

## Deteksi Objek Rambu Lalu Lintas Indonesia Berbasis Algoritma *You Only Look Once* Version 8 Nano (YOLOv8n) Menggunakan Python

Atikah Surriani<sup>1,\*</sup>, Tegar Arif Berbudi<sup>2</sup>

<sup>1,2</sup>Departemen Teknik Elektro dan Informatika, Universitas Gadjah Mada; Jl. Yacarana, Sekip Unit III, Yogyakarta, 55281

Received: 29-06-2026  
Accepted: 23-08-2026

**Keywords:**  
Object Detection;  
Machine Learning;  
Traffic Signs;  
YOLOv8n.

**Correspondent Email:**  
[atikah.surriani.sie13@ugm.ac.id](mailto:atikah.surriani.sie13@ugm.ac.id)

**Abstrak.** Sistem persepsi atau penginderaan merupakan kunci utama kendaraan otonom untuk dapat bergerak secara andal. Salah satu sistem persepsi yang digunakan pada kendaraan otonom adalah sistem deteksi objek. Agar kendaraan otonom dapat bergerak dengan tepat, responsif, dan terhindarkan dari risiko kecelakaan maka dibutuhkan sistem deteksi rambu lalu lintas. Pada penelitian ini, dibangun sistem deteksi objek rambu lalu lintas Indonesia berbasis algoritma deteksi objek *You Only Look Once* version 8 nano (YOLOv8n). Dataset yang digunakan berjumlah 1520 buah citra RGB yang terdiri dari sepuluh jenis kategori rambu lalu lintas beserta kategori background. Pada proses training sistem, dilakukan pengaturan, tiga buah optimizer yaitu SGD, RMSProp, dan Adam dieksperimentasikan pada dataset yang sama untuk mengetahui optimizer terbaik untuk kasus ini. Setelah proses training, dilakukan pengujian pada video rekaman rambu lalu lintas. Dari ketiga hasil training sistem sebanyak 20 epochs, penggunaan optimizer SGD menghasilkan metrik akurasi yang terbaik, yaitu precision sebesar 0.995, recall sebesar 0.985, dan mAP50 sebesar 0.993. Sementara itu, dari hasil pengujian sistem menggunakan optimizer SGD, didapatkan kecepatan deteksi rata-rata sebesar 26 bingkai per detik. Hasil ini menunjukkan bahwa jenis optimizer berpengaruh terhadap tingkat akurasi sistem deteksi objek meskipun dilatih menggunakan dataset yang sama dan optimizer SGD.

**Abstract.** The perception or sensing system is the main key for autonomous vehicles to be able to move reliably. The object detection system is one of the perception systems used in autonomous vehicles. To make autonomous vehicles that can move precisely, responsively, and can avoid the risk of accidents, traffic signs detection system is needed. In this experiment, an Indonesian traffic signs object detection system based on *You Only Look Once* version 8 nano (YOLOv8n) object detection algorithm is built. The dataset used is 1520 RGB images consisting of ten types of traffic sign categories and background images. In the training process, optimizers, SGD, RMSProp, and Adam were experimented on the same dataset to find out the best optimizer for this particular case. From the three system training results of 20 epochs each, the use of the SGD optimizer produces the best accuracy, precision of 0.995, recall of 0.985, and mAP50 of 0.993. Meanwhile, from the test results of that system trained using the SGD optimizer, an average detection speed of 26 frames per second was obtained. The results show that optimizers affect the accuracy of the object detection system even though it is trained using the same dataset and SGD optimizer achieves the best performance.

## 1. PENDAHULUAN

Kendaraan otonom merupakan kendaraan yang dapat beroperasi di jalan raya tanpa memerlukan kendali manusia. Kendaraan otonom dapat berjalan dengan sendirinya karena adanya sistem kontrol cerdas yang menggantikan peran pengemudi sepenuhnya. Sistem kontrol tersebut terdiri dari beberapa bagian dan salah satu bagiannya adalah sistem persepsi atau pengindraan. Dengan tidak adanya campur tangan manusia, sistem persepsi atau pengindraan merupakan kunci utama kendaraan otonom untuk dapat secara andal beroperasi di jalan raya [1].

Salah satu sistem persepsi yang digunakan pada kendaraan otonom adalah sistem deteksi objek. Deteksi objek bekerja dengan mendeteksi jenis objek sekaligus letaknya pada suatu citra atau video. Pada kendaraan otonom, deteksi objek digunakan untuk mendeteksi objek yang terdapat pada citra hasil pengindraan sensor kamera pada lingkungan dan kondisi sekitar kendaraan. Hasil dari deteksi objek dapat digunakan kendaraan otonom untuk berbagai hal seperti pengenalan rambu-rambu lalu lintas, pencegahan tabrakan, pengaturan kecepatan, dan lain-lain [2].

Saat ini, deteksi objek merupakan salah satu bidang kajian dalam visi komputer yang sedang giat diteliti seiring dengan perkembangan pesat teknologi kecerdasan buatan. Di Indonesia, purwarupa kendaraan otonom juga telah diteliti dan dirancang. Badan Riset dan Inovasi Nasional (BRIN) Indonesia sedang melakukan penelitian dan pengembangan purwarupa kendaraan listrik otonom yang dinamai *Micro Electric Vehicle-Teleoperated Driving System* atau MEVi-TDS untuk Ibu Kota Nusantara [3].

Pada eksperimen ini, akan dibuat sistem deteksi objek rambu lalu lintas Indonesia berbasis algoritma YOLO (*You Only Look Once*) yaitu dengan model *You Only Look Once version 8 nano* (YOLOv8n). Pada proses pelatihan dan validasi sistem, dilakukan eksperimen penggunaan tiga jenis *optimizer* yang berbeda yaitu *Stochastic Gradient Descent* (SGD), *Root Mean Square Propagation* (RMSProp), dan Adam. Penggunaan ketiga jenis *optimizer* ini ditujukan agar didapatkan tiga buah hasil pelatihan dan validasi sistem yang masing-masing memiliki tingkat akurasi tersendiri. Dari ketiga hasil

tersebut, dapat diketahui *optimizer* terbaik untuk sistem deteksi objek pada eksperimen ini. Hasil dari eksperimen ini diharapkan dapat menjadi sistem deteksi objek yang mampu secara *real-time* mendeteksi objek rambu lalu lintas Indonesia.

## 2. TINJAUAN PUSTAKA

YOLO (*You Only Look Once*) merupakan algoritma deteksi objek satu tahapan atau disebut *one-stage detector* [16] [17]. YOLO mampu memproses dan mendeteksi objek pada video dengan kecepatan yang sangat tinggi, mencapai 45 bingkai per detik. Algoritma YOLO sangat cocok untuk dijadikan sebagai algoritma deteksi objek *real-time* karena tingkat kecepatan dan keakuratannya yang tinggi [4]. Versi terbaru dari YOLO adalah versi kedelapan, *You Only Look Once version 8* atau YOLOv8 yang rilis pada tahun 2023. Seperti pada versi sebelumnya, YOLOv8 memiliki kinerja yang sangat baik dalam hal kecepatan dan akurasi deteksi. YOLOv8 dapat digunakan dalam berbagai aplikasi deteksi objek dan mudah diimplementasikan dalam berbagai perangkat keras. Kemudahan dalam penggunaan dan implementasi tersebut disebabkan karena desain model yang ringkas dan relatif kecil [5].

## 3. METODE PENELITIAN

### 3.1. Bahan dan Alat

Perancangan deteksi objek rambu lalu lintas ini memerlukan dua bahan utama, yaitu kumpulan data citra atau yang nanti disebut sebagai *dataset* dan data video yang nanti disebut sebagai data uji. Citra pada *dataset* diperoleh dari *Google Maps* dengan cara mengambil tangkapan layar. Data uji yang diambil merupakan video berjalan yang merekam rambu lalu lintas di jalan raya.

Tabel 1 merupakan jenis rambu, jumlah, beserta pembagian data pada *dataset*. Kesepuluh rambu yang digunakan merupakan rambu yang sering dijumpai di jalan raya di Indonesia. Selain itu, rambu yang dipilih merupakan rambu peringatan dan rambu larangan yang merupakan jenis rambu yang penting di jalan raya. Rasio pembagian *dataset* yang digunakan untuk data latihan dan data validasi adalah 8 banding 2. Untuk mengurangi *false positif* pada saat pelatihan dan validasi

sistem, ditambahkan citra *background* yang jumlahnya berkisar pada satu persen dari jumlah total *dataset*.

Tabel 1. *Dataset* yang digunakan

| Kelas Kategori           | Jumlah      | Training Split | Validation Split |
|--------------------------|-------------|----------------|------------------|
| Hati-hati                | 150         | 120            | 30               |
| Turunan                  | 150         | 120            | 30               |
| Tanjakan                 | 150         | 120            | 30               |
| Tikungan ke Kiri         | 150         | 120            | 30               |
| Tikungan ke Kanan        | 150         | 120            | 30               |
| Lampu Lalu Lintas        | 150         | 120            | 30               |
| Persimpangan             | 150         | 120            | 30               |
| Larangan Parkir          | 150         | 120            | 30               |
| Larangan Berhenti        | 150         | 120            | 30               |
| Tempat Pompa Bahan Bakar | 150         | 120            | 30               |
| <i>Background</i>        | 20          | 16             | 4                |
| <b>Total</b>             | <b>1520</b> | <b>1216</b>    | <b>304</b>       |

Tabel 2. Data uji yang digunakan

| Metadata       | Keterangan            |
|----------------|-----------------------|
| Format         | .mp4 .h264            |
| Total durasi   | 29.57 detik           |
| Total frame(s) | 887 frames            |
| Ukuran         | 720 x 1280 x 3 piksel |

Tabel 2 menunjukkan data uji yang digunakan pada eksperimen ini. Data uji merupakan video rekaman rambu lalu lintas yang menyimulasikan tangkapan sensor kamera pada kendaraan saat kendaraan berjalan. Performa deteksi model pada data uji ini menentukan seberapa baik model dalam melakukan deteksi objek rambu lalu lintas pada kondisi yang mendekati kondisi sesungguhnya.

Perangkat keras yang digunakan adalah laptop, sebagai perangkat pelatihan dan

pengujian sistem berbasis model YOLOv8n dan ponsel pintar, untuk merekam video sebagai data uji. Laptop yang digunakan merupakan laptop dengan prosesor *Intel Core i5-8265U Quad-Core* 1,6 GHz, dengan *Random Access Memory* (RAM) *DDR4* 12 GB 2400 MHz, dengan penyimpanan internal berjenis *Solid State Drive* (SSD) *PCIe M.2*, dengan *Graphical Processing Unit* (GPU) *Nvidia GeForce MX250* 2 GB *GDDR5*. Perangkat lunak yang digunakan antara lain bahasa *interpreter Python* versi 3.9, *framework* *LabelImg* versi 1.8.6, *framework* *Anaconda* versi 3, *framework* *PyTorch* versi 2.3.1, platform *NVIDIA CUDA* (*Compute Unified Device Architecture*) versi 12.1, dan pustaka *Ultralytics* versi 8.2.

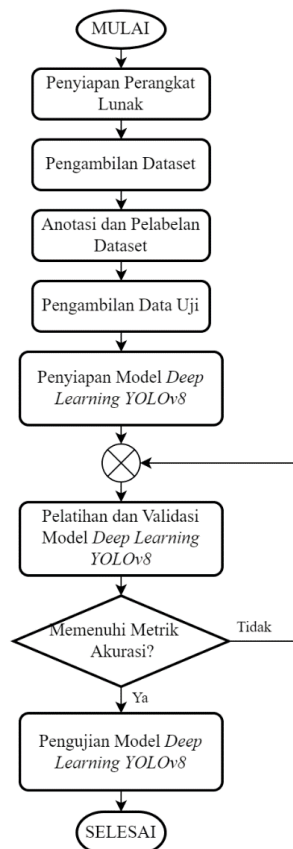
### 3.2. Perancangan Sistem

Sistem yang dieksperimentasikan berbasis model *deep learning* YOLOv8n dari *Ultralytics*, yang merupakan versi terkecil dari YOLOv8. Model ini dipilih karena memiliki waktu deteksi yang paling singkat dari versi YOLOv8 lainnya. Dibutuhkan waktu deteksi yang cepat oleh karena sistem yang dirancang merupakan sistem deteksi rambu lalu lintas yang membutuhkan responsivitas tinggi. Model akan dilatih ulang dengan menggunakan dataset rambu-rambu lalu lintas Indonesia. Secara garis besar, perancangan sistem dimulai dengan penyiapan perangkat lunak pada laptop, pengambilan dataset, anotasi dan pelabelan dataset, pengambilan data uji, pelatihan dan validasi model, dan diakhiri dengan pengujian model. Diagram alur perancangan sistem pada eksperimen ini dapat dilihat pada Gambar 1.

Setelah mendapatkan data yang cukup, dilakukan *pre-processing* data sebelum dataset bisa digunakan sebagai data masukan pada model *deep learning* YOLOv8n. Langkah *pre-processing* data tersebut adalah membuat label dan letak objek (anotasi) dari masing-masing citra. Langkah ini mendefinisikan *bounding box* beserta kelas objek dari setiap objek yang ada pada sebuah citra. Hasil anotasi merupakan fail dengan ekstensi .txt yang berisi lima buah nilai yang merepresentasikan kelas beserta letak dari objek tersebut. Karena terdapat 1500 fail citra bukan *background* pada dataset maka fail label yang didapatkan setelah anotasi juga berjumlah 1500.

Setelah didapatkan *dataset* yang cukup, dilakukan tahapan pelatihan pada sistem

berbasis model deteksi objek YOLOv8n. Pada tahapan pelatihan dan validasi sistem ini, akan dilakukan variasi penggunaan tiga jenis *optimizer* yang berbeda. Optimizer yang akan digunakan secara bergantian adalah SGD (*Stochastic Gradient Descent*), RMSProp, dan Adam.



Gambar 1. Diagram alur perancangan sistem

Ketiga jenis *optimizer* tersebut akan diujicobakan untuk mengetahui *optimizer* terbaik untuk sistem ini pada *dataset* rambu lalu lintas. Pada eksperimen ini, *hyperparameters* pelatihan sistem yang digunakan ditunjukkan pada Tabel 3. Setelah ketiga proses pelatihan sistem selesai, dilakukan pengujian model dengan menggunakan data uji. Pengujian dilakukan pada model yang akurasinya paling tinggi dari ketiga model yang telah dilatih. Pengujian ini akan menghasilkan prediksi model sekaligus waktu deteksi rata-rata. Perangkat yang digunakan pada proses pengujian sistem adalah sama dengan perangkat yang digunakan pada proses pelatihan sistem. Model hasil dari pelatihan sistem, yang kemudian diuji menggunakan perangkat yang

berbeda bisa saja menghasilkan hasil pengujian yang berbeda.

Tabel 3. *Hyperparameters* pada proses pelatihan sistem

| <i>Hyperparameters</i> | Keterangan             |
|------------------------|------------------------|
| <i>Image size</i>      | 640 x 640              |
| <i>Epochs</i>          | 20                     |
| <i>Batch Size</i>      | 10                     |
| <i>Optimizer</i>       | SGD<br>RMSProp<br>Adam |

### 3.3 Evaluasi Sistem

Setelah diuji, sistem akan dievaluasi berdasarkan beberapa parameter, yaitu *Precision*, *Recall*, dan *Mean Average Precision (mAP)* [16]. *Precision* merupakan salah satu metrik yang digunakan untuk mengetahui performansi model. Nilai *precision* merupakan nilai tingkat keakuratan model dalam memprediksi suatu kelas kategori dengan benar dari kemungkinan memprediksi kelas kategori lain. Nilai *precision* sebuah kelas kategori dapat dijadikan acuan ketika kasus yang ingin diklasifikasi membutuhkan hasil prediksi kelas kategori tersebut dengan baik dari banyaknya jumlah kelas kategori. Nilai *precision* dihitung dari perbandingan *true positive (TP)* dengan jumlah *true positive (TP)* dan *false positive (FP)*, sebagaimana persamaan (1)

$$Precision = \frac{TP}{TP+FP} \quad (1)$$

*Recall* merupakan salah satu metrik untuk mengetahui performansi model. Nilai *recall* mengindikasikan tingkat keberhasilan model dalam mengklasifikasi sebuah kelas kategori dengan benar. Nilai *recall* pada sebuah kelas kategori dapat dijadikan acuan ketika kasus yang ingin diklasifikasi membutuhkan prediksi yang baik pada kelas kategori (*ground truth*) tersebut dari banyaknya variasi data pada kelas kategori tersebut. Nilai *recall* dihitung dari perbandingan *true positive (TP)* dengan jumlah *true positive (TP)* dan *false negative (FN)*.

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

*Mean Average Precision (mAP)* merupakan metrik yang umum digunakan untuk mengevaluasi kemampuan sistem deteksi objek. Terdapat beberapa jenis *mAP*, seperti *mAP50*, *mAP75*, dan *mAP90*. Dua angka pada akhiran tersebut menandakan nilai *threshold* dari IoU (*Intersection over Union*). Persamaan untuk menghitung nilai *mAP* ditunjukkan pada Persamaan 3 [20].

$$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP_k. \quad (3)$$

#### 4. HASIL DAN PEMBAHASAN

Pada tahap pelatihan sistem, dilakukan eksperimen pada tiga buah *optimizer* yang berbeda, yaitu SGD, RMSProp, dan Adam. Kemudian, pada tahap pengujian sistem, hasil pelatihan sistem yang terbaik dari ketiga variasi penggunaan *optimizer* diambil untuk diuji menggunakan data uji

##### 4.1 Hasil Latih Sistem

Pada proses pelatihan sistem menggunakan *optimizer* SGD, didapatkan nilai *precision*, *recall*, *mAP50*, dan *mAP50-95* yang meningkat seiring pengulangan latihan. Tabel 4 menunjukkan perubahan metrik akurasi selama 20 kali pengulangan latihan.

Dari Tabel 4, nilai *precision* dan *recall* pada akhir pengulangan latihan adalah 99,51 % dan 98,49 %. Nilai ini menunjukkan bahwa prediksi rata-rata pada seluruh kelas objek sangat baik dalam mengkategorikan objek sesuai kelasnya. Nilai *mAP50* dan *mAP50-95* yang didapatkan juga tinggi yaitu 99,26 % dan 93,71 %. Nilai *mAP* yang tinggi ini menunjukkan bahwa model yang dilatih telah mampu secara baik untuk mendeteksi kesepuluh rambu lalu lintas beserta letaknya pada citra.

Selanjutnya, pada proses pelatihan sistem menggunakan *optimizer* RMSProp, didapatkan nilai *precision*, *recall*, *mAP50*, dan *mAP50-95* yang sangat kecil selama pengulangan latihan dan tidak ada peningkatan yang signifikan. Metrik *precision*, *recall*, dan *mAP50* pada proses ini ditunjukkan pada Tabel 5.

Dari Tabel 5, nilai *precision* dan *recall* pada akhir pengulangan latihan adalah 0.00258 % dan 4 %. Nilai ini menunjukkan bahwa prediksi rata-rata pada seluruh kelas objek

sangat buruk di mana model tidak mampu memprediksi kelas objek dengan benar.

Tabel 4. Metrik akurasi pada penggunaan *optimizer* SGD

| <i>Epochs</i> | <i>Precision</i> | <i>Recall</i> | <i>mAP50</i> |
|---------------|------------------|---------------|--------------|
| 1             | 0.25334          | 0.58986       | 0.3234       |
| 2             | 0.50822          | 0.72922       | 0.60199      |
| 3             | 0.76297          | 0.78819       | 0.83775      |
| 4             | 0.87515          | 0.89211       | 0.93322      |
| 5             | 0.91011          | 0.92449       | 0.96553      |
| 6             | 0.96222          | 0.96134       | 0.98088      |
| 7             | 0.98187          | 0.96931       | 0.9878       |
| 8             | 0.98719          | 0.98272       | 0.99019      |
| 9             | 0.99479          | 0.98416       | 0.99086      |
| 10            | 0.98554          | 0.98567       | 0.9908       |
| 11            | 0.99567          | 0.98399       | 0.99201      |
| 12            | 0.98843          | 0.99047       | 0.99106      |
| 13            | 0.9943           | 0.98531       | 0.99232      |
| 14            | 0.99213          | 0.98875       | 0.99199      |
| 15            | 0.99377          | 0.98911       | 0.99215      |
| 16            | 0.99306          | 0.98727       | 0.99263      |
| 17            | 0.99187          | 0.98979       | 0.99248      |
| 18            | 0.99315          | 0.98835       | 0.99239      |
| 19            | 0.99508          | 0.98615       | 0.99257      |
| 20            | 0.99505          | 0.98485       | 0.99264      |

Nilai *mAP50* dan *mAP50-95* yang didapatkan juga sangat rendah yang bernilai 0.00189 % dan 0.000581 %. Nilai *mAP* yang rendah ini menunjukkan bahwa model yang dilatih tidak mampu secara baik untuk mendeteksi kesepuluh rambu lalu lintas beserta letaknya pada citra.

Pada pelatihan dengan *optimizer* RMSProp ini, terlihat bahwa peningkatan nilai metrik telah berhenti pada pengulangan (*epochs*) ke-8. Dari pengulangan ke-8 sampai dengan ke-20, nilai *precision*, *recall*, dan *mAP* sudah tidak lagi mengalami peningkatan. Kondisi ini mengindikasikan bahwa proses *gradient descent* terjebak pada *local minimum* di mana *optimizer* sudah tidak dapat memperbarui nilai parameter untuk menurunkan nilai *loss*.

Selanjutnya, pada proses pelatihan sistem menggunakan *optimizer* Adam, didapatkan nilai *precision*, *recall*, dan *mAP50* yang meningkat tetapi tidak signifikan dibandingkan dengan penggunaan *optimizer* SGD. Tabel 6 menunjukkan metrik *precision*, *recall*, dan *mAP50* yang dihasilkan. Dari Tabel

6, nilai *precision* dan *recall* pada akhir pengulangan latihan adalah 95,67 % dan 93,59 %. Nilai ini menunjukkan bahwa prediksi rata-rata pada seluruh kelas objek relatif baik di mana model mampu memprediksi kelas objek dengan benar. Nilai mAP50 dan mAP50-95 yang didapatkan adalah sebesar 97,89 % dan 82,60 %. Nilai mAP yang tinggi ini menunjukkan bahwa model yang dilatih mampu secara baik untuk mendeteksi kesepuluh rambu lalu lintas beserta letaknya pada citra.

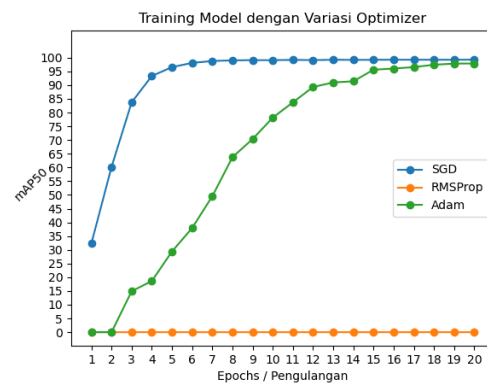
Tabel 5. Metrik akurasi pada penggunaan optimizer RMSProp

| <i>Epochs</i> | <i>Precision</i> | <i>Recall</i> | <i>mAP50</i> |
|---------------|------------------|---------------|--------------|
| 1             | 0                | 0             | 0            |
| 2             | 0                | 0             | 0            |
| 3             | 0,000003         | 0,00333       | 0,0000018    |
| 4             | 0,000012         | 0,0233        | 0,0000085    |
| 5             | 0,000012         | 0,0233        | 0,0000088    |
| 6             | 0,000012         | 0,0233        | 0,0000088    |
| 7             | 0,000012         | 0,0267        | 0,0000088    |
| 8             | 0,000025         | 0,04          | 0,0000189    |
| 9             | 0,000025         | 0,04          | 0,0000189    |
| 10            | 0,000025         | 0,04          | 0,0000189    |
| 11            | 0,000025         | 0,04          | 0,0000189    |
| 12            | 0,000025         | 0,04          | 0,0000189    |
| 13            | 0,000025         | 0,04          | 0,0000189    |
| 14            | 0,000025         | 0,04          | 0,0000189    |
| 15            | 0,000025         | 0,04          | 0,0000189    |
| 16            | 0,000025         | 0,04          | 0,0000189    |
| 17            | 0,000025         | 0,04          | 0,0000189    |
| 18            | 0,000025         | 0,04          | 0,0000189    |
| 19            | 0,000025         | 0,04          | 0,0000189    |
| 20            | 0,000025         | 0,04          | 0,0000189    |

Hasil pelatihan sistem pada variasi *optimizer* secara garis besar dapat dilihat pada nilai mAP50-nya. Untuk mengetahui *optimizer* terbaik pada kasus ini, Gambar 2 menunjukkan kenaikan nilai mAP50 pada setiap *epochs* atau pengulangan latihan pada ketiga penggunaan *optimizer*.

Tabel 6. Metrik akurasi pada penggunaan optimizer Adam

| <i>Epochs</i> | <i>Precision</i> | <i>Recall</i> | <i>mAP50</i> |
|---------------|------------------|---------------|--------------|
| 1             | 0                | 0             | 0            |
| 2             | 0.00018          | 0.03656       | 0.00017      |
| 3             | 0.86895          | 0.08667       | 0.14919      |
| 4             | 0.18316          | 0.29591       | 0.1863       |
| 5             | 0.24287          | 0.487         | 0.29391      |
| 6             | 0.34436          | 0.56205       | 0.37952      |
| 7             | 0.45158          | 0.52323       | 0.49512      |
| 8             | 0.59911          | 0.57483       | 0.63732      |
| 9             | 0.71341          | 0.6176        | 0.70347      |
| 10            | 0.77375          | 0.68898       | 0.78132      |
| 11            | 0.79014          | 0.74229       | 0.83728      |
| 12            | 0.88009          | 0.78886       | 0.89293      |
| 13            | 0.88802          | 0.83631       | 0.90973      |
| 14            | 0.87465          | 0.83748       | 0.91374      |
| 15            | 0.91892          | 0.88252       | 0.95613      |
| 16            | 0.92231          | 0.90901       | 0.96027      |
| 17            | 0.93551          | 0.89414       | 0.96553      |
| 18            | 0.94711          | 0.93134       | 0.97404      |
| 19            | 0.9625           | 0.91662       | 0.97832      |
| 20            | 0.95668          | 0.93593       | 0.97892      |



Gambar 2. Hasil pelatihan sistem dengan tiga variasi penggunaan optimizer

Dari hasil latihan yang ditunjukkan pada Gambar 2, *optimizer* SGD dan Adam berhasil meningkatkan akurasi sistem pada *dataset* rambu lalu lintas. Sementara itu, *optimizer* RMSProp terjebak pada *local minimum loss function* sehingga metrik tidak mengalami peningkatan yang signifikan. Peningkatan nilai mAP pada *optimizer* SGD merupakan yang paling cepat dan paling tinggi dibandingkan dengan *optimizer* lain. Penggunaan *optimizer* Adam juga menghasilkan peningkatan nilai mAP yang baik namun tidak secepat pada

penggunaan *optimizer* SGD. Sementara itu, pelatihan dengan *optimizer* RMSProp menghasilkan nilai mAP yang sangat rendah dan tidak begitu signifikan perubahannya.

Dari Gambar 2 juga, dapat dilihat bahwa penggunaan *optimizer* SGD memang menghasilkan nilai mAP yang paling tinggi. Penggunaan *optimizer* ini juga menghasilkan peningkatan nilai mAP yang lebih cepat. Jika dibandingkan dengan penggunaan *optimizer* Adam, nilai mAP yang dihasilkan memang tidak lebih tinggi dari penggunaan *optimizer* SGD. Namun, dari *epoch* ke *epoch*, nilai mAP mengalami peningkatan secara konsisten. Peningkatan nilai mAP ini mendekati *linear*, yang berarti penggunaan *optimizer* Adam ini menghasilkan pembelajaran model yang terus belajar setiap *epoch*-nya. Kondisi ini berbeda dibandingkan dengan penggunaan *optimizer* SGD yang naik secara cepat akan tetapi nilai mAP tidak lagi naik setelah mencapai beberapa *epoch*. Dari hasil pelatihan dengan tiga *optimizer* tersebut, dapat disimpulkan bahwa *optimizer* Adam merupakan *optimizer* terbaik pada kasus ini karena nilai mAP yang selalu naik pada tiap perulangan latihan yang kenaikannya mendekati *linear*.

Gambar 3 merupakan *confusion matrix* yang didapatkan pada proses pelatihan sistem menggunakan *optimizer* SGD. *Confusion matrix* yang dihasilkan mendekati matriks diagonal yang berarti model dapat secara baik memprediksi kesepuluh kelas objek rambu lalu lintas. Terlihat bahwa tidak terdapat kesulitan bagi model untuk membedakan suatu kelas objek dari kesepuluh kelas objek rambu lalu lintas. Dari *confusion matrix* tersebut, sangat sedikit *background* yang diprediksi sebagai objek (sangat sedikit *false positive*). Hal ini menunjukkan bahwa model juga dapat dengan baik mengenali dan membedakan antara *background* dan objek.

Dari hasil pelatihan sistem, dapat disimpulkan bahwa sistem berbasis model YOLOv8n yang dilatih memiliki tingkat akurasi yang tinggi pada dataset rambu lalu lintas. Model telah mampu untuk mempelajari dengan baik kesepuluh kelas rambu lalu lintas pada dataset. Hasil perbandingan sistem berdasarkan *optimizer* ditampilkan pada Tabel 7.

Tabel 7. Perbandingan Metrik pada setiap Optimizer

| <i>Epoch</i> | <i>Optimizer</i> | <i>Precision</i> | <i>Recall</i> | mAP50   |
|--------------|------------------|------------------|---------------|---------|
| 20           | SGD              | 99,51            | 98,49         | 99,26   |
| 20           | RMSProp          | 0,00258          | 4             | 0,00189 |
| 20           | Adam             | 95,67            | 93,59         | 97,89   |

Pada proses pelatihan sistem sebelumnya, penggunaan *optimizer* SGD menghasilkan nilai mAP50 yang tertinggi. Maka dari itu, pengujian sistem dilakukan pada hasil tersebut. Hasil pengujian pada data uji menghasilkan hasil prediksi yang baik. Seluruh kelas objek yang ada pada data uji berhasil untuk diprediksi dengan tingkat *confidence* yang juga tinggi. Tidak terdapat kesalahan prediksi (*false positive*) pada hasil pengujian ini. Model dapat dengan benar mendeteksi jenis kelas objek rambu lalu lintas. (*Bounding box*) prediksi yang dihasilkan juga sangat baik di mana model mampu memprediksi letak dan ukuran objek rambu lalu lintas dengan tepat. Gambar 6 menunjukkan beberapa buah frame prediksi dari pengujian model pada data uji.



Gambar 6. Beberapa frame hasil prediksi pada pengujian sistem.

## 5. KESIMPULAN

Sistem deteksi objek rambu lalu lintas Indonesia berbasis model YOLOv8n yang dihasilkan pada eksperimen ini mampu untuk mendeteksi sepuluh jenis objek rambu lalu lintas Indonesia dengan nilai mAP50 sebesar 99,26 %. Dengan hasil ini, model YOLOv8n dapat dijadikan sebagai sistem deteksi objek rambu lalu lintas Indonesia. Penggunaan optimizer SGD menghasilkan parameter nilai mAP50 tertinggi sebesar 99,26 %, precision 99,51% dan recall 98,49%. Hal ini menunjukkan SGD menjadi optimizer terbaik.

## UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada pihak-pihak terkait yang telah memberi dukungan terhadap penelitian ini.

## DAFTAR PUSTAKA

- [1] Y. Zhang, A. Carballo, H. Yang, and K. Takeda, "Perception and sensing for autonomous vehicles under adverse weather conditions: A survey," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 196, pp. 146–177, 2023, doi: <https://doi.org/10.1016/j.isprsjprs.2022.12.021>.
- [2] "Advancement of Deep Learning and its Applications in Object Detection and Recognition," in *Advancement of Deep Learning and its Applications in Object Detection and Recognition*, 2022, pp. i–xxxii.
- [3] A. Waskito *et al.*, "Analysis of Power Consumption for Electrical Motor on Automatic Micro Electric Vehicle (MEVi)," in *2023 International Conference on Advanced Mechatronics, Intelligent Manufacture and Industrial Automation (ICAMIMIA)*, 2023, pp. 637–643. doi: 10.1109/ICAMIMIA60881.2023.10427582.
- [4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.
- [5] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8," 2023. [Online]. Available: <https://github.com/ultralytics/ultralytics>
- [6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014.
- [7] H. Qian, Y. Ma, W. Chen, T. Li, Y. Zhuo, and W. Xiang, "Traffic signs detection and segmentation based on the improved mask r-cnn," in *2021 40th Chinese Control Conference (CCC)*, 2021, pp. 8241–8246.
- [8] R. Qian, Q. Liu, Y. Yue, F. Coenen, and B. Zhang, "Road surface traffic sign detection with hybrid region proposal and fast r-cnn," in *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, 2016, pp. 555–559.
- [9] Z. Zuo, K. Yu, Q. Zhou, X. Wang, and T. Li, "Traffic signs detection based on faster r-cnn," in *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, 2017, pp. 286–288.
- [10] P. Long and Y. Hu, "Multiple image augmentations for enhanced yolo-based traffic sign detection," in *2023 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML)*, 2023, pp. 132–137.
- [11] N. Choudhary, R. Sharma, D. Upadhyay, A. Verma, and V. Jain, "Enhanced traffic sign recognition using advanced yolov8 model," in *2024 4th International Conference on Intelligent Technologies (CONIT)*, 2024, pp. 1–4.
- [12] R. Gandhi. (2018) R-cnn, fast r-cnn, fasterr-cnn, yolo — object detection algorithms. [Online]. Available: <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>.
- [13] K. Geng and G. Yin, "Using deep learning in infrared images to enable human gesture recognition for autonomous vehicles," *IEEE Access*, vol. 8, pp. 88 227–88 240, 2020.
- [14] I. V. Fanthony, Z. Husin, H. Hikmarika, S. Dwijayanti, and B. Y. Suprpto, "Yolo algorithm-based surrounding object identification on autonomous electric vehicle," in *2021 8th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 2021, pp. 151–156.
- [15] A. Wibowo, B. R. Trilaksono, E. M. I. Hidayat, and R. Munir, "Object detection in dense and mixed traffic for autonomous vehicles with modified yolo," *IEEE Access*, vol. 11, pp. 134 866–134 877, 2023.
- [16] Y. F. Yudha, A. A. Y. Aditya, R. A. Y. Rasyid, N. I. A. Indra, and M. W. W. Melati, "Perancangan Sistem Deteksi Objek Pada Robot Transporter Menggunakan Metode Darknet YOLOv8," *Electr. J. Rekayasa dan*

- Teknol. Elektro*, vol. 18, no. 2, pp. 161–170, 2024, doi: 10.23960/elc.v18n2.2595.
- [17 ] A. B. Harahap, N. D. G. Drantantiyas, and I. A. Sasmita, “Perancangan Sistem Deteksi Peta Panas (Heatmap) Keramaian Pengunjung di Area Publik Selama Pandemi COVID-19 Berbasis YOLOV4-Tiny,” *Electr. J. Rekayasa dan Teknol. Elektro*, vol. 19, no. 3, pp. 262–269, 2025, [Online]. Available: <https://electrician.unila.ac.id/index.php/ojs/article/view/2796>