SKANIKA: Sistem Komputer dan Teknik Informatika
Volume 7, Nomor 2, Juli 2024, Halaman 166-179 E-ISSN: 2721-4788

Pengembangan Sistem Perhitungan Jumlah Kendaraan Berdasarkan Jenis Kendaraan Menggunakan Algoritma YOLO Secara *Realtime*

Adri Surya Kusuma[1*], Afu Ichsan Pradana, Bondan Wahyu Pamekas[3]
[1,2,3]Fakultas Ilmu Komputer, Teknik Informatika, Universitas Duta Bangsa, Surakarta, Indonesia
E-mail: [1*]202030106@mhs.udb.ac.id, [2]afu_ichsan@udb.ac.id, [3]bondan_wahyupamekas@udb.ac.id
(* : corresponding author)

Abstrak

Tahun 2024 di Indonesia ditandai dengan pemilihan umum serentak, yang menyebabkan peningkatan signifikan dalam penggunaan media luar ruang untuk kampanye politik. Tantangan utama adalah mendapatkan data lalu lintas yang akurat dan real-time guna meningkatkan strategi serta efektivitas iklan media tersebut. Penelitian ini mengembangkan sistem perhitungan jumlah kendaraan berdasarkan jenis kendaraan menggunakan algoritma YOLO secara real-time untuk mengolah citra CCTV di media luar ruang. Metode penelitian melibatkan pelatihan model YOLOv9 dan pelacakan berbasis centroid pada 10000 dataset, 10 batch, dan 100 epoch yang mencakup empat class kendaraan. Evaluasi model mendapatkan nilai 97,5% accuracy pada tugas klasifikasi dan 90% mAP 90,9% precision 82,8% recall 86,7% F1-Score pada tugas deteksi. Pengujian perhitungan kendaraan pada area ROI dalam 1 menit mendapatkan 90,85% accuracy dengan 18-21 FPS. Kontribusi utama penelitian ini dalam menyediakan data lalu lintas yang akurat dan real-time untuk perencanaan lalu lintas, keamanan, dan analisis pemasaran yang dapat dimanfaatkan oleh pemerintah daerah, perusahaan manajemen lalu lintas, dan perusahaan iklan. Dari hasil penelitian mengindikasikan sistem yang dibangun memiliki kemampuan deteksi dan pelacakan kendaraan dengan akurasi tinggi, namun kinerja sistem masih dapat ditingkatkan dengan pelatihan model dan spesifikasi perangkat keras untuk komputasi yang lebih baik lagi.

Kata kunci: *YOLOv9, Deteksi Kendaraan, Confusion Matrix, Real-time Computer Vision*

Abstract

In 2024, Indonesia will experience simultaneous general elections, resulting in a significant increase in the use of outdoor media for political campaigns. The primary challenge is to obtain accurate and real-time traffic data to enhance the strategy and effectiveness of these media advertisements. This study develops a system for counting the number of vehicles based on their type using the YOLO algorithm in real-time, processing CCTV images in outdoor media. The research method involves training a YOLOv9 model and centroid-based tracking on a dataset of 10000 images, 10 batches, and 100 epochs, encompassing four vehicle classes. Model evaluation achieved 97,5% accuracy in classification tasks and 90% mAP 90,9% precision 82,8% recall and 86,7% F1-Score in detection tasks. Vehicle count testing in the ROI area within 1 minute obtained 90,85% accuracy with 18-21 FPS. The main contribution of this research is providing accurate and real-time traffic data for traffic planning, security, and marketing analysis, benefiting local governments, traffic management companies, and advertising firms. The results indicate that the developed system has high detection and tracking accuracy; however, system performance can be further improved with enhanced model training and better hardware specifications for improved computation.

***Keywords**: YOLOv9, Vehicle Detection, Confusion Matrix, Real-time Computer Vision*

1. PENDAHULUAN

Pemilihan umum serentak tahun 2024 di Indonesia memicu peningkatan penggunaan media luar ruang seperti *billboard*, baliho, videotron, JPO, dan bando untuk kampanye politik. Media ini digunakan untuk menarik perhatian publik dan meningkatkan kesadaran melalui penempatan iklan yang strategis. Dalam konteks ini, jumlah kendaraan yang melewati media iklan tersebut menjadi indikator penting efektivitas iklan. Semakin tinggi jumlah kendaraan yang melewati media *above the line* tersebut, maka semakin besar pula potensi paparan iklan tersebut untuk mendapatkan atensi publik serta meningkatkan *awareness* melalui *image* dan *positioning* nya [1]. Iklan melalui media luar ruang ini seperti baliho masih menjadi pilihan dalam kepentingan pemilihan umum untuk menerapkan iklan politik jika dibandingkan melalui media

elektronik [2], namun kelemahan media ini masih ada karena konten yang bersifat statis dan menyebabkan target konsumen dari iklan menjadi tidak tepat [3].

Seiring perkembangan teknologi *artificial intelligence* khususnya bidang *computer vision*, integrasi sistem kamera pengawas memungkinkan informasi visual dapat dideteksi, di intepretasikan, dan direspon [4]. Secara umum perkembangan ini juga dapat menjadi solusi efisien bagi pemerintah daerah, perusahaan manajemen lalu lintas, pengembang properti, dan melalui perusahaan iklan itu sendiri untuk memperoleh data yang akurat dan *real-time* dalam perencanaan lalu lintas, keamanan, dan analisis pemasaran. Salah satu solusi untuk mencapai hal tersebut adalah dengan mengembangkan sistem perhitungan jumlah kendaraan berdasarkan jenis kendaraan menggunakan algoritma YOLO.

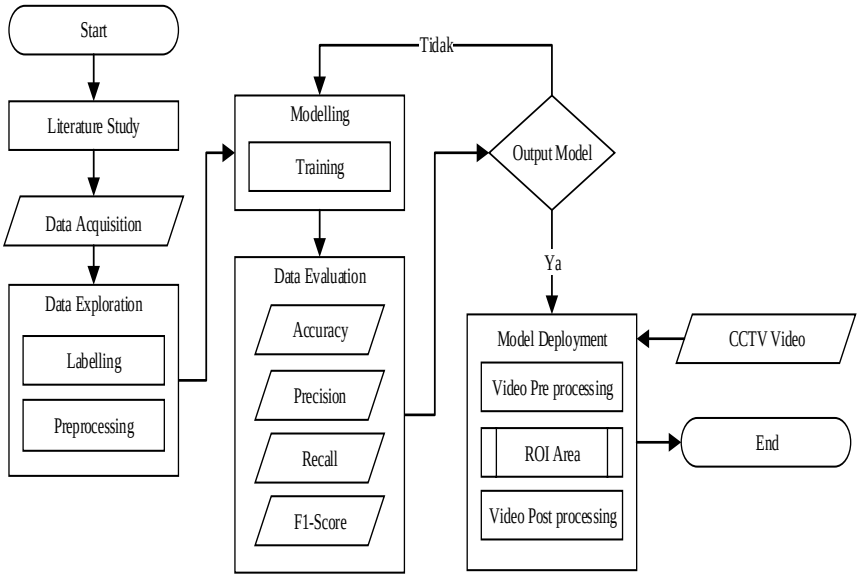
YOLO kepanjangan dari *You Only Look Once* memisahkan gambar *input* menjadi bagian-bagian dalam bentuk grid yang terdiri dari beberapa sel. [5]. Dengan karakter *state of the art* nya dalam kondisi *real-time*, algoritma ini berfungsi sebagai model *detector* terintegrasi dan terpadu yang secara langsung melakukan prediksi *bounding box* atau kotak pembatas dan probabilitas *class* untuk seluruh gambar dalam satu evaluasi tunggal (*single shot*). [6]. *Mask* R-CNN algoritma pada deteksi *object*, membutuhkan dua tahap untuk menghasilkan proposal *region* kemudian mengklasifikasi dan memfilternya. Pada Algoritma deteksi *object* lain, SSD menggunakan satu jaringan konvolusi untuk menghasilkan deteksi pada berbagai skala dengan prediksi *bounding box* dan skor *class* dari beberapa *feature maps*. Algoritma YOLO lebih menawarkan keseimbangan dan mampu mengatasi beberapa kelemahan yang mungkin dimiliki oleh algoritma deteksi objek sebelumnya seperti model *mask* R-CNN dan SSD [7]. YOLO memiliki keunggulan pada kemampuan dan kecepatan untuk mendeteksi *multiple object* berdasarkan kebenaran hasil keluaran sistem dan ketepatan waktu yang dihasilkan dengan tingkat akurasi yang tinggi [8]-[9]. Pada versi YOLOv9 memperkenalkan pendekatan *deep learning* yang berfokus pada *loss* dari informasi dan efisiensi arsitektur jaringan yang lebih dalam.

Pada penelitian sebelumnya oleh Saifullah, Hegarini, dan Wardijono telah berhasil membuat sistem perhitungan jumlah kendaraan mobil menggunakan metode *Haar Cascade Classifier*. Tingkat keberhasilan sistem ini 40% dalam menghitung mobil dengan kecepatan yang berbeda-beda melalui ROI *line* yang dibuat. Penelitian ini mengharapkan teknik pelacakan kontinyu dan peningkatan *preprocessing* data pada penelitian selanjutnya. Sistem perhitungan kendaraan pada penelitian lain menggunakan algoritma *K-Nearest Neighbor* oleh Suradi, Rasyid, dan Nasaruddin mampu melakukan perhitungan dengan akurasi rata-rata 78,5%. Penelitian yang pernah dilakukan menggunakan algoritma YOLOv8 pada perhitungan kendaraan oleh Hayati, Singasatia, dan Muttaqin dengan jumlah *custom* dataset sebanyak 4680, *preprocessing image size* 416x416, dan *epochs* 50 telah berhasil mendapatkan nilai model 86% *accuracy*, 86% *precision*, 86% *recall*, dan 85% *F1-Score*. Penelitian ini mengharapkan pengumpulan dataset dari pihak ketiga dan membandingkan dengan versi YOLO yang lain di penelitian selanjutnya.

Pengembangan sistem perhitungan jumlah kendaraan berdasarkan jenis kendaraan menggunakan algoritma YOLO secara *real-time* dalam penelitian ini diharapkan mampu menghasilkan solusi yang optimal dan efisien untuk meningkatkan strategi periklanan serta berbagai pihak yang tertarik untuk melakukan analisis *traffic* kendaraan, keperluan bisnis maupun dalam tujuan penelitian yang lain.

2. METODE PENELITIAN

Dalam membuat metode, terdapat serangkaian tahapan atau kerangka kerja yang diikuti untuk memastikan alur penelitian tersampaikan secara keseluruhan. Dalam penelitian ini, flowchart pada gambar 1 digunakan sebagai representasi visual dari tahapan metode, yang menggambarkan penerapan model algoritma YOLO untuk mengembangkan sistem perhitungan jumlah kendaraan berdasarkan jenisnya secara *real-time*.



Gambar 1. Tahapan Pengembangan Sistem Perhitungan Kendaraan Menggunakan YOLO

2.1. YOLOv9

YOLOv9 adalah YOLO versi yang sedang dikembangkan Ultralytics bekerjasama dengan *open-source team* dibangun berdasarkan basis kode kuat yang disediakan pada YOLOv5 dengan memperkenalkan peningkatan arsitektur. YOLOv9 menggabungkan kemajuan seperti PGI kepanjangan dari *Programmable Gradient Information* (PGI) dan GELAN kepanjangan dari *Generalized Efficient Layer Aggregation Network* [13]. PGI melakukan pencegahan dari data yang *loss* selama pembaruan gradien, sementara GELAN mengoptimalkan model ringan melalui perencanaan jalur gradien. Melalui penggabungan PGI dan adaptasi Arsitektur GELAN, YOLOv9 tidak hanya meningkatkan kemampuan pembelajaran model tetapi juga menjamin informasi penting selama proses deteksi [14].

2.2. Data *Acquisition*

Data *acquisition* pada penelitian ini bersifat mengkombinasikan beberapa sumber dataset. Selanjutnya, dataset ini dikumpulkan melalui platform Roboflow yang memiliki fungsionalitas terkait dengan pengumpulan data. Roboflow juga menawarkan fitur-fitur seperti *anotation*, *preprocessing*, *augmentation*, dan dataset *export* yang mampu meningkatkan kecepatan dalam pengembangan sistem kecerdasan buatan. [15].

2.3. Data *Exploration*

Data *exploration* dilakukan dengan *image labeling* atau menandai objek dengan kotak pembatas dan menetapkan *class* yang tepat untuk setiap gambar untuk mendapatkan *anotation* pada data citra dan dilanjutkan pembagian set data dengan perbandingan 80% *train* data, 15% *valid* data, dan 5% *test* data. Tahap ini juga terdapat *preprocessing* yang merupakan langkah awal sebelum dilakukan proses pengenalan pola [16] dengan menerapkan *Auto-orient* dan *resize* semua data citra dengan ukuran 640x640.

2.4. *Modelling*

Modelling merupakan tahap pelatihan yang dilakukan untuk mengidentifikasi pola-pola dalam dataset dengan tujuan membuat prediksi dan mengambil keputusan. Tahap ini diawali dengan melakukan *export* dataset hasil data *exploration* dari Roboflow ke Google Colab, sebuah platform *cloud* Google berbasis Python yang memberikan akses layanan GPU sebagai *backend* komputasi [17].

2.5. Data *Evaluation*

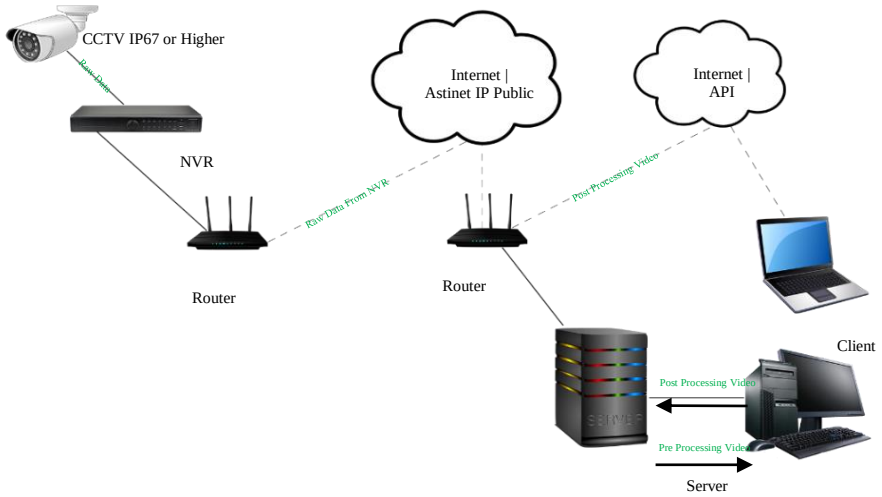
Data *evaluation* mengacu pada *confusion matrix* untuk menghasilkan 4 parameter, yaitu *accuracy*, *precision*, *recall*, dan *F1-Score* [18]. *Confusion matrix* menilai kinerja suatu model pembelajaran mesin dengan memperlihatkan prediksi klasifikasi aktual dan prediksi klasifikasi yang dilakukan [19]. Tabel 1 menampilkan *confusion matrix* sebagai representasi variabel untuk memenuhi keempat parameter tersebut.

Tabel 1. *Confusion Matrix*

Confusion Matrix		**Actual**	
		True	**False**
Prediction	**True**	*True Positive* (TP): mengacu pada tipe sampel yang termasuk dalam *class positive* kemudian terklasifikasi dengan benar.	*False Positive* (FP): mengacu pada tipe sampel yang termasuk dalam *class negative* tetapi salah diklasifikasikan sebagai milik *class positive*.
	False	*False Negative* (FN): mengacu pada tipe sampel yang termasuk dalam *class positive* tetapi salah diklasifikasikan sebagai milik *class negative*.	*True Negative* (TN): mengacu pada tipe sampel yang termasuk dalam *class negative* yang diklasifikasikan dengan benar.

2.6. Model *Deployment*

Tahap ini merupakan pengujian hasil *output* model yang telah dilatih. Pengujian dilakukan pada rekaman CCTV *billboard* dengan setiap *class* yang terdeteksi dilacak (*tracking*) menggunakan metode pelacakan berbasis centroid, di mana posisi pusat (centroid) dari setiap *class* dihitung. Gambar 2 menunjukan *workflow* dimana data RTSP CCTV diambil melalui *ip public* yang disediakan Astinet kemudian server melakukan *preprocessing* video dengan model latih.



Gambar 2. *Workflow Processing* Data CCTV Pada Sistem

Server akan memproses video jika posisi pusat *class* ini berpindah dari satu *frame* ke *frame* berikutnya, pelacakan memastikan bahwa *class* yang sama dikenali berdasarkan jarak antara pusat-pusat tersebut. Area ROI (*Region of Interest*) didefinisikan dalam *frame* video untuk menentukan apakah kendaraan berada dalam area tertentu [20]. Tabel 2 menunjukan tiga komponen utama yang digunakan pada pelacakan berbasis centroid ini.

Tabel 2. Komponen Utama Sistem Perhitungan Jumlah Kendaraan

Komponen	**Input**	**Output**
Detector	*frame* dari video CCTV	Lokasi *bounding boxes* dan jenis kendaraan yang terdeteksi
Tracker	*frame* baru	*Bounding box* yang diperbarui untuk setiap objek
Counter	Centroid dan posisi dari area ROI.	kendaraan berada di dalam atau di luar area ROI.

Detector menggunakan model YOLOv9 yang telah dilatih untuk mendeteksi kendaraan dalam *frame* video dan mengembalikan daftar *bounding boxes* serta jenis kendaraan yang terdeteksi ke *tracker*. *Tracker* kemudian menggunakan *bounding boxes* ini untuk melacak posisi kendaraan di *frame* berikutnya menggunakan metode pelacakan berbasis centroid, yang menghitung posisi pusat (centroid) setiap kendaraan. *Detector* juga digunakan untuk memperbarui *tracker* secara berkala guna memastikan pelacakan tetap akurat. *Counter* menggunakan dua area ROI untuk menentukan apakah kendaraan berada dalam area yang relevan.

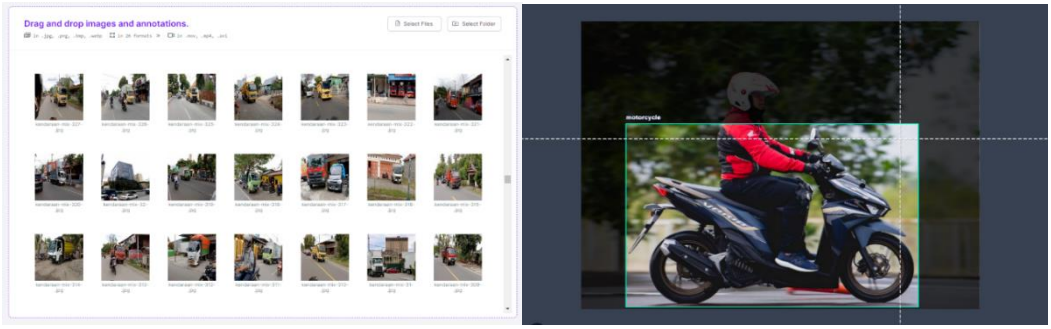
Pada perhitungan total seluruh *class* jalur kendaraan yang menuju *billboard*, *counter* akan mengkategorikan *watch billboard.* Jalur dengan arah meninggalkan *billboard*, *counter* akan mengkategorikan *no watch billboard.* Saat sebuah kendaraan *class* tertentu memasuki atau keluar dari area ROI, sistem ini mencatat dan menghitung jumlah kendaraan yang termasuk dalam setiap *class* serta total seluruh *class* dengan kategori *watch billboard* dan *no watch billboard*.

Hasil *post processing* berupa deteksi dan pelacakan ini ditampilkan dalam video berformat MJPEG dengan anotasi visual, dan jumlah setiap *class*, seperti *car*, *motorcycle*, *bus*, dan *truck* dihitung berdasarkan apakah mereka berada dalam atau di luar area ROI. Video *post processing* dapat diakses melalui domain dari server, sedangkan data perhitungan ini akan dikirim ke *cloud* dari *firebase real-time* database untuk dapat di akses oleh *client.*

3. HASIL DAN PEMBAHASAN

3.1 Data *Acquisition*

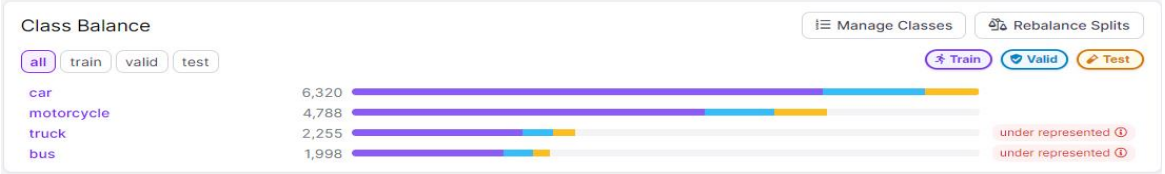
Tahap data *acquisition* yang dikumpulkan pada penelitian ini berjumlah 5000 dengan pembagian 20% untuk masing-masing *class* yaitu *car*, *motorcycle*, *bus*, dan *truck* berasal dari *hunting* citra dijalan secara langsung, Kaggle, serta pencarian di Google ditunjukan pada gambar 3. Data 20% terakhir berasal dari keluaran citra yang dihasilkan oleh CCTV billboard yang digunakan pada penelitian.



Gambar 3. Tahapan Data *Acquisition* (kiri) dan *Labelling* Data *Exploration* (kanan)

3.2 Data *Exploration*

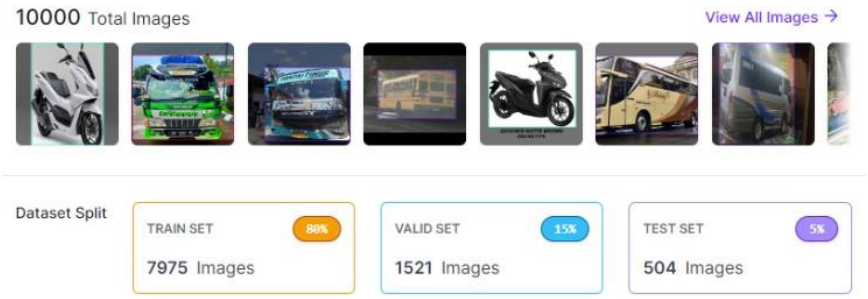
Tahapan ini diawali dengan *labelling* yang ditunjukan pada gambar 3. Hasil data *exploration* setelah pelabelan mendapatkan 15361 *anotation*, namun fitur *dataset health check* Roboflow menunjukkan ketidakseimbangan jumlah *class* yang dianotasi, terutama pada *class truck* dan *bus*. Ketidakseimbangan ini di ilustrasikan dalam gambar 4.



Gambar 4. *Unbalancing Class Dataset*

Data citra tambahan diambil dari dataset publik Roboflow *Universe* yang sesuai pada penelitian ini digunakan untuk lebih menyeimbangkan masing-masing *class*, sehingga total akhir dataset adalah 10000 data citra, data *exploration* 20428 anotasi dengan perbandingan antara *train set*, *valid set*, dan *test set* adalah 80:15:5 tampak pada gambar 5. Dataset publik yang dipilih menggunakan *augmentation rotation*, *shear*, dan *grayscale* pada level citranya. Hasil akhir dataset yang digunakan pada tahapan ini dapat dilihat pada gambar 6.

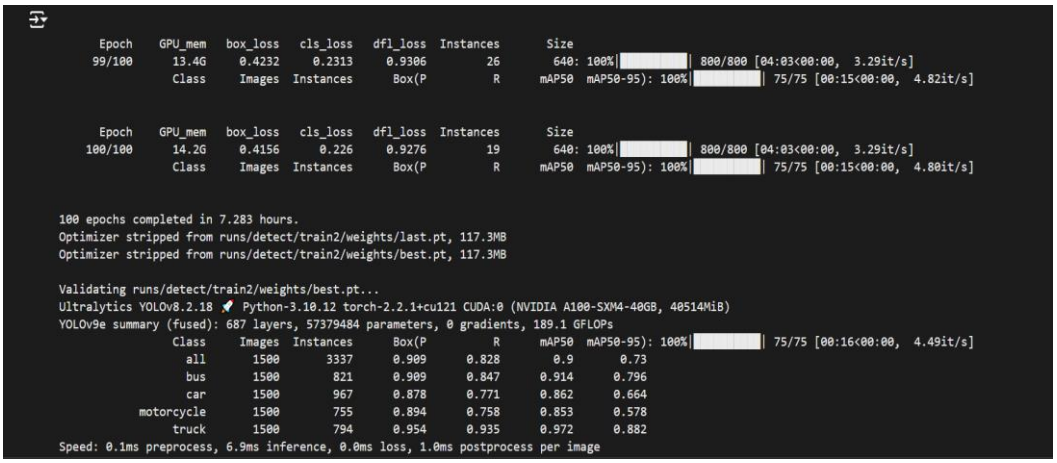
Gambar 5. *Balancing Class Dataset*



Gambar 6. Jumlah Dataset Yang Digunakan

3.3 *Modelling*

Tahap modelling merupakan training phase menggunakan Google Colab yang dapat dilihat pada gambar 7. Pada proses training konfigurasi menggunakan YOLOv9e (YOLOv9 larger model dengan FLOPs 192,5), 100 epochs, ukuran citra (imgz) 640, 10 batch, 8 workers, dan 10 close mosaic. Dengan 10 batch yang diterapkan pada dataset berjumlah 10000, maka dalam 1 epochs dibutuhkan 10000/10 = 1000 langkah, sehingga seluruh proses pelatihan membutuhkan total 10 juta langkah.

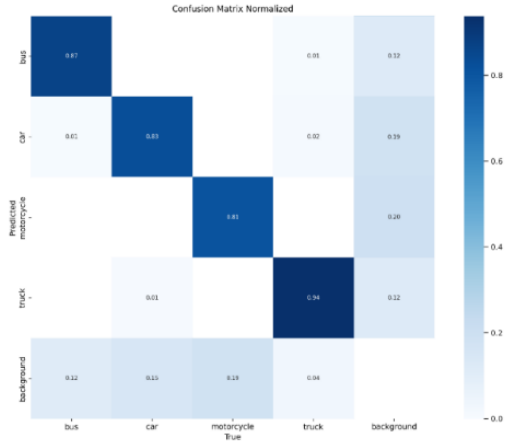


Gambar 7. Proses *Training* dan *Summary Custom* Dataset

3.4 Data *Evaluation*

hasil analisis model pada tugas klasifikasi citra *multi-class* yang disajikan dengan

menggunakan ouput tahap modelling yaitu *confusion matrix normalized* di gambar 8. Analisis dilakukan dengan mengukur *accuracy*, *precision*, *recall*, dan *F1-score* serta mAP pada evaluasi dari tugas deteksi *object*.



Gambar 8. Hasil *Confusion Matrix*

Dari *confusion matrix normalized* pada *class bus* tampak bahwa nilai TP_{bus} 0,87 FP_{bus} 0,01 yang berasal dari *miss class* pada *class car* FN_{bus} 0,01 dari *miss class* pada *class truck* serta TN_{bus} sebesar 0,11 yang berasal dari pengurangan nilai 1 dengan jumlah dari TP, FP, dan FN. Dengan persamaan 1 hingga 4 didapatkan perhitungan untuk *class bus*:

Accuracy pada persamaan 1 merupakan jumlah *class instance* yang diklasifikasikan dengan benar dari semua *class instance* yang ada di *train* set.

$$accuracy_{bus} = \frac{TP_{bus} + TN_{bus}}{TP_{bus} + FP_{bus} + TN_{bus} + FN_{bus}} = \frac{0{,}87 + 0{,}11}{0{,}87 + 0{,}01 + 0{,}11 + 0{,}01} = 0{,}98 \quad (1)$$

Precision pada persamaan 2 memberikan indikasi seberapa akurat model mampu mengidentifikasi *class instance* dalam *positive*. *Precision* yang tinggi akan berdampak baik dengan lebih sedikitnya model dalam kondisi *false positive*. Sangat penting untuk mempertimbangkan *Precision* bersamaan dengan *recall* (sensitivitas) dalam evaluasi model.

$$precision_{bus} = \frac{TP_{bus}}{TP_{bus} + FP_{bus}} = \frac{0{,}87}{0{,}87 + 0{,}01} = 0{,}9886 \quad (2)$$

Recall pada persamaan 3 menilai seberapa efektif model dalam mengidentifikasi semua *instance positive* pada dataset. Parameter ini memberikan gambaran tentang seberapa baik model dapat menemukan semua kondisi *positive* pada model.

$$recall_{bus} = \frac{TP_{bus}}{TP_{bus} + FN_{bus}} = \frac{0{,}87}{0{,}87 + 0{,}01} = 0{,}9886 \quad (3)$$

F1-score pada persamaan 4 bertujuan untuk mengevaluasi performa model klasifikasi [21]. ketika proporsi *class positive dan class negative* tidak seimbang maka *matrix* ini akan menjadi penyeimbang.

$$F1 - Score_{bus} = \frac{2\ x\ Precision\ _{bus} x\ Recall_{bus}}{Precision_{bus} + Recall_{bus}} = \frac{2x0{,}9886x0{,}9886}{0{,}9886 + 0{,}9886} = 0{,}9886 \quad (4)$$

Perhitungan klasifikasi citra pada setiap *class* dan rata-rata seluruh *class* disajikan pada tabel 3.

Evaluasi ini merujuk pada tugas klasifikasi dari model yang dilatih.

Tabel 3. Evaluasi Model Untuk Klasifikasi Berdasarkan *Confusion Matrix Normalized*

Class	TP	FP	FN	TN	Accuracy	Precision	Recall	F1-Score
Bus	0,87	0,01	0,01	0,11	0,98	0,9886	0,9886	0,9886
Car	0,83	0,01	0,03	0,13	0,96	0,9881	0,9652	0,9765
Motorcycle	0,81	0,00	0,00	0,19	1,00	1,00	1,00	1,00
Truck	0,94	0,03	0,01	0,02	0,96	0,9691	0,9895	0,9792
Macro Avg	-	-	-	-	0,975	0,98645	0,985825	0,986075

Berdasarkan dokumentasi Ultralytics dalam deteksi *object* sendiri merupakan hubungan antara *object* actual dan *object* prediksi dapat bersifat *many-to-many*. Evaluasi dapat menggunakan *matrix* khusus seperti *Intersection over Union* (IoU) atau *Mean Average Precision* (mAP). IoU mengukur tumpang tindih antara kotak pembatas atau bounding box yang diprediksi dan kotak pembatas hasil dari *labelling* pada dataset, mengevaluasi keakuratan lokalisasi *object*. mAP memperluas konsep AP dengan menghitung nilai rata-rata AP di beberapa *class object* dan berguna dalam skenario deteksi *object multi-class* untuk memberikan evaluasi komprehensif terhadap performa model. Tabel 4 menunjukan evaluasi hasil *training* model dalam mendeteksi *object* setiap 20 *epochs*, sementara tabel 5 menggambarkan hasil *training* model dalam mendeteksi *object* untuk setiap dan seluruh *class*.

Tabel 4. Hasil *Training* Model (Deteksi) / 20 *Epochs*

Epochs	Precision	Recall	mAP50
20	0,817	0,729	0,794
40	0,887	0,778	0,869
60	0,886	0,823	0,894
80	0,904	0,831	0,903
100	0,912	0,820	0,898

Tabel 5. Hasil *Training* Model (Deteksi) Setiap & Seluruh *Class*

Class	Precision	Recall	mAP50
Bus	0,909	0,847	0,914
Car	0,878	0,771	0,862
Motorcycle	0,894	0,758	0,853
Truck	0,954	0,935	0,972
All	**0,909**	**0,828**	**0,9**

$$F1 - Score = \frac{2\ x\ Precision\ x\ Recall}{Precision + Recall} = \frac{2\ x\ 0{,}909\ x\ 0{,}828}{0{,}909 + 0{,}828} = 0{,}86 \quad (4)$$

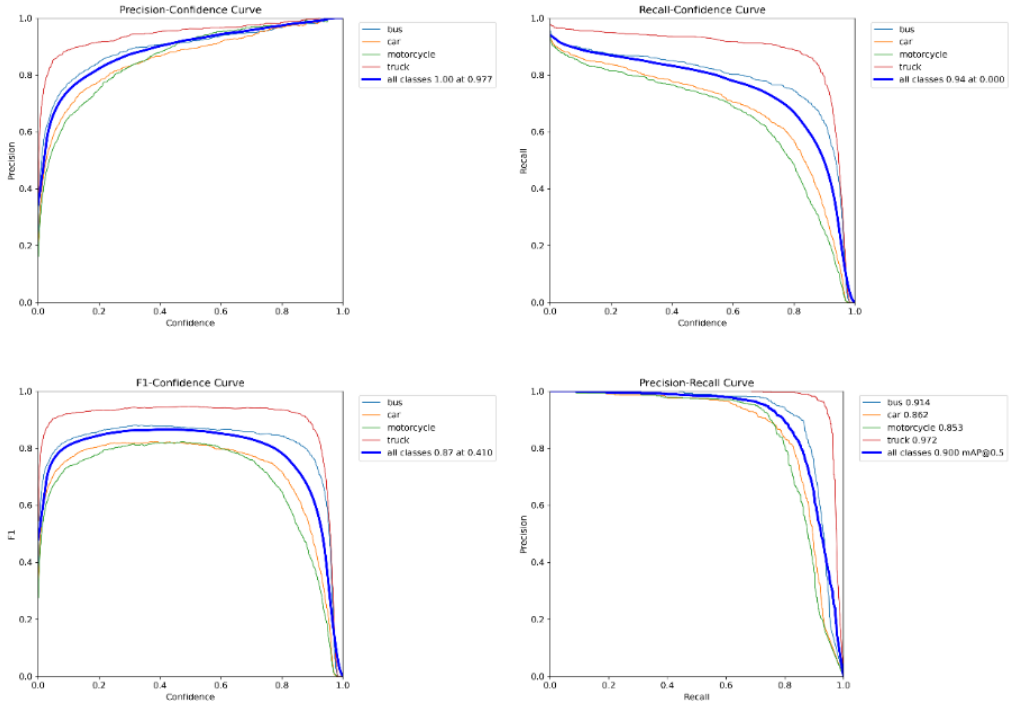
Validasi yang didapat dari nilai hasil *training* untuk keseluruhan *class* yaitu *precision* 0,909 *Recall* 0,828 mAP50 0,9 dan *F1 Score* dari model ini untuk mendeteksi adalah 0,867. Hasil model cukup baik dengan nilai mAP yang dihitung dari batas ambang IoU 50 sebesar 0,9 dan presisi yang menunjukan keakuratan *object* yang terdeteksi sebesar 0,909. Berikut tabel 6 memberikan tinjauan hasil evaluasi keseluruhan model pada tugas deteksi.

Tabel 6. Hasil Evaluasi Keseluruhan Model (Deteksi)

No	Matrix	Nilai
1.	mAP	0,9
2.	Precision	0,909
3.	Recall	0,828
4.	F1 Score	0,867

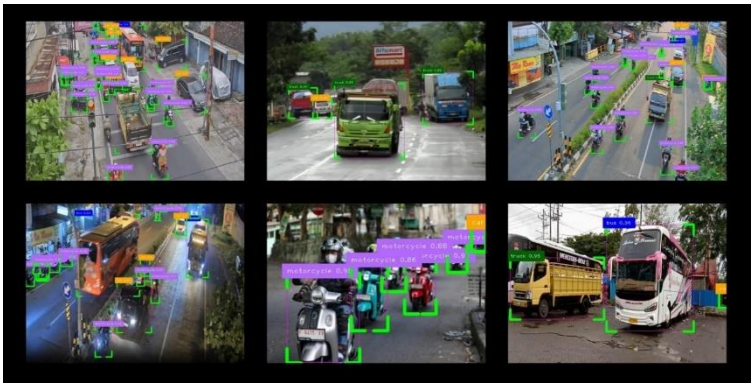
Kurva kinerja model disajikan dalam Gambar 9. Dari analisis gambar, kurva *precision* mencapai nilai puncak 1 pada tingkat kepercayaan 0,977 *recall* memperoleh nilai 0,94 pada tingkat kepercayaan 0, *F1-Score* mencapai 0,87 pada kepercayaan 0,41, dan *precision-recall*

mencapai 0,900 pada 0,5 mAP. *Class motorcycle* memiliki nilai evaluasi dibawah seluruh *class*, sedangkan *class truck* mendapatkan nilai paling baik.



Gambar 9. Kurva Kinerja Model

Hasil pendeteksian kendaraan dengan menggunakan model ini menunjukan akurasi dan presisi yang baik meskipun dalam kondisi *traffic* yang tinggi ditunjukan pada Gambar 10 yang merupakan citra diluar dataset (*train set*, *valid set*, dan *test set*).



Gambar 10. Hasil Pendeteksian Kendaraan Dengan Model Latih

3.5 Model *Deployment*

Dalam menerapkan model yang telah dilatih untuk mendeteksi kendaraan dilakukan pengkodean yang mencakup *detector*, *tracker*, dan *counter*. *Detector* bekerja setelah model memprediksi *bounding box* dalam *frame* dari CCTV, hasil prediksi diproses untuk mengidentifikasi *koordinat bounding box* dan jenis *class*. Hanya jenis *class* yang relevan yang ditambahkan ke daftar *mylist* untuk dilacak lebih lanjut. Pada setiap *frame*, *tracker* menerima daftar *bounding box* dari *detector* dan menghitung centroid dari setiap kendaraan. *Tracker* memeriksa apakah centroid yang baru dihitung berada dalam jarak tertentu dari centroid yang

telah dilacak sebelumnya untuk mendapatkan *id* yang konsisten dari *frame* ke *frame*. *Counter* menentukan apakah kendaraan berada di dalam area ROI yang didefinisikan membedakan antara kendaraan dalam kategori "*watch billboard*" dan "*no watch billboard*". Gambar 11 menunjukan potongan kode ketiga komponen yang digunakan.

```
results=model.predict(frame)
a=results[0].boxes.data
px = pd.DataFrame(a.cpu()).astype("float")
mylist = []
list=[]
for index,row in px.iterrows():
    x1=int(row[0])
    y1=int(row[1])
    x2=int(row[2])
    y2=int(row[3])
    d=int(row[5])
    c=class_list[d]
    confidence = row[4]
    if 'car' in c or 'motorcycle' in c or 'truck' in c or 'bus' in c:
        mylist.append([x1,y1,x2,y2,c,confidence])
bbox_idx=tracker.update(mylist)
```

```
import math
class Tracker:
    def __init__(self):
        self.center_points = {}
        self.id_count = 0
    def update(self, objects_rect):
        # Objects boxes and ids
        objects_bbs_ids = []
        for rect in objects_rect:
            x1, y1, x2, y2, object_type, confidence = rect
            cx = (x1 + x2) // 2
            cy = (y1 + y2) // 2
            same_object_detected = False
            for id, pt in self.center_points.items():
                dist = math.hypot(cx - pt[0], cy - pt[1])
                if dist < 35:
                    self.center_points[id] = (cx, cy)
                    objects_bbs_ids.append([x1, y1, x2, y2, id, object_type, confidence])
                    same_object_detected = True
                    break
            if same_object_detected is False:
                self.center_points[self.id_count] = (cx, cy)
                objects_bbs_ids.append([x1, y1, x2, y2, self.id_count, object_type, confidence])
                self.id_count += 1
        new_center_points = {}
        for obj_bb_id in objects_bbs_ids:
            x1, y1, x2, y2, _, _, _ = obj_bb_id
            cx = (x1 + x2) // 2
            cy = (y1 + y2) // 2
            object_id = obj_bb_id[4]
            center = (cx, cy)
            new_center_points[object_id] = center
        self.center_points = new_center_points.copy()
        return objects_bbs_ids
```

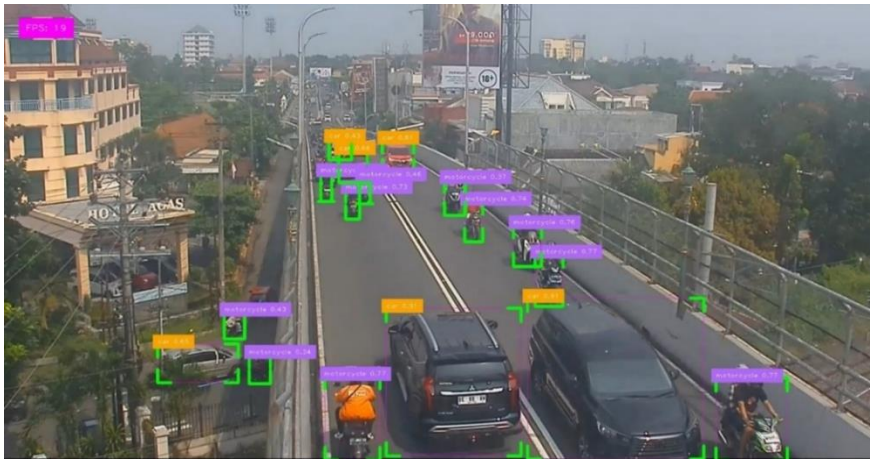
```
result= cv2.pointPolygonTest(np.array(area1, np.int32),((cx,cy)),False)

if result >=0:
    vehicleWatch[id1]=(cx,cy)

if id1 in vehicleWatch:
    result1=cv2.pointPolygonTest(np.array(area2,np.int32),((cx,cy)),False)
    if result1 >=0:
        cv2.circle(frame,(cx,cy),4,(0,0,255), 1)
        cv2.rectangle(frame,(x3,y3),(x4,y4),(255,255,255),2)
        cvzone.putTextRect(frame,f'{id1}',(x3,y3),1,1)
        if vehicleWatchCounter.count(id1)==0:
            vehicleWatchCounter.append(id1)
```

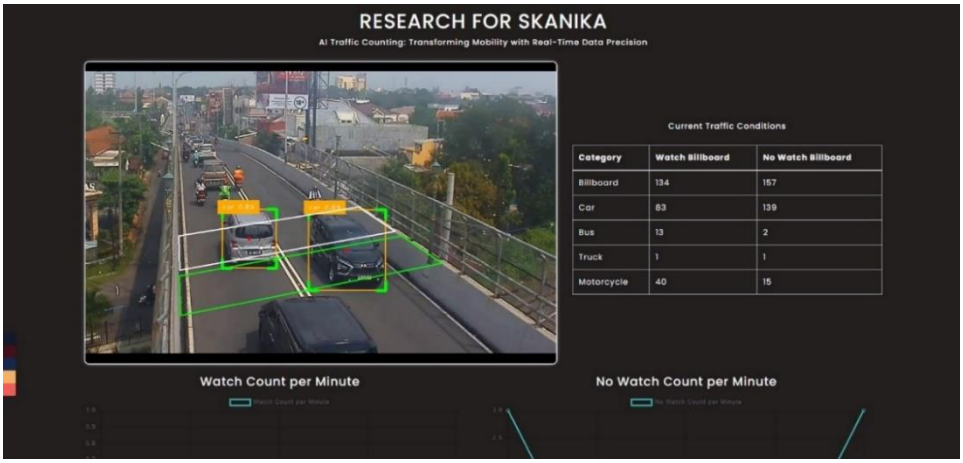
Gambar 11. Kode Program *Detector* (kiri), *Tracker* (tengah), *Counter* (kanan)

Sistem pada penelitian ini menggunakan spesifikasi perangkat keras processor 6-Core (12 CPUs) 3.6GHz, 16GB RAM 3200 MHz, serta GPU NVIDIA GeForce GTX 1660 dengan VRAM 6GB. Agar YOLO dapat beroperasi di lingkungan GPU yang optimal, model diimplementasikan menggunakan cuDNN dan CUDA. Dari integrasi ini sistem mendapatkan rata-rata rentan 18-21 FPS. FPS ini didapat dari perhitungan dengan membandingkan waktu saat ini dengan waktu sebelumnya untuk setiap iterasi pengambilan *frame*. Meskipun FPS masih belum optimal namun masih dapat memperlihatkan proses *real-time* cukup baik. Gambar 12 menunjukan hasil deteksi kendaraan sebelum metode pelacakan berbasis centroid di terapkan.



Gambar 12. Deteksi Kendaraan Pada CCTV Video Sebelum Pelacakan Berbasis Centroid

Hasil deteksi dan perhitungan kendaraan menunjukkan *bounding box* dan centroid yang di-*overlay* pada *frame* video, dengan anotasi visual yang menunjukkan jenis kendaraan dan tingkat kepercayaannya dari model YOLOv9 yang telah dilatih serta penerapannya pada metode pelacakan berbasis centroid dan area ROI serta sudah diterapkan pada *web platform* ditunjukan pada gambar 13.



Gambar 13. Deteksi dan Perhitungan Kendaraan Pada CCTV Video Menggunakan Pelacakan Berbasis Centroid

Tabel 7 menunjukan tahap akhir model *deployment* melalui pengujian manual sistem dilakukan untuk semua *class* dengan 1 area ROI pada waktu siang hari dan malam hari dengan data *streaming* CCTV *billboard* yang direkam selama 1 menit untuk mempermudah perhitungan manual, sedangkan dengan 2 area ROI ditunjukan pada tabel 8. Keakuratan sistem dihitung secara manual dengan menggunakan persamaan 5.

$$Akurasi_{System} = 100\% - (\frac{Deteksi\ Error}{Aktual}\ x100\%) \quad (5)$$

Tabel 7. Pengujian Sistem Perhitungan Secara Manual 1 Area ROI Dalam 1 Menit

Kondisi	Class	Aktual	Terdeteksi	Deteksi *Error*	Akurasi
Siang 	Bus	2	2	0	100%
	Car	14	14	0	100%
	Motorcycle	23	21	2	91,3%
	Truck	0	0	0	-
	All Class	**39**	**37**	**2**	**94,9%**
Malam 	Bus	1	1	0	100%
	Car	18	17	1	94,4%
	Motorcycle	30	27	3	90%
	Truck	1	1	0	100%
	All Class	**50**	**46**	**4**	**92%**

Tabel 8. Pengujian Sistem Perhitungan 2 Secara Manual Area ROI Dalam 1 Menit

Kondisi	Class	Aktual	Terdeteksi	Deteksi *Error*	Akurasi
Siang 	Bus	1	1	0	100%
	Car	26	23	3	88,5%
	Motorcycle	35	32	3	91,4%
	Truck	1	1	0	100%
	All Class	**63**	**57**	**6**	**90,5%**
Malam 	Bus	1	1	0	100%
	Car	15	13	2	86,6%
	Motorcycle	30	25	5	83,3%
	Truck	4	4	0	100%
	All Class	**50**	**43**	**7**	**86%**

Berdasarkan tabel 7 dan tabel 8 dapat dilihat pada *class car* dan *class motorcycle* mengalami beberapa deteksi *error*. Hasil pengamatan langsung saat perhitungan manual, pada *class car* keseluruhan *error* cenderung mengalami FN, sedangkan *class motorcycle* mengalami

FN dan sebagian kecil FP. Faktor penyebab kesalahan ini dapat dimungkinkan dari model yang dilatih, kurangnya data kendaraan pada kecepatan tinggi, atau pengaruh kebutuhan *source* yang digunakan untuk komputasi pada sistem terlebih pada ROI yang lebih dari satu.

Dari kedua tabel ini juga didapatkan nilai perhitungan akurasi rata-rata sistem secara manual dari total *All Class* sebesar 90,85%. Nilai ini cukup mendekati dari nilai mAP hasil *training* model dari tugas deteksi *object* yaitu 90%, namun memiliki selisih yang lebih pada perhitungan klasifikasi dengan nilai akurasi 97,5%. Tampak juga pada klasifikasi *class motorcycle* memiliki nilai akurasi 100%, berbeda jauh dengan dengan nilai mAP nya yang didapat pada tugas deteksi yaitu sebesar 85,3%. Besar kemungkinan bahwa dataset pada *class motorcycle* mendapatkan pengaruh dari *background* yang ditunjukan di gambar 8 *confusion matrix normalized* sebelumnya. Tugas klasifikasi berbeda dengan tugas deteksi *object*. Dalam tugas deteksi terdapat pengaruh dari nilai FP dan FN yang muncul disetiap *class* pada kolom dan baris *background*. *Background* atau gambar latar belakang ini bukan milik *class* manapun, menjadi citra tanpa *object* yang menarik serta memiliki *.txt file* terkait pada folder label. *Background* dapat terjadi dengan adanya beberapa *bounding box* yang menunjuk ke target yang sama, satu *bounding box* yang menunjuk ke beberapa target dari area tumpang tindih, atau sama sekali tidak mendeteksi apapun.

4. KESIMPULAN DAN SARAN

Penelitian ini berhasil mengembangkan sistem perhitungan jumlah kendaraan berdasarkan jenis kendaraan menggunakan algoritma YOLO secara *real-time*. Dalam penelitian ini sistem dijalankan pada spesifikasi perangkat keras processor 6-Core (12 CPUs) 3.6GHz, 16GB RAM 3200 MHz, serta GPU NVIDIA GeForce GTX 1660 dengan VRAM 6GB yang terintegrasi dengan cuDNN dan CUDA menunjukkan hasil FPS rata-rata yang didapat belum optimal pada rentan 18-21 FPS. Spesifikasi ini memungkinkan munculnya FN atau perhitungan kendaraan terlewatkan, khususnya pada kondisi kendaraan yang memiliki kecepatan sangat tinggi. Peningkatan spesifikasi perangkat keras dapat dilakukan pada penelitian selanjutnya untuk meningkatkan kecepatan komputasi dan efisiensi sistem.

Evaluasi model YOLOv9 untuk deteksi *object* yang dilatih pada penelitian ini menghasilkan nilai mAP sebesar 90% *precision* sebesar 90,9% *recall* sebesar 82,8% dan *F1-score* sebesar 86,7%. Nilai ini sedikit lebih rendah dengan evaluasi model untuk tugas klasifikasi berdasarkan *confusion matrix normalized* yang dihasilkan dengan *accuracy* 97,5%. Hal ini disebabkan perilaku *object* latar belakang atau *background* pada deteksi memiliki nilai FP dan FN, yang berarti area tumpang tindih dari *bounding box* mengalami kebingungan atau tidak ada sama sekali dalam melakukan setiap prediksi. Kemunculan nilai ini menunjukan dibutuhkan keseimbangan yang lebih lagi dari kumpulan data atau melakukan penambahkan *class-class* yang di identifikasikan atau berpengaruh terhadap *background*, konsistensi label pada data *exploration*, dan *hyperparameter* yang tepat pada *train* model untuk penelitian selanjutnya.

Penelitian ini mendapatkan akurasi 90,85% pada pengujian perhitungan manual seluruh *class* yang terdeteksi di 1 atau 2 area ROI sistem di kondisi siang dan malam hari. Pengujian pada jumlah area ROI yang lebih banyak dan kondisi yang berbeda diperlukan pada penelitian selanjutnya untuk meningkatkan validitas dari model yang telah diterapkan.

Sistem yang dikembangkan dari penelitian ini cukup mampu mendeteksi dan melacak berbagai jenis kendaraan dengan tingkat akurasi tinggi meskipun dalam *traffic* yang tinggi, sehingga sistem ini juga dapat dimanfaatkan untuk analisis lalu lintas dan perencanaan periklanan. Pada penelitian selanjutnya eksplorasi penggunaan algoritma lain yang dapat bekerja secara sinergis dengan YOLO dapat dilakukan untuk meningkatkan performa sistem deteksi secara menyeluruh dan terintegrasi dengan sistem manajemen lalu lintas yang lebih besar untuk memberikan manfaat yang lebih luas dalam bidang pemantauan dan pengelolaan lalu lintas kota.

DAFTAR PUSTAKA

[1] B. Lavega Adiputra, P. Christin Harnita, P. Hergianansari, and I. Komunikasi Universitas Satya Kencana, "Penggunaan Media Baliho oleh Puan Maharani Menjelang Pemilu Indonesia 2024 di Era 5.0," *J. Neo Soc.*, vol. 8, no. 3, pp. 189–203, 2023.

[2] W. Kustiawan, K. Rizky Ramadhani, S. Valentina Damanik, and A. Muharramsyah, "Pengaruh Iklan Politik Dalam Mengambil Aspirasi Rakyat," *SIBATIK J. J. Ilm. Bid. Sos. Ekon. Budaya, Teknol. dan Pendidik.*, vol. 1, no. 8, pp. 1371–1380, 2022.

[3] W. Swastika, A. Kurniawan, and H. Setiawan, "Deteksi dan Klasifikasi Merek Mobil untuk Penentuan Iklan Billboard Menggunakan Convolution Neural Network," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 7, no. 4, pp. 701–708, 2020.

[4] G. P. A. Dipura, F. Amanda, M. R. Firmansyah, M. R. Rizky, and M. N. K. Jamal, "Teknologi Komputer Vision dalam Kamera Pengawas," *Karimah Tauhid*, vol. 3, no. 3 SE-Articles, pp. 3754–3760, Mar. 2024.

[5] I. Maulana, N. Rahaningsih, and T. Suprapti, "Analisis Penggunaan Model YOLOv8 (You Only Look Once) Terhadap Deteksi Citra Senjata Berbahaya," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 7, no. 6, pp. 3621–3627, 2023.

[6] J. S. W. Hutauruk, T. Matulatan, and N. Hayaty, "Deteksi Kendaraan secara Real Time menggunakan Metode YOLO Berbasis Android," *J. Sustain. J. Has. Penelit. dan Ind. Terap.*, vol. 9, no. 1, pp. 8–14, 2020.

[7] M. Ghulam, A. Binuri, M. Amirul, T. Haryanti, and A. Peningkatan, "Penerapan Algoritma Yolo V7 Sebagai Dekteksi Kecelakaan Pada Lalu Lintas," *Prosiding-Seminar Nas. Teknol. Inf. Ilmu Komput.*, vol. 2, no. 1, pp. 35–41, 2023.

[8] Randy Moh Yusup, Aldof Faris Anugrah, D. D. Muslimah, S. M. W. N. Permana, and Shindi Yuliani, "Pendeteksi Objek Menggunakan OpenCV dan Metode YOLOv4-Tiny Untuk Membantu Tuna Netra," *J. Comput. Sci. Inf. Technol.*, vol. 1, no. 2, pp. 59–68, 2024.

[9] E. S. Manapa, E. A. M. Sampetoding, T. W. Sagala, and H. R. Taluay, "Ulasan Penelitian Sistem Operasi Waktu Nyata di Indonesia," *J. Dyn. Saint*, vol. 5, no. 2, pp. 973–979, 2021.

[10] A. A. Saifullah, E. Hegarini, and A. Wardijono, "Sistem Penghitung Jumlah Kendaraan dan Pendeteksi Kecepatan pada Ruas Jalan Menggunakan Metode Haar Cascade Classifie," *J. Ilm. Komputasi*, vol. 14, no. 1, pp. 95–100, 2024.

[11] A. A. M. Suradi, M. F. Rasyid, and N. Nasaruddin, "Sistem Perhitungan Jumlah Kendaraan Berbasis Computer Vision," *Pros. Semin. Ilm. Sist. Inf. Dan Teknol. Inf.*, vol. XI, no. 1, pp. 89–97, 2022.

[12] N. J. Hayati, D. Singasatia, and M. R. Muttaqin, "Object Tracking Menggunakan Algoritma You Only Look Once (YOLO)v8 untuk Menghitung Kendaraan," *Komputa J. Ilm. Komput. dan Inform.*, vol. 12, no. 2, pp. 91–99, 2023.

[13] H. M. Zayani *et al.*, "Unveiling the Potential of YOLOv9 through Comparison with YOLOv8," *IJISAE*, vol. 12, no. 3, pp. 2845–2854, 2024.

[14] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information," 2024.

[15] M. R. Sholahuddin, F. Atqiya, S. R. Wulan, M. Harika, S. Fitriani, and Y. Sofyan, "Implementasi Sistem Identifikasi Senjata Real Time Menggunakan YOLOv7 dan Notifikasi Chat Telegram," *J. Inf. Syst. Res.*, vol. 4, no. 2, pp. 598–606, 2023.

[16] W. S. Sari and C. A. Sari, "Klasifikasi Bunga Mawar Menggunakan Knn Dan Ekstraksi Fitur Glcm Dan Hsv," *Skanika*, vol. 5, no. 2, pp. 145–156, 2022.

[17] Muhammad Nur Ihsan Muhlashin and A. Stefanie, "Klasifikasi Penyakit Mata Berdasarkan Citra Fundus Menggunakan YOLO V8," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 7, no. 2, pp. 1363–1368, 2023.

[18] W. Wijiyanto, A. I. Pradana, and S. Sopingi, "Perbandingan Data Untuk Memprediksi Ketepatan Studi Berdasarkan Atribut Keluarga Menggunakan Machine Learning," *JIKA (Jurnal Inform.*, vol. 8, no. 2, pp. 221–228, Apr. 2024, Accessed: May 26, 2024.

[19] G. Zeng, "On the confusion matrix in credit scoring and its analytical properties," *Commun. Stat. - Theory Methods*, vol. 49, no. 9, pp. 2080–2093, May 2020.
[20] S. Meldiyana, I. S. Fathina, R. Yuner, S. Rachmat, and M. Harika, "Line Crossing Detector System Pada Real-Time Situational Awareness Dengan Menggunakan Spatial Sample Difference Consensus," *JTT (Jurnal Teknol. Ter.*, vol. 8, no. 1, p. 43, 2022.
[21] H. Tantyoko, D. K. Sari, and A. R. Wijaya, "Prediksi Potensial Gempa Bumi Indonesia Menggunakan Metode Random Forest Dan Feature Selection," *IDEALIS Indones. J. Inf. Syst.*, vol. 6, no. 2, pp. 83–89, 2023.