMA KASIH

Terima kasih disampaikan kepada pihak-pihak yang telah mendukung penulis dalam melakukan kegiatan penelitian ini, di antaranya Pusat Mekatronika Cerdas Badan Riset dan Inovasi Nasional, Sekolah Tinggi Elektronika dan Informatika Institut Teknologi Bandung, dan Program beasiswa dalam negeri Kementerian Komunikasi dan Informatika Republik Indonesia.

REFERENSI

[1] "Lalu Lintas dan Angkutan Jalan," Undang-Undang Republik Indonesia No. 22, 2009.

[2] "Buku Statistik Bidang Perkeretaapian Tahun 2020," Direktorat Jenderal Perkeretaapian, 2020.

[3] A. Sianipar, "Kajian penerapan teknologi pintu dengan pagar otomatis dan yellow box di perlintasan sebidang," *J. Penelit. Transp. Darat.*, vol. 22, no. 1, hal. 91–102, Jun. 2020, doi: 10.25104/jptd.v22i1.1603.

[4] S.H. Supangkat, A.A. Arman, R.A. Nugraha, dan Y.A. Fatimah, "The implementation of Garuda Smart City Framework for smart city readiness mapping in Indonesia," *J. Asia-Pac. Stud.*, vol. 32, hal. 169-176, Mar. 2018, doi: 10.57278/wiapstokyu.32.0_169.

[5] V. Tsakanikas dan T. Dagiuklas, "Video surveillance systems-current status and future trends," *Comput. Elect. Eng.*, vol. 70, hal. 736–753, Agu. 2018, doi: 10.1016/j.compeleceng.2017.11.011.

[6] R.P. Pratama dan S.H. Supangkat, "Smart video surveillance system for level crossing: A systematic literature review," *2021 Int. Conf. ICT Smart Soc. (ICISS)*, 2021, hal. 1–5, doi: 10.1109/ICISS53185.2021.9533222.

[7] M.H. Kolekar, *Intelligent Video Surveillance Systems An Algorithmic Approach*. New York, NY, USA: CRC Press, 2017, doi: 10.1201/9781315153865.

[8] A. Hampapur dkk., "Smart video surveillance: Exploring the concept of multiscale spatiotemporal tracking," *IEEE Signal Process. Mag.*, vol. 22, no. 2, hal. 38–51, Mar. 2005, doi: 10.1109/MSP.2005.1406476.

[9] G.F. Shidik dkk., "A systematic review of intelligence video surveillance: Trends, techniques, frameworks, and datasets," *IEEE Access*, vol. 7, hal. 170457–170473, Nov. 2019, doi: 10.1109/ACCESS.2019.2955387.

[10] H. Sun, Y. Yu, K. Sha, dan B. Lou, "mVideo: Edge computing based mobile video processing systems," *IEEE Access*, vol. 8, hal. 11615–11623, Des. 2020, doi: 10.1109/ACCESS.2019.2963159.

[11] W. Yu dkk., "A survey on the edge computing for the internet of things," *IEEE Access*, vol. 6, hal. 6900–6919, Nov. 2018, doi: 10.1109/ACCESS.2017.2778504.

[12] W. Shi dkk., "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, hal. 637–646, Okt. 2016, doi: 10.1109/JIOT.2016.2579198.

[13] F. Wang dkk., "Deep learning for edge computing applications: A state-of-the-art survey," *IEEE Access*, vol. 8, hal. 58322–58336, Mar. 2020, doi: 10.1109/ACCESS.2020.2982411.

[14] D.R. Patrikar dan M.R. Parate, "Anomaly detection using edge computing in video surveillance system: Review," *Int. J. Multimed. Inf. Retr.*, vol. 11, no. 2, hal. 85–110, Jun. 2022, doi: 10.1007/s13735-022-00227-8.

[15] A.C. Cob-Parro dkk., "Smart video surveillance system based on edge computing," *Sensors*, vol. 21, no. 9, hal. 1-20, Mei 2021, doi: 10.3390/s21092958.

[16] A. Gupta dan P. Prabhat, "Towards a resource efficient and privacy-preserving framework for campus-wide video analytics-based applications," *Complex Intell. Syst.*, vol. 9, no. 1, hal. 161–176, Feb. 2023, doi: 10.1007/s40747-022-00783-w.

[17] M. Aazam, S. Zeadally, dan K.A. Harras, "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities," *Future Gener. Comput. Syst.*, vol. 87, hal. 278–289, Okt. 2018, doi: 10.1016/j.future.2018.04.057.

[18] A. Zaman, B. Ren, dan X. Liu, "Artificial intelligence-aided automated detection of railroad trespassing," *Transp. Res. Rec., J. Transp. Res. Board,*, vol. 2673, no. 7, hal. 25–37, Jul. 2019, doi: 10.1177/0361198119846468.

[19] M.A.B. Fayyaz dan C. Johnson, "Object detection at level crossing using deep learning," *Micromachines*, vol. 11, no. 12, hal. 1-16, Des. 2020, doi: 10.3390/mi11121055.

[20] P. Sikora dkk., "Artificial intelligence-based surveillance system for railway crossing traffic," *IEEE Sens. J.*, vol. 21, no. 14, hal. 15515–15526, Jul. 2021, doi: 10.1109/jsen.2020.3031861.

[21] C. Sun dkk., "MCA-YOLOV5-light: A faster, stronger and lighter algorithm for helmet-wearing detection," *Appl. Sci.*, vol. 12, no. 19, hal. 1-19, Okt. 2022, doi: 10.3390/app12199697.

[22] X. Xu, X. Zhang, dan T. Zhang, "Lite-YOLOv5: A lightweight deep learning detector for on-board ship detection in large-scene Sentinel-1 SAR images," *Remote Sens.*, vol. 14, no. 4, hal. 1-27, Feb. 2022, doi: 10.3390/rs14041018.

[23] M. Ali dkk., "RES: Real-time video stream analytics using edge enhanced clouds," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, hal. 792–804, Apr.-Jun. 2022, doi: 10.1109/TCC.2020.2991748.

[24] X. Xia dkk., "Cost-effective app data distribution in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, hal. 31–44, Jan. 2021, doi: 10.1109/TPDS.2020.3010521.

[25] H. Alawad, S. Kaewunruen, dan M. An, "Learning from accidents: Machine learning for safety at railway stations," *IEEE Access*, vol. 8, hal. 633–648, Des. 2020, doi: 10.1109/ACCESS.2019.2962072.

[26] U. Nepal dan H. Eslamiat, "Comparing YOLOv3, YOLOv4 and YOLOv5 for autonomous landing spot detection in faulty UAVs," *Sensors*, vol. 22, no. 2, hal. 1-15, Jan. 2022, doi: 10.3390/s22020464.

[27] P. Sikora, M. Kiac, dan M.K. Dutta, "Classification of railway level crossing barrier and light signalling system using YOLOv3," *2020 43rd Int. Conf. Telecommun. Signal Process. (TSP)*, 2020, hal. 528–532, doi: 10.1109/TSP49548.2020.9163535.

[28] J. Redmon, S. Divvala, R. Girshick, dan A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, hal. 779–788, doi: 10.1109/CVPR.2016.91.

[29] J. Solawetz (2020) "What is Mean Average Precision (mAP) in Object Detection?," [Online], https://blog.roboflow.com/mean-average-precision/, tanggal akses: 15-Jul-2023.

[30] M. Veeramanikandan dan S. Sankaranarayanan, "Publish/subscribe based multi-tier edge computational model in internet of things for latency reduction," *J. Parallel Distrib. Comput.*, vol. 127, hal. 18–27, Mei 2019, doi: 10.1016/j.jpdc.2019.01.004.

[31] D.J. Shin dan J.J. Kim, "A deep learning framework performance evaluation to use YOLO in Nvidia Jetson platform," *Appl. Sci.*, vol. 12, no. 8, hal. 1-19, Apr. 2022, doi: 10.3390/app12083734.

[32] I. Resmadi, "Kajian moralitas teknologi pintu perlintasan kereta api (Studi kasus: Pintu perlintasan kereta api Cikudapateuh Bandung)," *J. Sosioteknol.*, vol. 13, no. 2, hal. 84–90, Agu. 2014, doi: 10.5614/sostek.itbj.2014.13.2.2.

[33] S. Valladares dkk., "Performance evaluation of the Nvidia Jetson Nano through a real-time machine learning application," *Int. Conf. Intell. Hum. Syst. Integr.*, 2021, hal. 343–349, doi: 10.1007/978-3-030-68017-6_51.

[34] A. Al-Qamash, I. Soliman, R. Abulibdeh, dan M. Saleh, "Cloud, fog, and edge computing: A software engineering perspective," *2018 Int. Conf. Comput. Appl. (ICCA)*, 2018, hal. 276–284, doi: 10.1109/COMAPP.2018.8460443.