Terakreditasi SINTA Peringkat 4
Surat Keputusan Direktur Jenderal Pendidikan Tinggi, Riset, dan Teknologi, Nomor: 158/E/KPT/2021
masa berlaku mulai Volume 5 Nomor 2 Tahun 2018 sampai Volume 10 Nomor 1 Tahun 2023

Terbit online pada laman web jurnal: **https://jurnal.plb.ac.id/index.php/tematik/index**

TEMATIK

Jurnal **Teknologi Informasi Komunikasi (e-Journal)**

Vol. 9 No. 2 (2022) 119-127 ISSN Media Elektronik: 2443-3640

Pendeteksian Pelanggaran Pada Penyebrangan Jalan Menggunakan *Single-Shot Detector* Pada ESP32

Pedestrian Crossing Violation Detection Using Single-Shot Detector on ESP32

Fahri Novaldi[1], Iqbal Amrulloh[1], I Wayan Wiprayoga Wisesa[1,2], Martin Clinton Tosima Manullang[1,2,*]
[1] Program Studi Teknik Informatika, Institut Teknologi Sumatera, Lampung Selatan, Indonesia
[2] Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan

Abstract

The high number of motorized vehicles and their growth in big cities, and the high rate of violations make it challenging to identify violations of motorized vehicle drivers, especially in the case of drivers who stop at road crossing markings (zebra cross). The utilization of computer vision technology is expected to help identify violations by recognizing objects in the form of motorized vehicles contained in a visual area captured by the camera. Using a single-shot detector method from a model trained and implemented on an embedded device with ESP32, a violation detection system is generated. The system developed is not only in the form of hardware but also interface software that can be used to configure and determine the region of interest. Two kinds of tests were carried out, four testing in real time scenario and 20 offline testing using the Pedestrian Traffic Light dataset. All of them can be detected by the system from all real-time experiments. Meanwhile, an offline experiment using a dataset from the Pedestrian Traffic Light Dataset resulted in an accuracy of 96.78%.

Keywords: zebra-cross, object detection, ESP32, single-shot detector

Abstrak

Tingginya jumlah kendaraan bermotor dan pertumbuhannya di kota-kota besar, serta tingginya angka pelanggaran membuat identifikasi pelanggaran terhadap pengendara kendaraan bermotor menjadi sulit, terutama dalam hal pengendara yang berhenti di marka persimpangan jalan *(zebra cross)*. Pemanfaatan teknologi *computer vision* diharapkan dapat membantu mengidentifikasi pelanggaran dengan mengenali objek berupa kendaraan bermotor yang terdapat pada area visual yang tertangkap kamera. Sistem menggunakan metode pendeteksi *single-shot detector* dari model yang dilatih dan diimplementasikan pada perangkat keras ESP32. Sistem yang dikembangkan tidak hanya berupa perangkat keras tetapi juga perangkat lunak antarmuka yang dapat digunakan untuk mengkonfigurasi dan menentukan *region* yang diinginkan. Dua macam pengujian dilakukan, empat pengujian dalam skenario *real-time* dan 20 pengujian secara *offline* menggunakan *dataset Pedestrian Traffic Light*. Seluruh keadaan pada skenario *real-time* dapat dideteksi dengan tepat. Sementara itu, eksperimen *offline* menggunakan dataset dari *Dataset Pedestrian Traffic Light* menghasilkan akurasi 96,78%.

Kata kunci: penyebrangan jalan, deteksi objek, ESP32, single-shot detector.

1. Pendahuluan

Laju kenaikan jumlah populasi dan lapangan pekerjaan di kota mendukung tingginya tingkat perpindahan individu di dalam kota. Jika dibandngkan dengan laju perkembangan infrastruktur jalan di kota, laju pertambahan kendaraan telah meningkat secara eksponensial [1]. Hal ini mengakibatkan tingginya tingkat kepadatan lalu lintas dalam kota. Tingginya tingkat kepadatan lalu lintas dapat mengakibatkan mobilitas terhambat, polusi udara, dan bahkan dapat juga meningkatkan kemungkinan terjadinya kecelakaan dalam berkendara. Menurut laporan WHO [2], diperkirakan sekitar 1.3 juta nyawa telah hilang setiap tahunnya dalam kecelakaan lalu lintas. Lebih dari setengahnya, terjadi pada pejalan kaki, pesepeda, dan juga pengendara sepeda motor. Berdasarkan data ini, pemerintah setiap negara perlu mengambil tindakan secara efektif dalam mengurangi tingkat kecelakaan lalu lintas. Salah satu caranya dengan merancang infrastruktur jalan yang dapat menjamin keamanan pengguna jalan.

Diterima Redaksi: 08-08-2022 | Selesai Revisi: 06-10-2022 | Diterbitkan Online: 01-12-2022

Perkembangan teknologi terkini memungkinkan pengawasan secara jarak jauh dengan memanfaatkan teknologi *computer vision*. Pemantauan kondisi lalu lintas dapat dengan mudah dilakukan secara jarak jauh dan dalam waktu nyata *(real-time)* dengan memanfaatkan kamera *CCTV* yang dipasang pada sisi jalan raya. Data yang dihasilkan dari pemantauan secara waktu nyata mengakibatkan tingginya volume data yang harus diproses. Sumber daya komputasi mutakhir dapat dimanfaatkan dalam membantu pemrosesan data ini. Perpaduan antara teknik *computer vision* dan pembelajaran mesin telah meningkatkan efektifitas dalam pemantauan kondisi lalu lintas seperti yang ada pada [3] dan [4].

Mengingat pentingnya dalam mengurangi tingkat kecelakan dalam berlalu lintas, kami menyadari bahwa diperlukan sebuah sistem pemantauan lalu lintas secara cerdas dan efektif. Dalam penelitian ini, kami membangun sebuah perangkat cerdas pendeteksi pelanggaran lalu lintas khususnya pada penyebrangan jalan raya. Berdasarkan data pengamatan dari kamera pemantau, kami memanfaatkan implementasi deep learning untuk mendeteksi terjadinya pelanggaran dalam penyebrangan jalan raya dengan mengimplementasikannya pada perangkat berbiaya terjangkau berupa perangkat prototipe ESP32.

Sistem pengawasan lalu lintas telah beralih dari pengamatan secara manual menuju sistem pengawasan secara cerdas dan otomatis. Pengawasan lalu lintas secara otomatis erat kaitannya dengan deteksi objek dalam data citra maupun video. Dalam hal ini, objek yang dideteksi dapat berupa kendaraan, pejalan kaki, bahkan hingga rambu lalu lintas. Ho et.al [4] mengembangkan sistem pendeteksi parkir kendaraan di tepi jalan raya dengan menerapkan arsitektur *Internet-of-Things (IoT)*. Menggunakan kamera resolusi tinggi dan jaringan nirkabel, data kejadian parkir kendaraan dapat dianalisis secara otomatis dengan bantuan logika *fuzzy*. Hasil analisis ini dapat diberikan pada otoritas terkait dalam mengembangkan kebijakan manajemen parkir di tepi jalan kota Hong Kong.

Selain menggunakan kamera statis yang dipasang pada sisi jalan raya, penggunaan pesawat tanpa awak juga dapat dimanfaatkan dalam pemantauan kondisi lalu lintas dari udara sehingga dapat melakukan pengawasan dalam area yang lebih luas [5]. Liu et.al [6] menggunakan arsitektur komputasi tepi *(edge computing)* untuk menstabilkan komunikasi data dalam jaringan pada pusat manajemen lalu lintas *(traffic management center)*. Penggunaan perangkat yang tersebar pada jalan raya yang disebut sebagai *cloudlet* dapat secara dini melakukan pemrosesan data untuk menghasilkan informasi yang lebih ringkas untuk nantinya dapat diteruskan ke pusat manajemen lalu lintas.

Pemanfaatan inferensi menggunakan deep learning model dalam kasus pendeteksian pelanggaran pengendara roda dua (sepeda motor dan sepeda juga dimanfaatkan oleh Arshad, et.al [7]. Dalam kasus ini, peneliti tersebut mengembangkan model deteksi dengan memanfaatkan Faster RCNN dan RetinaNet untuk mendeteksi pelanggaran tidak menggunakan helm.

Untuk kasus pendeteksian objek bergerak di jalan raya, pengembangan dapat dilakukan dengan menggunakan beberapa model seperti Faster RCNN [8], Retina Net [9], *Single Shot Detector* (SSD) [10], maupun YOLO [11] yang memiliki beberapa versi. Penelitian yang dilakukan oleh Gupta [12] memperbandingkan kehandalan dari beberapa arsitektur pendeteksian di atas. Kesimpulannya, YOLO (dalam hal ini versi 4) menunjukkan performa yang paling baik. Namun, dalam hal kecepatan inferensi, SSD jauh lebih unggul dibandingkan arsitektur lainnya. Kecepatan inferensi tersebut merupakan aspek yang sangat penting jika sistem yang dikembangkan berjalan secara *real-time* lebih lagi inferensi dilakukan di perangkat dengan komputasi sederhana.

Berdasarkan rujukan diatas dan mengamati dari trend penelitian serupa, belum ditemukan adanya implementasi antara sistem pendeteksi pelanggaran berbasis *computer vision* yang diintegrasikan pada perangkat berdaya rendah. Umumnya penelitian serupa melakukan inferensi pada komputer personal yang memiliki kemampuan komputasi tinggi. Sehingga tujuan dari penelitian ini adalah untuk mengembangkan sebuah perangkat terintegrasi berbasis *computer vision* yang diimplementasikan pada perangkat *ESP32-CAM*. Selain itu, penelitian ini mengulas kapabilitas dan kehandalan dari sistem yang dikembangkan apabila inferensi dilakukan pada perangkat *ESP32-CAM*.

Terdapat beberapa kontribusi yang dihasilkan dari penelitian ini. Pertama, penelitian ini akan mengukur kinerja dari inferensi yang dilakukan pada perangkat *ESP32-CAM*. Kedua, penelitian ini akan mengulas bagaimana merancang sebuah antarmuka pengguna yang dapat secara bebas mengatur *region of interest* berbasis *ray-casting* untuk mempermudah pengaturan batas-batas area marka *zebra-cross*. Terakhir, penelitian ini mengulas secara keseluruhan bagaimana mengintegrasikan sebuah sistem *computer-vision*, *internet-of-things*, serta *content delivery network* dalam sebuah perangkat tertanam.

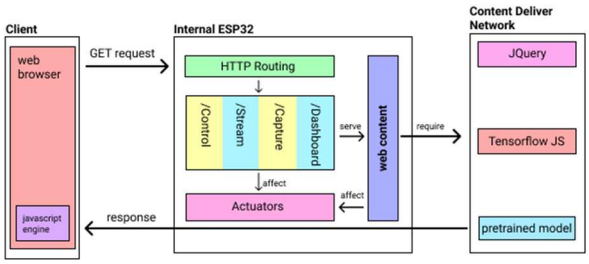
2. Metode Penelitian

2.1. Arsitektur Sistem

Dalam pengembangan sistem deteksi khususnya pada komponen perangkat keras, kami memanfaatkan perangkat berbiaya terjangkau dan rendah daya, yaitu *ESP32-CAM*. Perangkat tersebut merupakan sebuah

DOI 10.38204/tematik.v9i2.997

papan purwarupa yang menggunakan prosesor ESP32 (*Espressif Systems*, Shanghai, China) dengan koneksi *wifi* dan kamera *OV2640* yang terintegrasi. ESP32-CAM ditempatkan pada sisi jalur penyebrangan jalan untuk mengakuisisi citra di sekitar jalur penyebrangan jalan (*zebra-cross*). Sementara itu sistem pengenalan objek pada citra akan dilakukan oleh perangkat *client* dengan memanfaatkan *javascript* dan *framework Tensorflow Lite (TF-Lite)* [13] yang berjalan pada aplikasi peramban. Gambar 1 merangkum arsitektur dari sistem yang diusung.

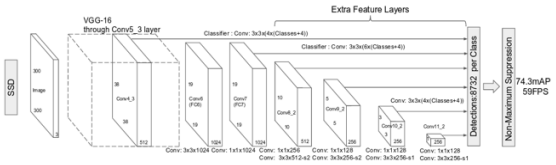


Gambar 1. Arsitektur Sistem

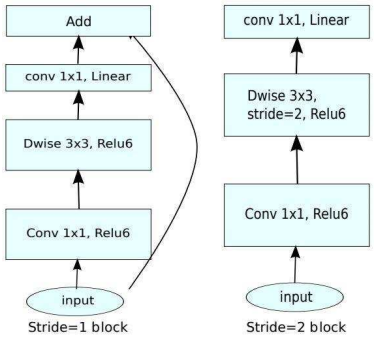
Client akan mengakses sistem melalui *web browser* dengan metode *HTTP request (GET)* yang mana permintaan ini akan ditangani oleh *HTTP routing subservice* yang berjalan pada ESP32 sesuai dengan parameter *URL*-nya. Setelah itu jika *route* valid, maka ESP32 akan melayani *web content* berupa file html. File ini bergantung (*dependent*) kepada beberapa komponen yang berukuran cukup besar untuk ESP32 seperti *JQuery, Tensorflow JS, & pretrained model (COCO SSD)* yang mana komponen ini berada pada jaringan *cloud* atau *content delivery network*. Dependensi ini akan dijalankan secara lokal pada peramban *client* dengan *javascript engine* yang berjalan diatas *web browser* sehingga proses interferensi citra akan menggunakan sumber daya *client* dan bukan pada ESP32.

2.2. Metode Pengenalan Pelanggaran

Sistem yang dikembangkan menggunakan sebuah model bernama *MobileNetV2* [14], yaitu sebuah metode pengenalan citra dengan arsitektur *convolutional neural network* (CNN). Selain daripada itu, sistem yang dikembangkan menggunakan sebuah model yang telah dilatih (*pre-trained model)* bernama COCO-SSD. Model tersebut dikembangkan dengan menggunakan metode *single-shot multi-box detector* (SSD) [10] terhadap sebuah model publik milik Microsoft dan Facebook bernama *Common Object in Context* (COCO) [15].



Gambar 2. Arsitektur CNN *Single-Shot Multi-Box Detection* [10]



Gambar 3. Arsitektur *MobileNetV2* [14]

Algoritma *Non Maximum Supression*
Masukan: Daftar *Bounding Box* yang Diusulkan, *Confidence Score*, Ambang batas *overlap*
Luaran: Daftar *Bounding Box* yang telah difilter

procedure $NMS(B, c)$
  $B_{nms} \leftarrow \emptyset$
  for $b_i \in B$ **do**
    $discard \leftarrow False$
    for $b_j \in B$ **do**
      if $same(b_i, b_j) > \lambda_{nms}$ **then**
        if $score(c, b_j) > score(c, b_i)$ **then**
          $discard \leftarrow True$
    if not $discard$ **then**
      $B_{nms} \leftarrow B_{nms} \cup b_i$
  return B_{nms}

SSD terdiri atas sebuah *backbone model* dan *head*. Dalam sistem yang dikembangkan, *backbone model* yang digunakan berupa jaringan klasifikasi citra yang telah dilatih (*pre-trained image classification network*) untuk ekstraksi fitur dari sebuah citra. Gambar 2 memuat lapisan-lapisan yang ada pada arsitektur SSD dimana terdapat arsitektur VGG16 [16] yang juga merupakan sebuah arsitektur konvolusi yang bergerak terhadap citra. Selanjutnya, terdapat berbagai lapisan konvolusi yang berjenis konvolusi penuh (*fully convolution networks*) dimulai dari ukuran jendela yang besar hingga semakin mengecil di setiap lapisannya. Di bagian akhir lapisan, terdapat sebuah pendeteksi yang terdiri atas 8732 kelas deteksi. Hasil akhir ditentukan berdasarkan metode *non-maximum supression* [17]. Hal tersebut dikarenakan bahwa pada tahapan deteksi, akan terjadi penumpukan area deteksi (*overlap*) sehingga untuk mereduksi beberapa *bounding box* yang mendeteksi objek yang sama, dilakukan perhitungan *non-maximum supression*, yaitu dengan memperbandingkan setiap *bounding box* dan mempertimbangkan nilai interseksinya (*Intersection over Union / IoU).*

DOI 10.38204/tematik.v9i2.997

Dalam implementasinya, arsitektur SSD yang dijelaskan diatas akan sangat bergantung kepada sumber daya pemrosesan yang tinggi. Hal tersebut akan kontradiktif dengan tujuan dari sistem yang dikembangkan, yaitu diimplementasikan kepada sebuah perangkat keras berbiaya terjangkau. Oleh karena itu, sistem yang dikembangkan mewujudkannya dengan arsitektur *MobileNetV2*. Metode ini pada dasarnya adalah sebuah penyesuaian (*fine-tune*) dari metode SSD yang lebih efisien untuk diimplementasikan pada perangkat *mobile*. Pada gambar 3 dapat dilihat bahwa arsitektur ini memiliki dua buah tahapan (*stride*). Pada kedua *stride*, terdapat beberapa lapisan konvolusi dan lapisan konvolusi *deepth-wise* dengan beragam fungsi aktivasi seperti *rectifier linear unit (ReLU)* dan aktivasi linear.

Citra sebagai masukan yang bersumber dari kamera akan diinterpretasikan sebagai matriks tiga dimensi, dimana dua dimensi pertama adalah sebagai nilai intensitas piksel untuk setiap baris dan kolom. Citra yang digunakan berupa citra berwarna (RGB) sehingga terdapat sebuah dimensi lainnya untuk menampung setiap kanal warna.

Selanjutnya, data dalam bentuk matriks tersebut akan diolah dan diinferensikan terhadap model yang telah dilatih untuk mendeteksi setiap objek yang ada pada citra. Terdapat tiga buah informasi luaran dari hasil inferensi ini. Luaran pertama berupa lokasi objek berupa *bounding box* yang terdiri atas koordinat x dan y, serta ukuran panjang dan lebar *bounding box*. Luaran lainnya berupa kelas dari objek yang dideteksi beserta nilai kepercayaan (*confidential score*). Contoh dari luaran tersebut ditampilkan luaran teks sebagai berikut.

```
[{
  bbox: [x, y, width, height],
  class: "person",
  score: 0.8380282521247864
}, {
  bbox: [x, y, width, height],
  class: "kite",
  score: 0.74644153267145157
}]
```

Berdasarkan output yang diperoleh, ukuran objek akan dikonversi menjadi titik koordinat pelanggaran menggunakan Algoritma Simplifikasi koordinat objek.

Setelahnya, koordinat ini akan ditinjau, apakah titik koordinat tersebut berada dalam *polygon* yang terbentuk dari larik koordinat *Region of Interest* (ROI). Terdapat beberapa metode untuk memastikan apakah titik objek berada dalam *polygon ROI*. Pada penelitian ini, dengan mempertimbangkan konteks penelitian dan kompleksitas algoritma, *ray-casting* [18] digunakan sebagai metode penentu apakah titik berada di dalam

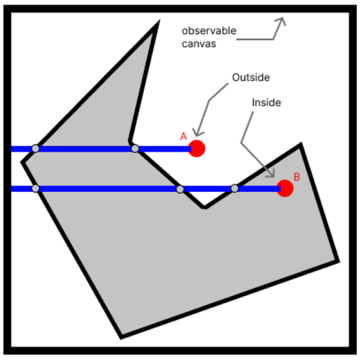
Algoritma Simplifikasi koordinat objek
Masukan: *Bounding box* dengan argumen (koord x, koord y, lebar, tinggi), parameter simplifikasi P
Luaran: Koordinat yang telah disimplifikasi

1 : **Procedure** *simplify*($O_{x,y,width,height}$, *P*)
2 : $\boldsymbol{S_{x,y}} \leftarrow \boldsymbol{O_{x,y}}$
3 : **switch** *P:*
4 : **case** *upper :* $\boldsymbol{S_x} \leftarrow S_x + \frac{O_{width}}{2}$ **end case**
5 : **case** *lower :* $\boldsymbol{S_x} \leftarrow S_x + \frac{O_{width}}{2}$, $\boldsymbol{S_y} \leftarrow S_y + O_{height}$ **end case**
6 : **case** *left :* $\boldsymbol{S_y} \leftarrow S_y + \frac{O_{height}}{2}$ **end case**
7 : **case** *right :* $\boldsymbol{S_x} \leftarrow S_x + O_{width}$, $\boldsymbol{S_y} \leftarrow S_y + \frac{O_{height}}{2}$ **end case**
8 : **case** *center :* $\boldsymbol{S_x} \leftarrow S_x + \frac{O_{width}}{2}$, $\boldsymbol{S_y} \leftarrow S_y + \frac{O_{height}}{2}$ **end case**

Algoritma *Ray-casting*
Masukan: $O_{x,y}$, $Polygon = \{ O_{1\,x,y} , O_{2\,x,y} , \dots , O_{n\,x,y} \}\ n > 2$ (larik koordinat yang membangun *ROI polygon*)
Luaran: Status koordinat

1 : **Function** *isPointInPolygon* :
2 : $isInside \leftarrow false$
3 : **for** $\boldsymbol{i}$ **from 0** $\boldsymbol{to\ n}$ **do**
4 : $j \leftarrow \boldsymbol{i} + \mathbf{1}$
5 : $x_i \leftarrow \boldsymbol{Polygon_{i\,x}}$
6 : $y_i \leftarrow \boldsymbol{Polygon_{i\,y}}$
7 : $x_j \leftarrow \boldsymbol{Polygon_{j\,x}}$
8 : $y_j \leftarrow \boldsymbol{Polygon_{j\,y}}$
9 : **if** $(y_i > O_y) \neq (y_j > O_y)$ **and** $O_x < x_i + \frac{(x_j - x_i) * (O_y - y_i)}{(y_j - y_i)}$ **then** *status_intersect* ← **True**
10 : **end if**
11 : **if** $status\_intersect$ **then**
12 : $isInside \leftarrow !\, isInside$
10 : **end if**
11 : **end for**
12: **return** $isInside$

ROI. Algoritma ini disebut *ray-casting* karena langkah-langkahnya seperti menarik titik vektor ke suatu bidang datar sehingga menyerupai fenomena sinar. Berikut adalah visualisasi sederhana dari algoritma *ray-casting*.



Gambar 4. Visualisasi Algoritma *Ray-Casting*

Seperti pada ilustrasi gambar di atas, titik koordinat A akan menarik sebuah garis ke salah satu bidang datar atau *nearest plane* dari kanvas kemudian menghitung berapa jumlah interseksi yang terjadi pada garis ini dengan bidang *polygon* yang terbentuk dari larik.

DOI 10.38204/tematik.v9i2.997

Karena garis yang dihasilkan dari titik A melakukan interseksi sebanyak genap, maka dapat dipastikan bahwa titik A berada diluar *polygon* begitu juga dengan titik B yang melakukan interseksi sebanyak ganjil dapat dipastikan titik B berada didalam polygon, aturan ini disebut *odd-even rule*. Tahapan-tahapan *Ray-Casting* dapat dilihat pada algoritma *Ray-Casting*.

Selain itu, berikut adalah algoritma deteksi pelanggaran secara keseluruhan:

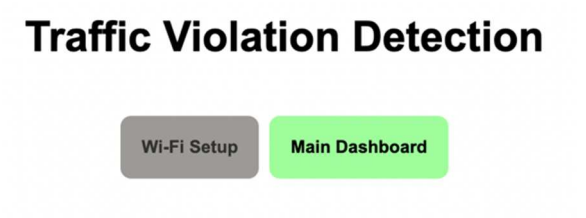
Algoritma Deteksi Pelanggaran
Masukan: p, $Polygon = \{ O_{1\,x,y}\ ,\ O_{2\,x,y}\ , \ldots , O_{n\,x,y}\}\ n > 2$
(larik koordinat yang membentuk *ROI* ditentukan oleh pengguna)
Luaran: Status Pelanggaran

1 : **Procedure** *violationDetection* :
2 : $status\_pelanggaran \leftarrow false$
3 : **for** setiap frame yang ditangkap kamera **do**
4 : $ObjectArray \leftarrow \boldsymbol{SingleShotDetector}(frame)$
5 : **for** setiap **O** pada **ObjectArray do**
6 : $O_{x,y} \leftarrow simplify(O_{x,y,width,height}\ , P)$
7 : **if** $isPointInPolygon(O_{x,y}\ ,\ Polygon)$ **then**
status_pelanggaran ← **True**
9 : **end if**
10 : **end for**
11 : **end for**
12: **return** *status_pelanggaran*

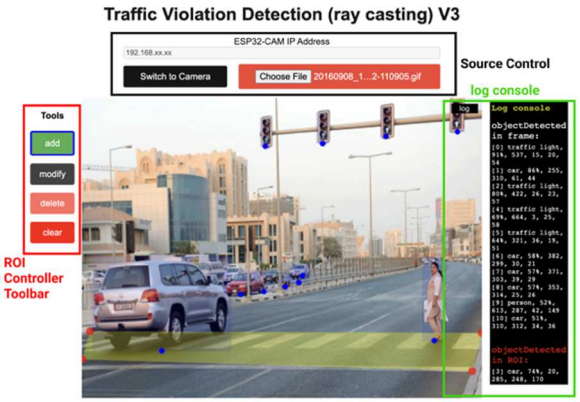
2.3. Antarmuka Pengguna

Implementasi sistem ini diharapkan dapat digunakan oleh pengguna awam, oleh karena itu diperlukan sebuah antarmuka visual yang mudah dioperasikan. Antarmuka ini tidak hanya berfungsi sebagai antarmuka untuk keperluan memonitor, tetapi juga sebagai media pengatur konfigurasi sistem. Antarmuka ini dapat diakses melalui aplikasi peramban yang terhubung pada jaringan yang sama dengan ESP32. Gambar 5 memuat tampilan antarmuka beberapa opsi tombol untuk layanan yang disediakan. Lalu, pada Gambar 6 memuat tampilan dashboard utama dari sistem yang menampilkan *source control* yang berfungsi untuk mengubah sumber gambar, *ROI controller toolbar* yang berfungsi untuk membuat dan memodifikasi area *ROI* pada *canvas*, hingga *log console* yang berfungsi sebagai logger untuk menampilkan objek yang tertangkap pada kamera secara detail. Sementara itu, Gambar 7 memuat tampilan konfigurasi dari sistem berupa objek yang ingin dideteksi, lokasi penanda, hingga ambang batas.

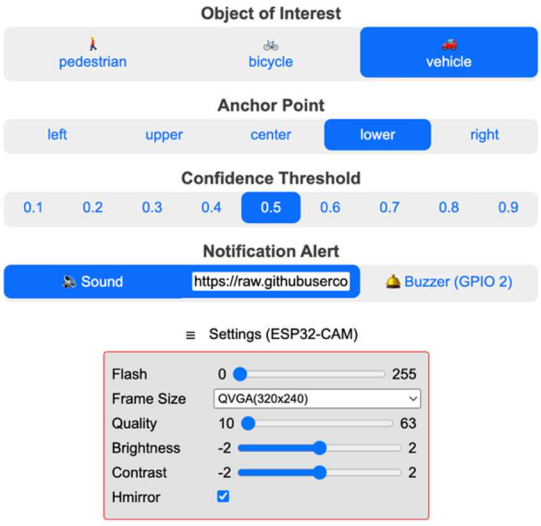
Untuk mempermudah konfigrasi nama *SSID wifi* yang digunakan, maka dibutuhkan antarmuka untuk mengatur konfigurasi *wifi*. Gambar 8 memuat tampilan menu konfigurasi *wifi* untuk memasukkan nama *SSID* dan password dari jaringan wifi.



Gambar 5. Tampilan Antarmuka pada Menu Awal



Gambar 6. Tampilan *Dashboard* Utama



Gambar 7. Tampilan Menu Konfigurasi

Gambar 8. Antarmuka konfigurasi wifi

3. Hasil dan Pembahasan

3.1. Skenario Pengujian

DOI 10.38204/tematik.v9i2.997

Pengujian dilakukan dengan dua skenario yaitu dengan pengujian *non real time* yang menggunakan dataset gambar yang bertujuan untuk menguji kemampuan inferensi dari sistem dan pengujian *real time* yang bertujuan untuk menguji peforma sistem secara keseluruhan.

Pengujian *non real time (offline)* difokuskan pada kemampuan inferensi sistem yaitu hasil observasi dari tingkat keyakinan atau akurasi sistem pendeteksi yang berhasil mendeteksi objek dan mengaktifkan alarm saat ada objek memasuki *ROI* (*region of interest).* Dengan jumlah gambar yang diuji adalah 20 gambar bervariasi. *Dataset* untuk pengujian ini didapatkan dari *dataset* publik bernama *Pedestrian Traffic Light Dataset* (PTL) [19]. Pengujian dilakukan dengan beberapa konfigurasi yaitu:

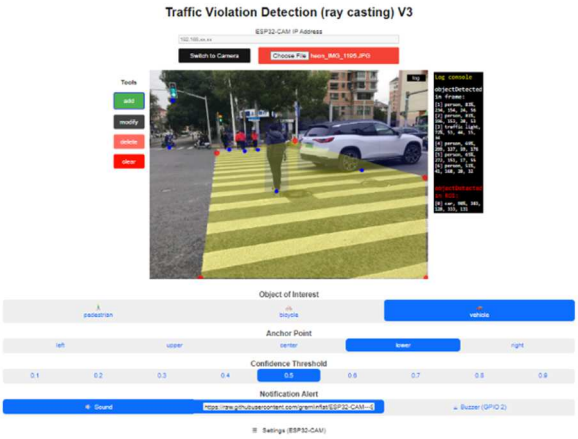
1. Target objek yang terdeteksi dalam *ROI* oleh sistem adalah kendaraan bermotor
2. *ROI* dibentuk dengan bervariasi menyesuaikan visualisasi *zebra cross*
3. Titik jangkar (*anchor points)* bervariasi pada tiap gambar
4. Peringatan pemberitahuan dengan menggunakan sumber suara (*github*)

Pengujian *real time* untuk menguji fungsionalitas dan peforma sistem secara keseluruhan. Pengujian ini dilakukan dengan pendeteksian sepeda motor yang tertangkap oleh kamera esp32. Dengan konfigurasi sistem seperti berikut:

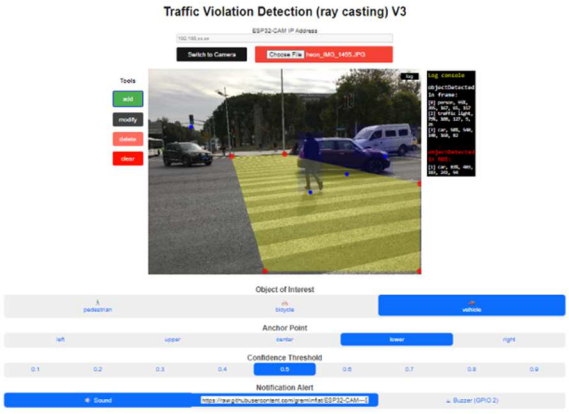
1. Target objek yang terdeteksi dalam *ROI* oleh sistem adalah kendaraan bermotor (sepeda motor).
2. Peringatan pemberitahuan dengan menggunakan *Buzzer*
3. Titik jangkar (*anchor points)* bervariasi

3.2. Hasil Pengujian dan Analisis

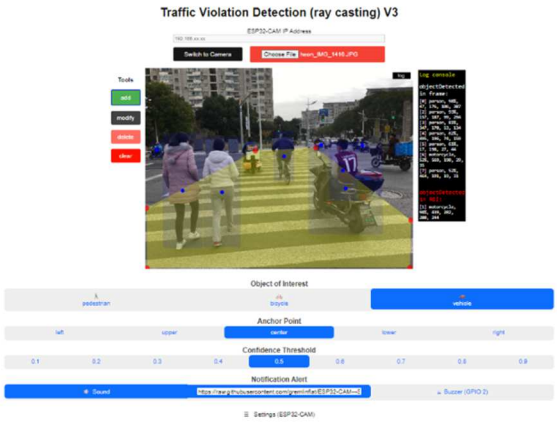
Hasil pengujian pertama dilihat dengan 5 pengujian *non real time* pertama pada beberapa gambar berikut. Gambar 9 adalah hasil sebuah simulasi pengujian sistem yang berhasil mendeteksi objek di dalam *ROI* yang dapat dilihat hasil detailnya pada *log console* dengan deskripsi objek "[0] *car*, 98%, 383, 128, 333, 131" yang menjelaskan bahwa objek berada pada indeks 0, kelas objek mobil, koordinat x dan y serta lebar dan tingginya. Simulasi selanjutnya dimuat pada Gambar 10 dengan konfigurasi titik jangkar "*lower*" berhasil mendeteksi objek di dalam *ROI* dengan tingkat keyakinan yang lebih rendah yaitu 83%. Pada Gambar 11 sistem berhasil mendeteksi satu objek pada kelas sepeda motor dengan keyakinan 98%, namun sistem gagal mendeteksi sepeda motor lain yang berada di dalam *ROI.* Sistem berhasil mendeteksi sepeda motor yang terdapat pada Gambar 11 dengan keyakinan 82%. Simulasi ke-5 pada Gambar 12 menunjukkan sistem berhasil mendeteksi objek pada kelas mobil dengan keyakinan yang besar yaitu 97%. Gambar 9 hingga gambar 13 adalah tangkapan layar dari lima buah contoh hasil inferensi secara *offline* dari 20 sampel dataset uji.



Gambar 9. Hasil Skenario Pengujian 1

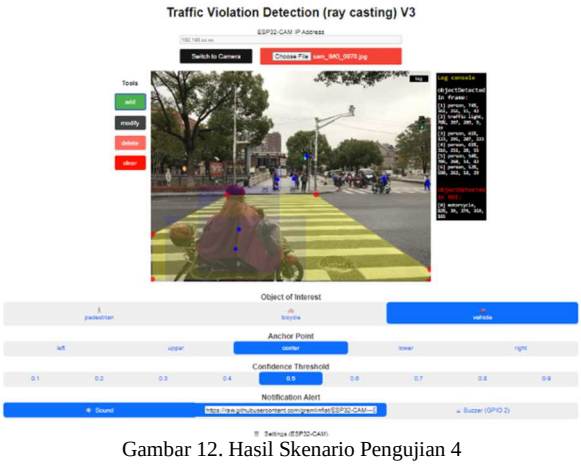


Gambar 10. Hasil Skenario Pengujian 2

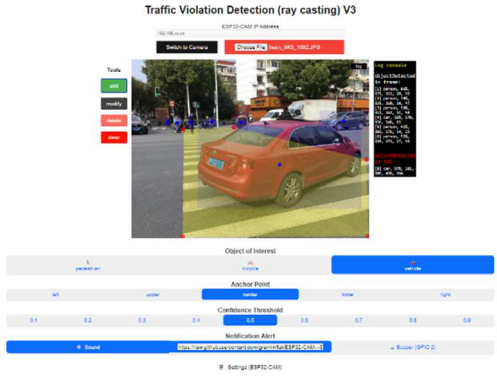


Gambar 11. Hasil Skenario Pengujian 3

DOI 10.38204/tematik.v9i2.997



Gambar 12. Hasil Skenario Pengujian 4



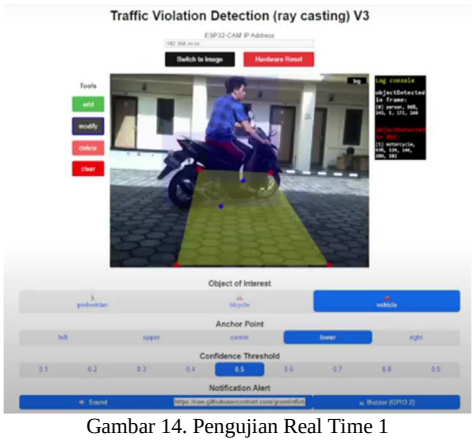
Gambar 13. Hasil Skenario Pengujian 5

Pengujian nilai persentase keyakinan dan akurasi dilakukan pada 20 gambar. Keseluruhan gambar yang telah dilakukan pendeteksian oleh sistem dan memiliki nilai keyakinan akan dihitung rata-rata nilai keyakinan sistem saat mendeteksi objek. Sedangkan untuk persentase akurasi diperoleh dengan membagi dua variable yaitu objek terdeteksi dalam *ROI* (sistem) dibagi dengan jumlah objek dalam *ROI (ground truth).* Rangkuman dari hasil pengujian pada skenario ini dapat dilihat pada tabel 1.

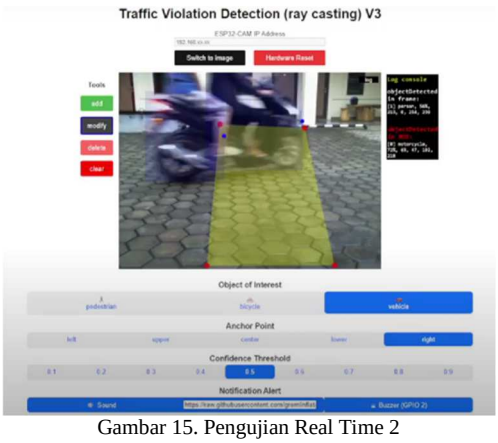
Berdasarkan perhitungan di atas maka diperoleh hasil bahwa sistem ini memiliki rata- rata nilai keyakinan untuk mendeteksi objek sebesar 86% dan untuk presentase akurasi pendeteksian objek juga memiliki nilai yang tinggi yaitu 96.78%.

Untuk hasil pengujian *real time* dapat dilihat hasilnya dengan beberapa gambar di bawah ini. Gambar 14 adalah simulasi pertama dilakukan pada objek sepeda motor yang berhasil mendeteksi objek dengan keyakinan 93% saat objek berada di tengah *ROI* dan memicu aktifnya *buzzer* dan alarm. Pada Gambar 15 simulasi dilakukan dengan objek motor saat bergerak dan sistem dapat mendeteksi objek tersebut dengan keyakinan 72% dan juga menyebabkan *buzzer* aktif. Simulasi ke-3 pada Gambar 16 objek yang berada dekat dengan kamera dan memasuki *ROI* berhasil dideteksi oleh sistem dengan keyakinan 69% dan dapat mengaktifkan sistem notifikasi pelanggaran yaitu *buzzer* dan alarm. Gambar 17 menunjukkan hasil simulasi dengan objek yang berada pada *ROI* bahwa sistem berhasil mendeteksi target objek dengan keyakinan 73% dan memicu aktifnya notifikasi pelanggaran.

Dengan menganalisa hasil pengujian tersebut dapat diperoleh faktor yang menyebabkan nilai keyakinan cukup tingi adalah kualitas dari citra yang digunakan saat pengujian *offline* dan posisi objek pada citra, hal ini juga berlaku untuk pengujian *real-time* dengan menggunakan *ESP32-CAM* bahwa posisi sebuah objek mempengaruhi nilai keyakinan seperti yang dapat diperhatikan pada Gambar 14 hingga Gambar 17 dengan posisi objek (sepeda motor) yang berbeda menghasilkan nilai keyakinan berbeda. Hal ini juga membutikan bahwa fungsionalitas sistem secara keseluruhan dapat diimplementasikan dengan baik, karena tidak adanya objek yang gagal terdeteksi dan gagal untuk mengaktifkan notifikasi pelanggaran baik dengan alarm maupun *buzzer* saat objek memasuki *ROI* walaupun sistem menggunakan perangkat dengan biaya terjangkau yaitu *ESP32-CAM* yang menjadi keunggulan arsitektur sistem yang dirancang. Namun pengujian ini tidak dilakukan pada situasi yang kekurangan cahaya seperti malam hari yang akan mungkin menjadi salah satu faktor yang mempengaruhi nilai keyakinan.



Gambar 14. Pengujian Real Time 1

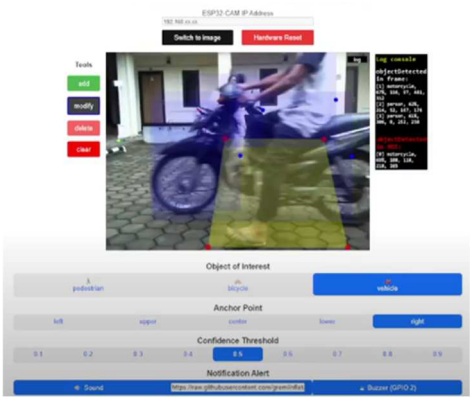


Gambar 15. Pengujian Real Time 2

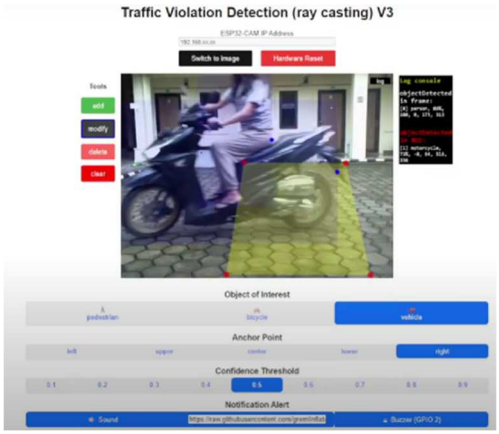
DOI 10.38204/tematik.v9i2.997

Tabel 1. Hasil Pengujian Pada Skenario Offline

No	Jumlah Objek Terdeteksi Pada Gambar	Jumlah Objek Terdeteksi Pada ROI (sistem)	Jumlah Objek Pada ROI (Ground Truth)	Akurasi (%)	Kelas Objek	Nilai Keyakinan (%)
1	1	1	1	100	mobil	96
2	7	2	2	100	mobil	91
					mobil	77
3	5	1	1	100	motor	99
4	4	1	1	100	mobil	86
5	2	1	1	100	mobil	99
6	3	1	2	50	motor	94
7	3	1	1	100	mobil	85
8	5	1	1	100	mobil	98
9	9	1	1	100	motor	76
10	9	3	5	60	mobil	95
					motor	73
					motor	54
11	2	1	1	100	mobil	79
12	8	4	4	100	motor	99
					mobil	81
					mobil	80
					motor	55
13	4	1	1	100	mobil	99
14	4	1	1	100	motor	83
15	2	2	2	100	motor	66
					mobil	56
16	5	2	2	100	mobil	99
					mobil	97
17	6	1	1	100	mobil	94
18	10	1	1	100	motor	94
19	8	1	1	100	mobil	96
20	7	1	1	100	motor	98



Gambar 16. Pengujian Real Time 3



Gambar 17. Pengujian Real Time 4

4. Kesimpulan

Berdasarkan dari hasil perancangan dan pengujian sistem, didapatkan beberapa kesimpulan sebagai berikut. Secara fungsionalitas, sistem yang dirancang mampu mengidentifikasi pelanggaran. Antarmuka yang dirancang juga dapat digunakan untuk mengkonfigurasi beberapa aturan dan persiapan seperti mendefinisikan akses point dan ambang batas kepercayaan. Sistem peringatan berupa *buzzer* sederhana juga berhasil digunakan dan untuk pengembangan kedepan dapat diimplementasikan dalam sistem peringatan yang lebih kompleks.

Di sisi lain, dari aspek inferensi, sistem mampu melakukan pendeteksian pelanggaran baik secara *real-time* maupun *offline*. Secara *real-time*, dari empat kali percobaan dengan berbagai sudut dan dimensi, seluruhnya dapat dideteksi sebagai pelanggaran secara tepat. Dalam pengujian *offline* menggunakan dataset PTL, didapatkan rata-rata hasil akurasi sebesar 96.78% serta rata-rata nilai keyakinan (*average certainty*) sebesar 86%. Untuk nilai keyakinan yang cukup tinggi ini diperoleh sistem dikarenakan sistem berhasil melakukan penyesuai *(fine tune)* dengan menggunakan *MobileNetV2* yang dapat mendukung deteksi objek pada kualitas citra dari *device* yang lebih terjangkau yang menjadi salah satu tujuan penelitian dan dipengaruhi oleh posisi dari objek yang dideteksi. Nilai akurasi yang diperoleh ini adalah hasil pembagian

DOI 10.38204/tematik.v9i2.997

dengan membagi dua variabel yaitu objek terdeteksi dalam *ROI* (sistem) dibagi dengan jumlah objek dalam *ROI (ground truth).* Nilai akurasi yang tinggi ini telah menunjukkan bahwa arsitektur sistem untuk mendeteksi sebuah objek yang terdapat dalam *ROI* berhasil terdeteksi oleh algoritma simplifikasi koordinat dan algoritma *ray casting*. Hal ini menunjukkan bahwa sistem secara reliabel dapat digunakan untuk mendeteksi pelanggaran marka penyebrangan oleh pengendara kendaraan bermotor.

Beberapa tantangan pengembangan dan penelitian lanjutan dapat dilakukan pada penelitian ini. Diantaranya adalah melakukan *transfer learning* dengan menambahkan beberapa dataset yang lebih spesifik terhadap jenis-jenis kendaraan bermotor dan non-bermotor yang ada di Indonesia, misalnya seperti becak, delman, andong, bajaj, dan kendaraan lain yang tidak dilibatkan dalam proses pelatihan model. Tantangan lainnya dapat berupa pengembangan sistem deteksi plat nomor pelanggar sehingga sistem dapat dikembangkan ke arah proses identifikasi pelanggar dan pengenaan sanksi hukum.

Daftar Rujukan

[1] N. K. Jain, R. K. Saini, and P. Mittal, "A Review on Traffic Monitoring System Techniques," in *Soft Computing: Theories and Applications*, 2019, pp. 569–577, doi: 10.1007/978-981-13-0589-4_53 [Online]. Available: http://dx.doi.org/10.1007/978-981-13-0589-4_53

[2] World Health Organization, "Road traffic injuries." [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries. [Accessed: Feb. 20, 2022]

[3] A. Mhalla, T. Chateau, S. Gazzah, and N. E. B. Amara, "An Embedded Computer-Vision System for Multi-Object Detection in Traffic Surveillance," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 11, pp. 4006–4018, Nov. 2019, doi: 10.1109/TITS.2018.2876614. [Online]. Available: http://dx.doi.org/10.1109/TITS.2018.2876614

[4] G. T. S. Ho, Y. P. Tsang, C. H. Wu, W. H. Wong, and K. L. Choy, "A Computer Vision-Based Roadside Occupation Surveillance System for Intelligent Transport in Smart Cities," *Sensors* , vol. 19, no. 8, Apr. 2019, doi: 10.3390/s19081796. [Online]. Available: http://dx.doi.org/10.3390/s19081796

[5] N. A. Khan, N. Z. Jhanjhi, S. N. Brohi, R. S. A. Usmani, and A. Nayyar, "Smart traffic monitoring system using Unmanned Aerial Vehicles (UAVs)," *Comput. Commun.*, vol. 157, pp. 434–443, May 2020, doi: 10.1016/j.comcom.2020.04.049. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366420300189

[6] G. Liu *et al.*, "Smart Traffic Monitoring System Using Computer Vision and Edge Computing," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–12, undefined 2021, doi: 10.1109/TITS.2021.3109481. [Online]. Available: http://dx.doi.org/10.1109/TITS.2021.3109481

[7] M. Arshad and P. Kumar, "Detection of Two-Wheeler Traffic Rule Violation Using Deep Learning," in *2022 IEEE World Conference on Applied Intelligence and Computing (AIC)*, Jun. 2022, pp. 109–114, doi: 10.1109/AIC55036.2022.9848979 [Online]. Available: http://dx.doi.org/10.1109/AIC55036.2022.9848979

[8] Ren, He, Girshick, and Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Adv. Neural Inf. Process. Syst.*, 2015 [Online]. Available: https://proceedings.neurips.cc/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html

[9] H. Zhang, H. Chang, B. Ma, S. Shan, and X. Chen, "Cascade RetinaNet: Maintaining Consistency for Single-Stage Object Detection," *arXiv [cs.CV]*, Jul. 16, 2019 [Online]. Available: http://arxiv.org/abs/1907.06881

[10] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," in *Computer Vision – ECCV 2016*, 2016, pp. 21–37, doi: 10.1007/978-3-319-46448-0_2 [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2

[11] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv [cs.CV]*, Apr. 23, 2020 [Online]. Available: http://arxiv.org/abs/2004.10934

[12] H. Gupta and O. P. Verma, "Monitoring and surveillance of urban road traffic using low altitude drone images: a deep learning approach," *Multimed. Tools Appl.*, vol. 81, no. 14, pp. 19683–19703, Jun. 2022, doi: 10.1007/s11042-021-11146-x. [Online]. Available: https://doi.org/10.1007/s11042-021-11146-x

[13] R. David *et al.*, "TensorFlow lite micro: Embedded machine learning for TinyML systems," *Proceedings of Machine Learning and Systems*, vol. 3, pp. 800–811, Mar. 2021 [Online]. Available: https://proceedings.mlsys.org/papers/2021/73. [Accessed: Feb. 20, 2022]

[14] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *arXiv [cs.CV]*, Jan. 13, 2018 [Online]. Available: http://arxiv.org/abs/1801.04381

[15] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," *arXiv [cs.CV]*, May 01, 2014 [Online]. Available: http://arxiv.org/abs/1405.0312

[16] H. Qassim, A. Verma, and D. Feinzimer, "Compressed residual-VGG16 CNN model for big data places image recognition," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan. 2018, pp. 169–175, doi: 10.1109/CCWC.2018.8301729 [Online]. Available: http://dx.doi.org/10.1109/CCWC.2018.8301729

[17] J. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4507–4515 [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2017/html/Hosang_Learning_Non-Maximum_Suppression_CVPR_2017_paper.html

[18] M. Shimrat, "Algorithm 112: Position of point relative to polygon," *Commun. ACM*, vol. 5, no. 8, p. 434, Aug. 1962, doi: 10.1145/368637.368653. [Online]. Available: https://doi.org/10.1145/368637.368653

[19] S. Yu, H. Lee, and J. Kim, "LYTNet: A Convolutional Neural Network for Real-Time Pedestrian Traffic Lights and Zebra Crossing Recognition for the Visually Impaired," *arXiv [cs.CV]*, Jul. 23, 2019 [Online]. Available: http://arxiv.org/abs/1907.09706

DOI 10.38204/tematik.v9i2.997