Perhitungan Kendaraan Menggunakan Yolo Dari Ultralytics Untuk Sistem Pemantauan Lalu Lintas

Rona Mahendra[1*], Nibras Faiq Muhammad [2], Afu Ichsan Pradana [3]

[1]Fakultas Ilmu Komputer Universitas Duta Bangsa Surakarta.
Jl. Bhayangkara No.55, Tipes, Kec. Serengan, kota Surakarta, Jawa Tengah 57154
[1*]220103136@mhs.udb.ac.id

[2]Fakultas Ilmu Komputer Universitas Duta Bangsa Surakarta.
Jl. Bhayangkara No.55, Tipes, Kec. Serengan, kota Surakarta, Jawa Tengah 57154
[2]nibras_faiqmuhammad@udb.ac.id

[3]Fakultas Ilmu Komputer Universitas Duta Bangsa Surakarta.
Jl. Bhayangkara No.55, Tipes, Kec. Serengan, kota Surakarta, Jawa Tengah 57154
[3]afu_ichsan@udb.ac.id

***Abstrak*— Pertumbuhan volume kendaraan di perkotaan memerlukan sistem pemantauan otomatis seperti Intelligent Transportation System (ITS). Namun, model deep learning sering menghadapi kendala ketidakseimbangan kelas (class imbalance) ekstrem yang menurunkan akurasi objek minoritas. Penelitian ini menerapkan arsitektur ringan YOLO26s dengan strategi random oversampling untuk mendeteksi empat kelas kendaraan: Bus, Mobil, Sepeda Motor, dan Truk. Dataset gabungan sejumlah 19.389 citra bersumber dari UA-DETRAC dan Roboflow, lalu dilatih memakai konfigurasi Distributed Data Parallel (DDP) pada dua GPU NVIDIA Tesla T4. Hasil pengujian membuktikan teknik oversampling meningkatkan performa model secara signifikan tanpa memicu overfitting. Nilai mAP@0.5 global meningkat dari 0,982 menjadi 0,993, dan mAP@0.5:0.95 naik dari 0,842 menjadi 0,911. Peningkatan krusial terjadi pada Recall kelas Truk yang melonjak dari 0,93 menjadi 0,98. Model juga mempertahankan efisiensi tinggi dengan kecepatan inferensi mencapai 85,51 FPS (11,69 ms), sehingga sangat ideal untuk implementasi pemantauan lalu lintas secara real-time.**
Kata kunci— Class Imbalance, Deteksi Kendaraan, Intelligent Transportation System, Random Oversampling, YOLO26s.

***Abstract*— The rapid growth of motor vehicle volume in urban areas triggers traffic congestion, necessitating automated monitoring solutions such as the Intelligent Transportation System (ITS). However, deep learning-based object detection models frequently encounter extreme class imbalance challenges, which degrade detection accuracy for minority classes. This study proposes the implementation of the lightweight YOLO26s architecture integrated with a random oversampling strategy to detect four vehicle categories: Bus, Car, Motorcycle, and Truck. A merged dataset comprising 19,389 images was constructed from UA-DETRAC and Roboflow sources. Training was executed using Distributed Data Parallel (DDP) configuration on two NVIDIA Tesla T4 GPUs. Experimental results demonstrate that the oversampling technique successfully enhances model performance without inducing overfitting. The global mAP@0.5 score increased from 0.982 to 0.993, and the mAP@0.5:0.95 rose from 0.842 to 0.911. The most critical improvement was observed in the Truck class Recall, which surged from 0.93 to 0.98. Furthermore, the model maintains high computational efficiency with an inference speed of 85.51 FPS (11.69 ms), making it highly ideal for real-time traffic stream monitoring deployment.**
***Keywords*— Class Imbalance, Intelligent Transportation System, Random Oversampling, Vehicle Detection, YOLO26s.**

I. PENDAHULUAN

Pertumbuhan volume kendaraan bermotor di area perkotaan memicu masalah kemacetan lalu lintas, peningkatan emisi, dan tingginya potensi kecelakaan [8]. Pendekatan konvensional dalam pemantauan lalu lintas seperti sensor induksi magnetik (inductive loop detector) terbukti tidak efektif karena memerlukan biaya instalasi yang mahal, rentan rusak, serta tidak mampu mengklasifikasikan jenis kendaraan secara spesifik [1]. Di sisi lain, pemanfaatan visi komputer (computer vision) berbasis pembelajaran mendalam (deep learning) kerap menghadapi kendala ketidakseimbangan data (class imbalance) di lapangan, di mana jumlah objek mayoritas seperti mobil dan sepeda motor jauh mendominasi dibandingkan kelas minoritas seperti bus dan truk, sehingga memicu bias prediksi pada model deteksi [2].

Sebagai solusi untuk mengatasi kendala tersebut, penelitian ini menerapkan arsitektur *one-stage detector* terbaru yaitu Ultralytics YOLO26s yang dikombinasikan dengan teknik rekayasa data *random oversampling*. Penggunaan YOLO26s dipilih karena membawa pembaruan arsitektur berupa sistem *end-to-end* tanpa komponen *non-maximum suppression* (NMS-free) dan kepala deteksi yang lebih ringan, serta dioptimasikan secara otomatis menggunakan pengoptimasi hibrida MuSGD (*Muon SGD*) untuk mempercepat kestabilan konvergensi pelatihannya [4]. Strategi *random oversampling* diintegrasikan secara khusus untuk menduplikasi jumlah kemunculan objek kelas bus dan truk pada data latih guna menetralisir bias

akibat *class imbalance* sebelum model memasuki tahap pelatihan [5].

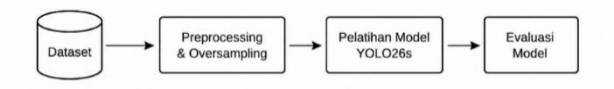
Evaluasi performa model ini dilakukan secara kuantitatif dan kualitatif menggunakan metrik *Precision*, *Recall*, *Mean Average Precision* (mAP@0.5 dan mAP@0.5:0.95), serta pengujian kecepatan inferensi dalam satuan *Frames Per Second* (FPS). Hasil eksperimen menunjukkan bahwa intervensi *oversampling* berhasil mendongkrak akurasi deteksi pada kelas minoritas tanpa mengorbankan performa kelas mayoritas. Selain itu, berkat struktur arsitektur YOLO26s yang efisien, model ini tidak hanya menghasilkan tingkat keyakinan (*confidence score*) yang tinggi pada deteksi multi-skala di jalan raya, melainkan juga mampu mencapai kecepatan pemrosesan yang sangat tinggi sehingga sangat ideal untuk mendukung kebutuhan sistem manajemen lalu lintas cerdas secara *real-time*.

II. METODOLOGI PENELITIAN

Bagian ini menguraikan tahapan sistematis yang dilakukan dalam penelitian, meliputi perancangan diagram alir penelitian, karakteristik dataset, teknik rekayasa data (*rebalancing* dan *oversampling*), arsitektur model YOLO26s, serta konfigurasi parameter proses pelatihan.

A. Alur Penelitian

Tahapan penelitian dirancang secara terstruktur untuk menjamin validitas hasil eksperimen. Alur komputasi dimulai dari pengumpulan dataset mentah, pembersihan dan pembagian partisi data (*rebalancing*), augmentasi data, penyeimbangan sebaran objek minoritas (*oversampling*), pelatihan model dengan *Automatic Mixed Precision* (AMP), hingga evaluasi metrik performa model.



Gambar 1 Alur Penelitian

B. Karakteristik Dataset dan Pembagian Partisi

Dataset yang digunakan dalam penelitian ini difokuskan pada visualisasi jenis kendaraan di jalan raya. Berdasarkan pemeriksaan awal terhadap karakteristik data asli (*dataset health check*) sebelum dilakukan intervensi penyeimbangan, tercatat total seluruh gambar sebanyak 10.905 gambar yang memiliki rasio persentase 88% *train* (9.543 gambar), 8% *validation* (909 gambar), dan 4% *test* (453 gambar) dan tercatat total seluruh objek (*instances*) sebanyak 83.612 objek. Distribusi kemunculan objek antar-kelas menunjukkan adanya ketimpangan yang signifikan (*class imbalance*), dengan rincian sebagai berikut:

Tabel 1. Perbandingan Dataset Sebelum dilakukan Teknik Oversampling

Kelas	Jumlah Instance	Persentase (%)
Bus	3.326	3,98%
Car	53.218	63,65%
Motorcycle	22.478	26,88%
Truck	4.590	5,49%
Total	83.612	100%

Kondisi sebaran data awal ini memperlihatkan dominasi yang sangat kuat dari kelas Mobil dan Sepeda Motor yang secara akumulatif menguasai lebih dari 90% dari keseluruhan dataset. Ketidakseimbangan yang ekstrem (*class imbalance*) ini memiliki risiko tinggi menyebabkan model mengalami bias spasial saat proses pelatihan, di mana bobot model cenderung mengabaikan fitur-fitur pembeda dari kelas kendaraan berdimensi besar (bus dan truk).

C. Pra-pemrosesan Data dan Penanganan Class Imbalance

Untuk meminimalkan risiko bias model dan meningkatkan validitas hasil evaluasi, dilakukan rekayasa data menggunakan teknik random oversampling yang dikombinasikan dengan augmentasi data adaptif khusus pada kelas minoritas (Bus dan Truk) di dalam folder train yang bersumber dari dataset UA-DETRAC [11].

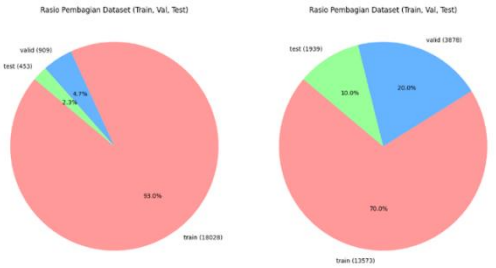
Setelah dilakukan intervensi rekayasa dataset, total seluruh objek (instances) meningkat signifikan menjadi 170.670 objek dengan sebaran kelas yang jauh lebih proporsional untuk memandu proses konvergensi bobot model:

Tabel 2. Perbandingan Setelah dilakukan Teknik Random Oversampling

Kelas	Jumlah Instance	Persentase (%)

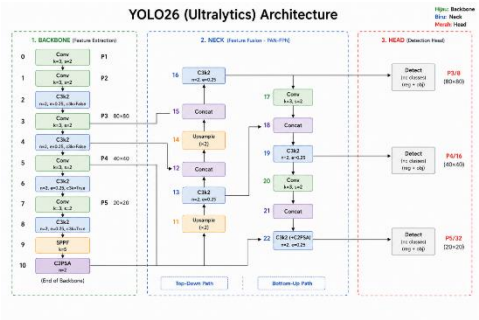
Bus	9.627	5.64%
Car	116.214	68.09%
Motorcycle	34.843	20.42%
Truck	9.986	5.85%
Total	170.670	100%

Setelah selesai dilakukan teknik random oversampling didapatkan dari 10.905 gambar menjadi 19.389 gambar dengan rasio 93% *train* (18.028 gambar), 4.7% *validation* (909 gambar), dan 2.3% *test* (453 gambar). lalu dikocok ulang (*shuffle*) dan dibagi kembali secara acak menggunakan proporsi standar ilmiah, yaitu 70% *train* (13.573 gambar), 20% *validation* (3.878 gambar), dan 10% *test* (1.939 gambar) agar dapat digunakan untuk pelatihan model sesuai proporsi standar ilmiah [10].



Gambar 2. Dataset Sebelum dan Sesudah Shuffle

D. Arsitektur YOLO



Gambar 3. Arsitektur model YOLO

Model yang digunakan dalam penelitian ini adalah YOLO26s, sebuah varian *small* dari keluarga YOLO26 yang diimplementasikan melalui framework Ultralytics versi 8.4.65. Arsitektur ini memiliki 260 lapisan dengan 9.950.960 parameter terlatih dan kebutuhan komputasi sebesar 22,5 GFLOPs, menjadikannya kandidat yang sesuai untuk aplikasi *real-time* pada perangkat keras terbatas [3].

Penelitian ini memanfaatkan empat komponen arsitektur utama dari YOLO26s untuk mengoptimalkan deteksi objek. Ekstraksi fitur multi-skala dilakukan secara simultan menggunakan Blok C3k2 yang menggabungkan *bottleneck cross-stage partial* dengan dua ukuran kernel berbeda, sementara agregasi konteks global dipercepat melalui operasi *max pooling* berulang pada modul SPPF. Untuk meningkatkan fokus model pada wilayah yang relevan, mekanisme *self-attention* diterapkan secara spasial melalui modul C2PSA. Terakhir, integrasi Kepala Deteksi Multi-Skala pada tiga resolusi berbeda (P3/8, P4/16, P5/32) memastikan model mampu mendeteksi kendaraan dari berbagai ukuran dan jarak pandang kamera secara efektif.

Model diinisialisasi dengan bobot pra-latih dari dataset ImageNet (*transfer learning*), dengan 696 dari 708 lapisan berhasil dipindahkan. Kepala deteksi dikonfigurasi ulang untuk mengakomodasi 4 kelas kendaraan yang menggantikan konfigurasi 80 kelas COCO asli.

E. Strategi Augmentasi Data

Strategi augmentasi pada penelitian ini dirancang secara komplementer untuk menghindari redundansi antara augmentasi yang telah diterapkan pada tahap prapemrosesan Roboflow dan augmentasi *on-the-fly* selama pelatihan. Tabel 3 merangkum konfigurasi augmentasi yang diterapkan.

Tabel 3. Konfigurasi Strategi Augmentasi Data

Teknik Augmentasi	Nilai
Mosaic	1.0
Copy-Paste	0.5
Mixup	0.1
Scale	0.5
HSV-Hue	0.015
Flip LR	0.0
Rotasi	0.0
HSV-Value	0.0
HSV-Saturation	0.0

Augmentasi *mosaic* menggabungkan empat gambar acak menjadi satu gambar komposit, secara efektif meningkatkan keragaman konteks latar belakang dan densitas objek per gambar. Strategi *copy-paste* menyalin objek dari satu gambar ke gambar lain pada posisi acak, yang terbukti efektif untuk meningkatkan performa pada kelas dengan representasi terbatas. *Mixup* menghasilkan gambar interpolasi linier antara dua sampel pelatihan, berkontribusi pada regularisasi model dan peningkatan kalibrasi kepercayaan prediksi.

F. Strategi Pelatihan Model

Pelatihan dilakukan pada platform Kaggle menggunakan dua unit GPU NVIDIA Tesla T4 (masing-masing 14.913 MiB VRAM) dengan konfigurasi pelatihan terdistribusi *Distributed Data Parallel* (DDP) melalui modul *torch.distributed.run*. Tabel 4 merangkum seluruh *hyperparameter* yang digunakan dalam sesi pelatihan utama.

Tabel 4. Konfigurasi Hyperparameter Pelatihan

Parameter	Nilai
Model	YOLO26s
Framework	Ultralytics 8.4.65
Input Size	640 × 640 piksel
Batch Size	64
Epochs	100
Optimizer	MuSGD (lr=0.01, momentum=0.9)
Patience	50 epoch
Hardware	2× NVIDIA Tesla T4 (DDP)
Cache	Disk
Workers	4
Pretrained	Ya (ImageNet)

Optimasi menggunakan MuSGD (*Muon SGD*) yang dipilih secara otomatis oleh framework melalui parameter *optimizer='auto'*. *Automatic Mixed Precision* (AMP) diaktifkan untuk mengurangi penggunaan memori GPU dan mempercepat komputasi tanpa kehilangan presisi numerik yang signifikan. Strategi *early stopping* dengan *patience* = 50 epoch diterapkan untuk menghentikan pelatihan jika tidak terjadi peningkatan mAP50-95 pada data validasi.

G. Metrik Evaluasi

Performa akhir model Ultralytics YOLO26s diukur secara kuantitatif menggunakan metrik deteksi objek standar industri yang meliputi *Precision* (P), *Recall* (R), *Mean Average Precision* (*mAP*), dan kecepatan inferensi dalam satuan *Frames Per Second* (FPS). *Precision* digunakan untuk mengukur rasio ketepatan kotak pembatas (*bounding box*) kendaraan hasil prediksi model, sedangkan *Recall* menilai kemampuan model dalam menemukan kembali seluruh objek nyata (*ground-truth*) yang tersebar di dalam dataset. Evaluasi akurasi spasial secara menyeluruh untuk seluruh kategori kendaraan (Bus, *Car, Motorcycle, Truck*) dihitung melalui akumulasi nilai *Average Precision* (AP) yang diintegrasikan berdasarkan kurva *Precision-Recall* (P-R), yang kemudian dirata-ratakan menjadi nilai *mAP* pada ambang batas *Intersection over Union* (IoU) sebesar 0,50 (mAP@0.5) serta rentang IoU ketat 0,50 hingga 0,95 (mAP@0.5:0.95). Selain aspek akurasi lokalisasi objek, kelayakan implementasi model pada sistem manajemen lalu lintas cerdas secara *real-time* dievaluasi berdasarkan nilai FPS yang dihitung dari total akumulasi waktu komputasi GPU yang meliputi fase pra-pemrosesan, inferensi jaringan, dan pasca-pemrosesan citra. Secara matematis, rumus pengukuran untuk setiap metrik evaluasi tersebut disajikan berturut-turut pada Persamaan di bawah ini:

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

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

$$Average\ Precision = \int_0^1 \quad P(R)\, dR \quad (3)$$

$$mAP = \frac{1}{C}\sum_{i=1}^{C} \quad AP_i \quad (4)$$

$$FPS = \frac{1000}{T_{preprocess}+T_{inference}+T_{postprocess}} \quad (5)$$

III. HASIL DAN PEMBAHASAN

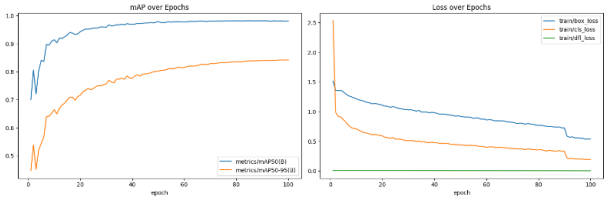
Bagian ini memaparkan hasil eksperimen pelatihan model Ultralytics YOLO26s pada dua skenario, yaitu skenario *baseline* (tanpa *oversampling*) dan skenario dengan penerapan teknik *random oversampling*. Analisis dilakukan terhadap dinamika pelatihan (kurva mAP dan *loss*), performa kuantitatif secara keseluruhan, performa per kelas, *confusion matrix*, serta pengujian penerapan model pada video lalu lintas secara *real-time*.

A. Hasil Pelatihan

a. Kurva mAP dan Loss

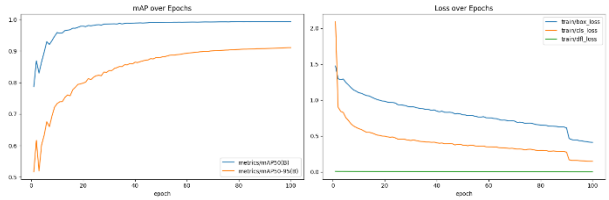
Kedua skenario dilatih selama 100 *epoch* dengan konfigurasi *hyperparameter* yang identik (Tabel 4), sehingga perbedaan performa dapat

diatribusikan secara langsung pada intervensi penyeimbangan data. Kurva mAP pada Gambar 3 dan Gambar 4 memperlihatkan tren kenaikan yang tajam pada 20 *epoch* pertama, kemudian melandai menuju kondisi konvergen, sementara kurva *loss* (*box loss*, *cls loss*, dan *dfl loss*) menurun secara konsisten tanpa indikasi *overfitting* yang berarti.



Gambar 3. Hasil pelatihan tanpa oversampling

Pada skenario tanpa *oversampling* (Gambar 3), nilai mAP@0.5 konvergen pada kisaran 0,98, sedangkan mAP@0.5:0.95 melandai pada kisaran 0,84. Pada skenario dengan *oversampling* (Gambar 4), kurva mAP@0.5 meningkat hingga mendekati 0,99 dan mAP@0.5:0.95 mencapai kisaran 0,91. Selain mencapai nilai akhir yang lebih tinggi, kurva pada skenario *oversampling* cenderung lebih halus dan stabil, yang mengindikasikan proses konvergensi bobot model yang lebih terarah berkat sebaran kelas yang lebih proporsional [7]. Penurunan *loss* yang lebih tajam pada *epoch* akhir juga sejalan dengan strategi penonaktifan augmentasi *mosaic* menjelang akhir pelatihan.



Gambar 4. Hasil pelatihan oversampling

b. *Perbandingan Performa Keseluruhan*

Perbandingan metrik evaluasi global untuk kedua skenario dirangkum pada Tabel 5. Penerapan *random oversampling* meningkatkan seluruh metrik dibandingkan *baseline*.

Tabel 5. Perbandingan Skenario

Skenario	Precision	Recall	mAP50	mAP50-95
Tanpa Oversampling	0,971	0,951	0,982	0,842
Dengan Oversampling	0,984	0,974	0,993	0,911

Berdasarkan Tabel 5, *Precision* meningkat dari 0,971 menjadi 0,984 (+1,3%) dan *Recall* meningkat dari 0,951 menjadi 0,974 (+2,4%). Kenaikan *Recall* yang lebih besar daripada *Precision* menunjukkan bahwa model dengan *oversampling* lebih jarang melewatkan objek nyata (*false negative* menurun), tanpa mengorbankan ketepatan prediksi. Peningkatan paling signifikan terlihat pada metrik mAP@0.5:0.95, yaitu dari 0,842 menjadi 0,911 (+6,9 poin atau +8,2%). Karena mAP@0.5:0.95 dihitung pada rentang ambang IoU yang ketat (0,50–0,95), kenaikan ini membuktikan bahwa *oversampling* tidak hanya memperbaiki keberhasilan klasifikasi, tetapi juga meningkatkan akurasi lokalisasi *bounding box* secara substansial. Sementara itu, mAP@0.5 yang sudah tinggi pada *baseline* (0,982) hanya naik tipis menjadi 0,993 karena mendekati titik jenuh (*saturation*).

B. *Analisi Performa Per Kelas*

Untuk menilai dampak *oversampling* terhadap permasalahan *class imbalance*, performa dianalisis pada tingkat per kelas sebagaimana disajikan pada Tabel 6.

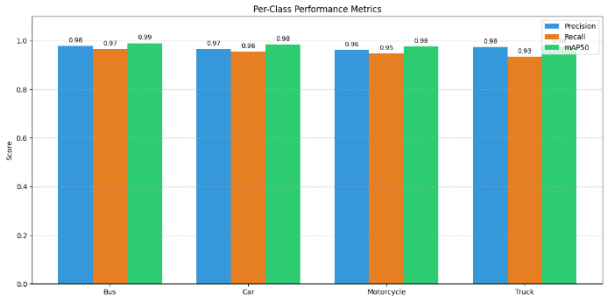
Tabel 6. Perbandingan Rata-Rata Performa

Kelas	Baseline	Oversampling
Bus (minoritas)	0,903	0,965
Truck (minoritas)	0,875	0,943
Motorcycle (menengah)	0,767	0,83
Car (mayoritas)	0,824	0,903

Tabel 6 memperlihatkan bahwa seluruh kelas mengalami peningkatan performa setelah penerapan *oversampling*. Kelas minoritas yang menjadi sasaran utama intervensi, yaitu Bus dan Truck, masing-masing meningkat sebesar +0,062 (0,903 → 0,965) dan +0,068 (0,875 → 0,943). Hal ini mengonfirmasi bahwa duplikasi dan augmentasi adaptif pada objek kelas minoritas berhasil menyediakan ragam fitur yang cukup bagi model untuk mengenali kendaraan berdimensi besar yang sebelumnya kurang terwakili.

Yang menarik, peningkatan tidak terbatas pada kelas minoritas: kelas Car (mayoritas) justru mencatat kenaikan tertinggi (+0,079) dan kelas Motorcycle (menengah) naik +0,064. Temuan ini

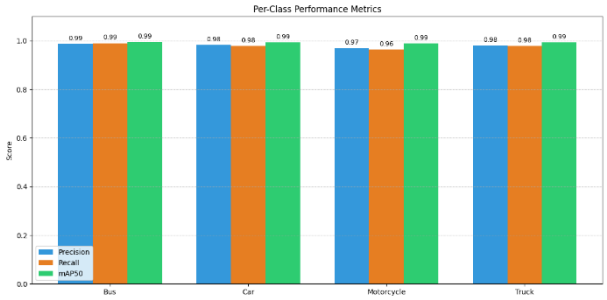
menunjukkan bahwa penyeimbangan distribusi data memberikan efek positif menyeluruh dengan berkurangnya dominasi kelas mayoritas pada proporsi data, model dapat mempelajari batas antar-kelas secara lebih jelas sehingga kebingungan (*confusion*) antar kelas menurun dan akurasi seluruh kelas meningkat. Dengan demikian, *random oversampling* terbukti mengatasi bias akibat *class imbalance* tanpa menimbulkan penurunan performa pada kelas mayoritas (*trade-off* negatif).



Gambar 5. Performa Per Kelas Tanpa Oversampling

Performa model pada skenario *baseline* (tanpa *oversampling*) ditunjukkan pada Gambar 5. Secara umum, model mampu mendeteksi kelas mayoritas dengan baik, namun terdapat kelemahan pada nilai *Recall* untuk kelas minoritas seperti *Truck* yang hanya mencapai 0,93. Rendahnya nilai *Recall* ini mengindikasikan bahwa model tanpa penyeimbangan data masih sering melewatkan objek truk di lapangan (*false negative* tinggi) akibat kurangnya representasi fitur truk selama proses pelatihan.

Setelah diintervensi dengan teknik *random oversampling* dan augmentasi adaptif, terjadi peningkatan keragaman data yang signifikan pada kelas minoritas. Hasil performa per kelas setelah penyeimbangan data disajikan pada Gambar 6.



Gambar 6. Performa Per Kelas dengan Oversampling

Berdasarkan Gambar 6, terjadi peningkatan metrik yang merata di seluruh kelas kendaraan. Perubahan paling krusial terlihat pada kelas *Truck*, di mana nilai *Recall* melonjak drastis dari 0,93 menjadi 0,98. Peningkatan ini membuktikan bahwa penambahan sampel objek berdimensi besar (bus dan truk) berhasil memandu kepala deteksi multi-skala YOLO26s untuk mengenali batas-batas spasial objek secara lebih sensitif [9]. Selain itu, kelas *Bus* dan *Car* juga mencapai kestabilan metrik yang sangat tinggi dengan nilai rata-rata menyentuh angka 0,99. Hasil ini menegaskan bahwa strategi *oversampling* mampu mengeliminasi bias *class imbalance* tanpa memicu terjadinya *overfitting* atau penurunan performa pada kelas mayoritas.

C. Analisi Confusion Matrix

Confusion matrix digunakan untuk menelaah pola kesalahan klasifikasi model secara lebih rinci. Tabel berikut merangkum jumlah prediksi benar (nilai diagonal) untuk masing-masing kelas pada kedua skenario.

Tabel 6 Analisis tanpa Oversampling dan Dengan Oversampling

Kelas	Prediksi Benar (Tanpa Oversampling)	Prediksi Benar (Dengan Oversampling)
Bus	386	879
Car	7.750	14.776
Motorcycle	788	941
Truck	718	1.438

Pada skenario *baseline*, sumber kesalahan terbesar berasal dari kebingungan antara kelas Car dengan background (458 objek) serta sejumlah objek kelas minoritas yang gagal terdeteksi (terklasifikasi sebagai *background*). Setelah penerapan *oversampling*, dominasi nilai pada diagonal utama (prediksi benar) semakin kuat dan kesalahan klasifikasi antar kelas kendaraan menurun secara relatif, khususnya pada kelas Bus dan Truck yang kebocoran prediksinya (*misclassification*) berkurang. Hal ini mempertegas temuan pada Tabel 6, yaitu bahwa penyeimbangan data memperjelas batas pemisah antar kelas.

D. Analisi Confusion Matrix

Selain aspek akurasi lokalisasi objek, kesiapan model untuk diimplementasikan pada Sistem Transportasi Cerdas (*Intelligent*

Transportation System) sangat bergantung pada kecepatan pemrosesan datanya. Pengujian kecepatan inferensi dilakukan menggunakan kartu grafis NVIDIA Tesla T4 dengan resolusi input gambar sebesar 640 × 640 piksel. Hasil komparasi performa kecepatan antara kedua skenario dirangkum dalam Tabel 7.

Tabel 7. Perbandingan FPS

Skenario	Kecepatan Inference	FPS
Tanpa Oversampling	11.95	83.67
Dengan Oversampling	11.69	85.51

Berdasarkan data pada Tabel 7, model YOLO26s tanpa *oversampling* menghasilkan kecepatan inferensi sebesar 11,95 ms per citra atau setara dengan 83,67 FPS. Sementara itu, model dengan penerapan *oversampling* mencatatkan waktu inferensi sebesar 11,69 ms dengan kecepatan mencapai 85,51 FPS.

Secara teoritis, intervensi rekayasa data lewat *oversampling* hanya memengaruhi distribusi dataset pada fase pelatihan (*training*) dan tidak mengubah arsitektur internal ataupun jumlah parameter komputasi model saat fase pengujian (*deployment/inference*). Oleh karena itu, kedua skenario menghasilkan performa kecepatan yang relatif identik dan stabil di kisaran 83–85 FPS.

E. Pengujian Implementasi Real-time

Untuk mengevaluasi kelayakan model pada sistem manajemen lalu lintas cerdas, kedua model diujicobakan pada rekaman video lalu lintas dengan metode penghitungan kendaraan (*vehicle counting*) berbasis algoritma pelacakan (*ByteTrack*), sebagaimana ditunjukkan pada Gambar 7 dan Gambar 8 [6],[12]. Secara kualitatif, model dengan *oversampling* menghasilkan kotak deteksi (*bounding box*) yang lebih konsisten dan tingkat keyakinan (*confidence score*) yang lebih tinggi pada deteksi multi-skala, terutama untuk kendaraan besar (bus dan truk) yang sebelumnya kerap terlewat pada model *baseline*. Peningkatan keandalan deteksi ini berdampak langsung pada akurasi penghitungan jumlah kendaraan yang melintas, sehingga model lebih layak diterapkan untuk pemantauan lalu lintas secara *real-time*.

Gambar 7. Hasil Perhitungan video Tanpa Oversampling

Gambar 8. Hasil Perhitungan video dengan Oversampling

IV. KESIMPULAN

Penelitian ini berhasil membuktikan bahwa penerapan teknik *random oversampling* pada model Ultralytics YOLO26s efektif mendongkrak seluruh metrik performa deteksi dan mengeliminasi bias akibat ketidakseimbangan data (*class imbalance*). Secara kuantitatif, nilai Precision global meningkat menjadi 0,984 (+1,3%), Recall menjadi 0,974 (+2,4%), mAP@0.5 menjadi 0,993, dan mAP@0.5:0.95 mengalami lonjakan paling signifikan menjadi 0,911 (+6,9 poin). Intervensi ini juga berhasil menaikkan performa di seluruh kelas kendaraan, terutama pada kelas minoritas seperti Truck yang nilai *Recall*-nya melonjak dari 0,93 menjadi 0,98 tanpa memicu *overfitting* atau penurunan performa pada kelas mayoritas. Selain unggul dalam konsistensi *bounding box* dan tingkat keyakinan (*confidence score*) multi-skala saat diintegrasikan dengan algoritma pelacak ByteTrack pada video *real-time*, model ini tetap mempertahankan kecepatan inferensi tinggi di kisaran 83–85 FPS (11,69 ms pada GPU NVIDIA Tesla T4). Meskipun demikian, model masih menyisakan sedikit kesalahan klasifikasi antar-kelas (*misclassification*) yang menjadi peluang untuk

pengembangan dan penyempurnaan sistem pada penelitian selanjutnya demi implementasi Sistem Transportasi Cerdas yang lebih andal.

UCAPAN TERIMA KASIH

Penulis menyampaikan rasa terima kasih yang sebesar-besarnya kepada Universitas Duta Bangsa Surakarta yang telah memberikan dukungan fasilitas akademik dan lingkungan penelitian yang kondusif selama penyusunan jurnal ini. Apresiasi setinggi-tingginya juga kami tujukan kepada Dinas Perhubungan (Dishub) Kabupaten Sukoharjo atas penyediaan data visual, akses informasi, serta infrastruktur jaringan kamera pemantau lalu lintas (CCTV) yang menjadi objek krusial dalam pengujian real-time pada penelitian ini. Tidak lupa, terima kasih kami sampaikan kepada rekan-rekan sejawat dan semua pihak yang telah meluangkan waktu serta memberikan masukan konstruktif demi kesempurnaan draf penelitian ini.

REFERENSI

[1] 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. 134866–134877, 2023, doi: 10.1109/ACCESS.2023.3335826.

[2] D. Katare, D. Solans Noguero, S. Park, N. Kourtellis, M. Janssen, and A. Y. Ding, "Analyzing and Mitigating Bias for Vulnerable Classes: Towards Balanced Representation in Dataset," arXiv preprint arXiv:2401.10397, Jan. 2024.

[3] [3] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," version 8.0, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[4] K. Jordan, "Muon: An optimizer for filters," GitHub Repository, 2024. [Online]. Available: https://github.com/kellerjordan/muon

[5] C. H. Lim, T. Connie, T. S. Ong, et al., "Visual-Based Vehicle Detection with Adaptive Oversampling," International Journal of Information Technology, vol. 16, pp. 4767–4777, 2024, doi: 10.1007/s41870-024-01977-w.

[6] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using YOLO: Challenges, architectural successors, datasets and applications," Multimedia Tools and Applications, vol. 82, no. 6, pp. 9243–9275, 2023, doi: 10.1007/s11042-022-13644-y.

[7] J. Liu, Y. Xie, Y. Zhang, and H. Li, "Vehicle Flow Detection and Tracking Based on an Improved YOLOv8n and ByteTrack Framework," World Electric Vehicle Journal, vol. 16, no. 1, p. 13, 2025, doi: 10.3390/wevj16010013.

[8] J. Zhao, J. Hao, X. Zhou, S. Xiao, and B. Wang, "Improved Vision-Based Vehicle Detection and Classification by Optimized YOLOv4," IEEE Access, vol. 10, pp. 8590–8603, 2022, doi: 10.1109/ACCESS.2022.3143365.

[9] C. Li dkk., "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," arXiv preprint arXiv:2209.02976, 2022.

[10] T. Kumar, R. Brennan, A. Mileo, and M. Bendechache, "Image Data Augmentation Approaches: A Comprehensive Survey and Future Directions," IEEE Access, vol. 12, pp. 187536–187571, 2024, doi: 10.1109/ACCESS.2024.3470122.

[11] L. Wen, D. Du, Z. Cai, Z. Lei, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang, and S. Lyu, "UA-DETRAC: A New Benchmark and Protocol for Multi-Object Detection and Tracking," Computer Vision and Image Understanding, vol. 193, p. 102907, Aug. 2020, doi: 10.1016/j.cviu.2020.102907.

[12] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "ByteTrack: Multi-Object Tracking by Associating Every Detection Box," Proceedings of the European Conference on Computer Vision (ECCV), pp. 1–21, Oct. 2022, doi: 10.1007/978-3-031-20047-2_1.