Published online on the page: https://journal.upp.ac.id/index.php/rjti/index

R J T I
Riau Jurnal Teknik Informatika
| E - ISSN (*Online*) 2829-2529 |

Optimasi *YOLO12n* untuk Deteksi Kendaraan pada Perangkat *Edge* melalui Ablasi *Batch size* dan Resolusi *Input*

Yusuf Anshori[1,*], Muhammad Yazdi[2], Yuri Yudhaswana Joefrie[3]

[1,2,3]Jurusan Teknologi Informasi, Universitas Tadulako, Indonesia

Informasi Artikel

Sejarah Artikel:
Submit: 17 Maret 2026
Revisi: 30 Maret 2026
Diterima: 31 Maret 2026
Diterbitkan: 31 Maret 2026

Kata Kunci

Yolo12n, Deteksi Kendaraan, *Batch Size*, Resolusi *Input*, *Edge Computing*.

Korespondensi

E-mail:
[1]yusuf.anshori@untad.ac.id*

A B S T R A C T

Real-time vehicle detection on edge devices is a critical component of Intelligent Transportation Systems (ITS). Detection performance on edge devices is heavily influenced by training configuration, such as batch size and input resolution, yet their combined effect on YOLO12n for PKJI vehicle classification has not been systematically studied. This study evaluates 16 training configurations derived from combinations of four batch sizes (8, 16, 32, and 64) and four input resolutions (320, 480, 640, and 720 pixels) for vehicle detection on NVIDIA Jetson Orin Nano. The resulting models were then evaluated on a dataset of 9,381 images across three inference formats: PyTorch, TensorRT FP32, and TensorRT FP16. Results reveal a non-linear interaction between batch size and input resolution, where three configurations exhibited training instability and higher resolution did not guarantee better real-time performance on edge devices. The optimal configuration in this experiment was TensorRT FP16 at 480-pixel resolution (FPS = 35.01, mAP@50 = 0.9109), consuming half the GPU memory of its FP32 counterpart. These findings indicate that training configuration and inference optimization strategy play a decisive role in determining the deployment viability of vehicle detection on edge devices.

Abstrak

Deteksi kendaraan *real-time* pada perangkat *edge* merupakan komponen penting dalam Sistem Transportasi Cerdas (ITS). Performa deteksi di perangkat *edge* sangat dipengaruhi oleh konfigurasi pelatihan seperti *batch size* dan resolusi *input*, namun pengaruh keduanya terhadap *YOLO12n* untuk klasifikasi kendaraan berbasis PKJI belum banyak dikaji secara sistematis. Penelitian ini mengevaluasi 16 konfigurasi pelatihan yang berasal dari kombinasi empat *batch size* (8, 16, 32, dan 64) dan empat resolusi *input* (320, 480, 640, dan 720 piksel) untuk deteksi kendaraan pada NVIDIA *Jetson Orin Nano*. Model yang dihasilkan kemudian diuji menggunakan *dataset* 9.381 citra pada tiga format inferensi, yaitu *PyTorch, TensorRT FP32*, dan *TensorRT FP16*. Hasil menunjukkan bahwa interaksi *batch size* dan resolusi *input* bersifat non-linier. Tiga konfigurasi mengalami ketidakstabilan selama pelatihan, sedangkan resolusi tinggi tidak menjamin performa *real-time* yang lebih baik pada perangkat *edge*. Konfigurasi terbaik dalam eksperimen ini adalah *TensorRT FP16* pada resolusi 480 piksel (FPS = 35,01, mAP@50 = 0,9109) dengan konsumsi memori GPU setengah dari konsumsi format FP32. Temuan ini menunjukkan bahwa konfigurasi pelatihan dan strategi optimasi inferensi berperan penting dalam menentukan kelayakan deployment deteksi kendaraan pada perangkat *edge*.

1. Pendahuluan

Deteksi kendaraan merupakan komponen penting dalam manajemen lalu lintas dan sistem transportasi cerdas (*Intelligent Transportation Systems,* ITS). Informasi mengenai keberadaan dan jenis kendaraan digunakan dalam berbagai aplikasi seperti pengendalian sinyal lalu lintas adaptif, pemantauan arus kendaraan, serta pengambilan keputusan berbasis data [1]. Sistem ITS yang efektif menuntut deteksi kendaraan yang akurat dan *real-time* pada kondisi lingkungan yang dinamis seperti variasi pencahayaan, kepadatan kendaraan tinggi, dan occlusion antar objek [2]. Implementasi sistem deteksi kendaraan pada infrastruktur ITS seperti titik pemantauan CCTV di persimpangan jalan menghadirkan tantangan komputasi yang berbeda dari sistem berbasis server GPU [3], sehingga mendorong penggunaan pendekatan *edge* computing sebagai solusi yang sesuai untuk aplikasi *real-time*. Kondisi ini mendorong penggunaan model deteksi objek yang mampu mempertahankan akurasi tinggi dengan latensi inferensi yang rendah pada perangkat dengan sumber daya terbatas.

Performa model pada perangkat *edge* tidak hanya ditentukan oleh arsitektur jaringan, tetapi juga oleh konfigurasi pelatihan yang digunakan. Dua parameter konfigurasi yang berpengaruh adalah *batch size* dan resolusi *input*. Resolusi yang lebih tinggi memperkaya representasi fitur visual objek, namun sekaligus meningkatkan beban komputasi yang berdampak pada latensi inferensi dan konsumsi memori GPU [4], [5]. *Batch size* memengaruhi stabilitas gradien, dinamika konvergensi, dan kemampuan generalisasi model terhadap data baru [6]. Pengaruh kombinasi *batch size* dan resolusi *input* terhadap performa model *YOLO12n* pada tugas deteksi kendaraan masih jarang dianalisis secara sistematis dalam literatur.

Penelitian ini mengusulkan studi ablasi terstruktur menggunakan model *YOLO12n*, yaitu varian nano dengan ~2,6 juta parameter untuk *edge deployment* [7]. Studi ini memvariasikan *batch size* dan resolusi *input* secara sistematis pada *dataset* kendaraan berdasarkan PKJI (Pedoman Kapasitas Jalan Indonesia). *YOLOv12* dipilih karena mekanisme *Area Attention* dan modul *Residual Efficient Layer Aggregation Network (R-ELAN)* [8] dengan performa yang terbukti lebih baik dibanding generasi *YOLO* sebelumnya pada *dataset* lalu lintas [1], [9]. Model diinisialisasi menggunakan bobot *pretrained MS-COCO* melalui pendekatan *transfer learning* [10], kemudian dievaluasi pada perangkat *NVIDIA Jetson Orin Nano* dalam tiga format inferensi: *PyTorch native, TensorRT FP32,* dan *TensorRT FP16.*

Berbagai penelitian terkait telah dilakukan namun dengan fokus dan batasan yang berbeda. Li et al. [11] mengusulkan *YOLO-edge* berbasis *YOLOv5s* yang mencapai 47 FPS pada Jetson Orin NX melalui desain *slim neck* dan *multi-scale feature fusion,* namun menggunakan arsitektur yang dimodifikasi dan *dataset* BDD100K yang tidak merepresentasikan klasifikasi PKJI. Song et al. [10] menerapkan *sparse training* dan *channel pruning* pada *YOLO-Fast* pada Atlas 200I, namun pada domain UAV yang berbeda dari deteksi kendaraan jalan raya. He et al. [12] memvalidasi YOLO-Lite *real-time* pada perangkat *edge* dengan *dataset* CCTV lalu lintas, namun tanpa variasi *batch size* dan resolusi secara sistematis. Hasan et al. [13] mengembangkan BDNet berbasis *YOLOv12* untuk pemantauan lalu lintas perkotaan dan memvalidasi ketahanannya terhadap oklusi padat dan variasi skala kendaraan, namun dengan modifikasi arsitektur backbone dan tanpa studi ablasi konfigurasi pelatihan. Ajayi et al. [6] secara empiris membuktikan pengaruh signifikan variasi *batch size* terhadap akurasi dan stabilitas konvergensi *YOLOv8*, namun pada domain klasifikasi citra UAV pertanian, bukan deteksi kendaraan pada perangkat *edge*. Chaman et al. [1] melakukan benchmarking menyeluruh *YOLOv8* hingga *YOLOv12* pada skenario ITS, namun seluruhnya dijalankan pada server GPU tanpa evaluasi konfigurasi pelatihan pada perangkat *edge*. Ringkasan perbandingan penelitian terkait dan posisi penelitian ini ditunjukkan pada Tabel 1.

Tabel 1. Penelitian terdahulu dan posisi penelitian ini

Referensi	Model	*Dataset*	Hardware	Strategi Utama	Fokus Studi
[1]	YOLOv8–v12 *benchmark*	BDD100K+CCTSDB	Server GPU	*Benchmarking* arsitektur	*Server* GPU; tidak ada evaluasi konfigurasi pelatihan pada *edge*
[6]	YOLOv8 + *hyper tuning*	multi-crop UAV	Colab GPU	Variasi *batch size*	Klasifikasi UAV; bukan deteksi kendaraan pada perangkat *edge*
[11]	YOLO-*edge* (YOLOv5s)	BDD100K	Jetson Orin NX	Slim neck, FSPPM	Arsitektur dimodifikasi; bukan *YOLO12n*; bukan *dataset* PKJI
[14]	YOLO-Fast (YOLOv8)	VisDrone	Atlas 200I	*Pruning*, distilasi	Domain UAV; bukan kendaraan jalan; bukan PKJI
[12]	YOLO-Lite (*improved*)	CCTV traffic	*Edge device*	*Lightweight architecture*	Tidak ada variasi *batch size*/resolusi secara sistematis
[13]	BDNet (*YOLOv12*)	BRVD + VisDrone	GPU	*Modified architecture*	Arsitektur dimodifikasi; tidak ada studi ablasi
Penelitian ini (*Proposed Study*)	*YOLO12n* (*pretrained* COCO)	PKJI 5 kelas	*Jetson Orin Nano*	Ablasi *batch* × *resolusi*	Studi ablasi *batch size* (8/16/32/64) × resolusi (320/480/640/720 px) pada *YOLO12n*, *dataset* PKJI, *Jetson Orin Nano*

Berdasarkan tinjauan tersebut, gap penelitian yang teridentifikasi adalah belum adanya studi yang secara sistematis mengevaluasi interaksi kombinasi *batch size* dan resolusi *input* terhadap performa *YOLO12n* standar berbobot pretrained COCO untuk deteksi kendaraan berbasis klasifikasi PKJI pada perangkat *edge NVIDIA Jetson Orin Nano,* sekaligus membandingkan dampak format inferensi *TensorRT* terhadap *throughput* dan latensi.

Penelitian ini memberikan tiga kontribusi utama: (1) studi ablasi terstruktur terhadap 16 konfigurasi *batch size* dan resolusi *input* pada *YOLO12n* untuk deteksi kendaraan berbasis klasifikasi PKJI; (2) evaluasi trade-off akurasi dan kecepatan pada tiga format inferensi: *PyTorch, TensorRT FP32,* dan *TensorRT FP16* di perangkat *Jetson Orin Nano*; serta (3) rekomendasi konfigurasi optimal untuk *deployment real-time* pada sistem ITS berbasis *edge* di Indonesia.

Penelitian ini bertujuan untuk: (1) menganalisis pengaruh empiris kombinasi *batch size* (8, 16, 32, 64) dan resolusi *input* (320, 480, 640, 720 piksel) terhadap mAP@50, mAP@50:95, *Precision*, dan *Recall* pada *YOLO12n* untuk *dataset* berbasis klasifikasi PKJI; dan (2) mengidentifikasi konfigurasi pelatihan optimal beserta format inferensi terbaik untuk deployment *real-time* pada *Jetson Orin Nano*. Hasil penelitian dapat memberikan kontribusi empiris pada kajian optimasi model deteksi kendaraan berbasis *edge* dan memberikan wawasan baru dalam pemilihan konfigurasi pelatihan dan strategi optimasi inferensi untuk sistem deteksi kendaraan ITS berbasis *edge* di Indonesia.

2. Metode Penelitian

2.1. Tahapan Penelitian

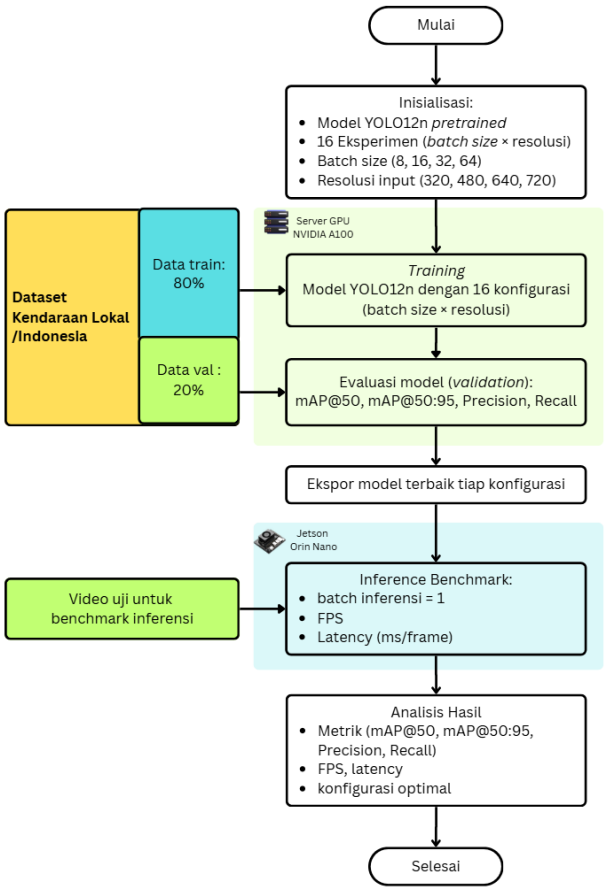
Penelitian ini menggunakan studi ablasi terstruktur untuk mengevaluasi pengaruh dua variabel independen, yaitu *batch size training* dan resolusi *input*, terhadap performa model *YOLO12n* pada skenario deteksi kendaraan berbasis *edge*. Empat nilai *batch size training* (8, 16, 32, 64) dan empat nilai resolusi *input* (320, 480, 640, 720 piksel) dikombinasikan sehingga menghasilkan 16 konfigurasi eksperimen (E1–E16). Setiap konfigurasi dilatih menggunakan bobot inisialisasi yang sama (*pretrained COCO*) untuk memastikan komparabilitas yang adil antar konfigurasi. Konfigurasi eksperimen dengan

batch size training 16 dengan resolusi 640 *piksel* dipilih sebagai baseline karena merupakan konfigurasi default yang digunakan dalam pelatihan model YOLO [1], [11], [15]. Adapun parameter pelatihan selain dua parameter tersebut dijaga konstan pada seluruh 16 konfigurasi eksperimen sebagaimana disajikan pada tabel 2.

Tabel 2. Parameter pelatihan yang digunakan pada seluruh eksperimen *training*

Parameter	Nilai	Keterangan
Model	*YOLO12n*	*Pretrained* COCO
Epoch (maks.)	100	—
Early Stopping Patience	5	Metrik: mAP@50:95
Jumlah kelas	5	SM, MP, KS, BB, TB
Batch size	8 / 16 / 32 / 64	Variabel independen
Resolusi *Input*	320 / 480 / 640 / 720 px	Variabel independen
Optimizer	Auto	-
Learning rate (lr0)	0,01	*Initial learning rate*
LR final ratio (lrf)	0,01	*Linear decay*
Momentum	0,937	-
Weight decay	0,0005	*Regularisasi* L2
Warmup epochs	3,0	-
Augmentasi	*mosaic*=1,0 *fliplr*=0,5 *scale*=0,5 *erasing*=0,4 hsv h=0,015 hsv s=0,7 hsv v=0,4	-
Seed	0	*deterministic=True*
AMP	*True*	*Mixed precision training*

Tahapan penelitian secara keseluruhan ditampilkan pada gambar 1. Penelitian ini dilaksanakan dalam dua tahap utama pada lingkungan komputasi yang terpisah. Tahap pertama adalah pelatihan dan validasi model pada *server GPU NVIDIA A100,* di mana setiap konfigurasi dilatih dan divalidasi menggunakan *dataset* kendaraan lokal Indonesia dan dievaluasi menggunakan metrik mAP@50, mAP@50:95, *Precision,* dan *Recall*. Model terbaik dari setiap konfigurasi kemudian diekspor untuk tahap kedua, yaitu *inference benchmark* pada perangkat *edge NVIDIA Jetson Orin Nano*, di mana setiap model diuji menggunakan video uji dengan *batch* inferensi tunggal (*batch*=1) untuk mengukur *throughput* (FPS) dan latensi dalam kondisi yang merepresentasikan *deployment* nyata.



Gambar 1. Tahapan penelitian

2.2. *Dataset*

Dataset pada penelitian ini diperoleh dari rekaman CCTV lokal di Kota Palu dan tangkapan layar video YouTube berlisensi *Creative Commons*. Pengumpulan data dilakukan pada lima rentang waktu, yaitu pagi (06.00–08.00), siang (11.00–13.00), sore (15.00–16.00), senja (18.00–19.00), dan malam (20.00–22.00) untuk merepresentasikan variasi pencahayaan dan kepadatan lalu lintas. Anotasi dilakukan menggunakan format YOLO dengan bounding box per kelas dan mengacu pada pedoman klasifikasi kendaraan PKJI 2023 yaitu Sepeda Motor (SM, *Motorcycle*), Mobil Penumpang (MP, *passenger_car),* Kendaraan Sedang (KS, *medium_vehicle*), Bus Besar (BB, *large_bus*), dan Truk Besar (TB, *large_truck*). *Dataset* dibagi menjadi dua: pelatihan (*train*) dan validasi (val) dengan rasio 80:20 dengan distribusi *dataset* yang ditampilkan pada tabel 3.

Tabel 3. Distribusi *dataset*

Kelas	Train Anotasi	Val Anotasi	Total Anotasi
(BB) *large_bus*	1.445	363	1.808
(TB) *large_truck*	3.675	895	4.570
(KS) *medium_vehicle*	1.395	291	1.686
(SM) *Motorcycle*	8.484	2.114	10.598
(MP) *passenger_car*	10.237	2.409	12.646
Total	**25.236**	**6.072**	**31.308**
Total Citra	**7.505**	**1.876**	**9.381**

Berdasarkan tabel 3, kelas *motorcycle* dan *passenger_car* memiliki jumlah anotasi yang jauh lebih besar dibandingkan *large_bus* dan *medium_vehicle*. Ketimpangan distribusi ini berpotensi mempengaruhi sensitivitas model terhadap kelas minoritas. Contoh variasi citra *dataset* pada berbagai kondisi ditunjukkan pada gambar 2.

Gambar 2. Contoh citra *dataset* pada berbagai kondisi pencahayaan dan kepadatan lalu lintas

2.3. Lingkungan Eksperimen

Model yang digunakan adalah *YOLOv12* varian nano (*YOLO12n*) karena memiliki ukuran model yang relatif ringan dengan jumlah parameter kecil sehingga mendukung proses inferensi pada perangkat dengan keterbatasan sumber daya komputasi [3], [7]. Model diinisialisasi menggunakan bobot *pretrained* dari *dataset* MS-COCO untuk memanfaatkan representasi fitur umum yang telah dipelajari sebelumnya melalui pendekatan *transfer learning* [10].

Penelitian ini menggunakan dua lingkungan komputasi yang terpisah sesuai fungsinya, yaitu: server GPU untuk proses pelatihan (*training*), dan perangkat *edge* untuk evaluasi kecepatan inferensi (*inference benchmark*). Pelatihan model dilakukan pada sebuah *workstation* yang dilengkapi unit *GPU NVIDIA A100 80GB PCIe* dan evaluasi inferensi (*FPS benchmark*) dilakukan pada *NVIDIA Jetson Orin Nano*.

2.4. Metrik Evaluasi

Evaluasi model deteksi objek menggunakan serangkaian metrik standar yang mengukur kinerja deteksi dan kecepatan inferensi [1], [8], [16]. Metrik yang digunakan adalah *Intersection over Union (IoU), Precision (P), Recall (R),* dan *Frames Per Second (FPS)*. Adapun metrik dinyatakan dalam persamaan berikut:

$$\mathrm{IoU} = \frac{|\mathrm{A} \cap \mathrm{B}|}{|\mathrm{A} \cup \mathrm{B}|} \tag{1}$$

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP+FP}} \tag{2}$$

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \tag{3}$$

$$\mathrm{FPS} = \frac{\mathrm{N\_frames}}{\mathrm{T\_inference}} \tag{4}$$

Pada persamaan (1), A menyatakan luas bounding box prediksi dan B menyatakan luas *bounding box ground-truth*. Seluruh metrik dihitung berdasarkan tiga komponen dasar, yaitu *true positive (TP), false positive (FP),* dan *false negative (FN)*. TP menyatakan deteksi yang benar dengan nilai *IoU* memenuhi ambang yang ditetapkan, FP menyatakan deteksi yang salah, sedangkan FN menyatakan objek yang tidak berhasil terdeteksi. Pada Persamaan (4), N_{frames} menyatakan jumlah frame yang diproses, sedangkan $T_{inference}$ menyatakan total waktu inferensi. Sebuah deteksi dinyatakan sebagai TP apabila IoU ≥ threshold yang ditetapkan.

Pada penelitian ini, dua threshold digunakan, yaitu: *IoU* = 0.50 (mAP@50) dan *IoU* = 0.50–0.95 (mAP@50:95). Seluruh model dievaluasi menggunakan validation set dan kinerja kecepatan inferensi diukur dalam satuan *Frames Per Second (FPS)* secara langsung pada perangkat *Jetson Orin Nano* untuk merepresentasikan kondisi implementasi pada lingkungan nyata.

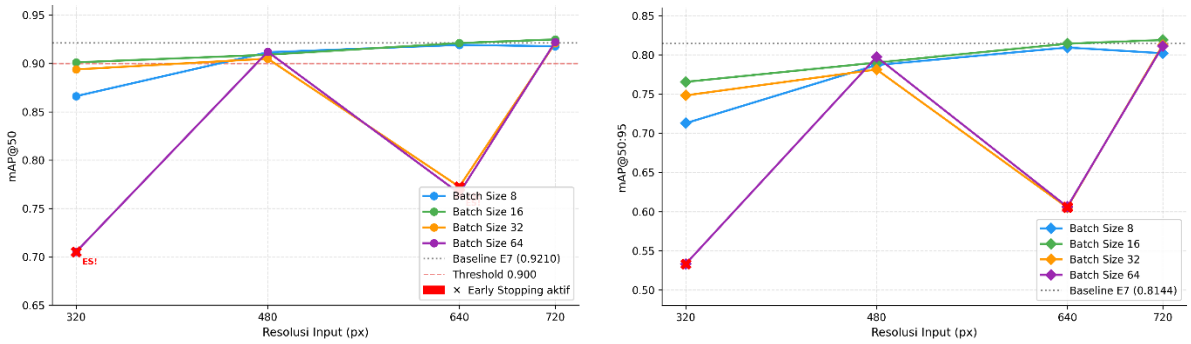
3. Hasil dan Pembahasan

3.1 Ablasi *hyperparameter*: *batch size* dan resolusi *input*

Eksperimen ablasi dirancang untuk mengetahui interaksi antara ukuran batch (8, 16, 32, 64) dan resolusi *input* (320, 480, 640, 720 piksel) terhadap performa model dan stabilitas pelatihan model. Sebanyak 16 konfigurasi dijalankan selama 100 *epoch* dengan *early stopping (patience=5)*. Hasil eksperimen dapat dilihat pada tabel 4 dan gambar 3.

Tabel 4. Hasil eksperimen ablasi *hyperparameter*: *batch size* dan resolusi *input*

Eksp ID	Konfigurasi		Epoch			Akurasi				Early Stopping Aktif
	Batch Size	Resolusi (px)	Max Epoch	Epoch Stop	Best Epoch	mAP@50	mAP @50:95	Precision	Recall	
E1	8	320	100	33	28	0,8661	0,7129	0,8442	0,8112	–
E2	8	480	100	48	43	0,9114	0,7871	0,8796	0,8563	–
E3	8	640	100	58	53	0,9191	0,8095	0,8830	0,8771	–
E4	8	720	100	42	37	0,9177	0,8022	0,8678	0,8745	–
E5	**16**	**320**	**100**	**81**	**76**	**0,9010**	0,7656	0,8705	0,8602	**–**
E6	16	480	100	53	48	0,9089	0,7902	0,8785	0,8538	–
E7	**16**	**640**	**100**	**65**	**60**	**0,9210**	0,8144	0,8840	0,8672	**–**
E8	**16**	**720**	**100**	**64**	**59**	**0,9247**	0,8191	0,8747	0,8854	**–**
E9	32	320	100	53	48	0,8937	0,7484	0,8570	0,8401	–
E10	32	480	100	50	45	0,9049	0,7814	0,8802	0,8355	–
E11	32	640	100	6	1	0,7725	0,6053	0,7516	0,7280	✓
E12	32	720	100	50	45	0,9208	0,8127	0,8906	0,8628	–
E13	64	320	100	6	1	0,7053	0,5333	0,7758	0,5741	✓
E14	**64**	**480**	**100**	**67**	**62**	**0,9120**	0,7975	0,8758	0,8599	**–**
E15	64	640	100	6	1	0,7659	0,6062	0,7265	0,6590	✓
E16	64	720	100	50	45	0,9221	0,8113	0,8856	0,8673	–



(a) mAP@50 (b) mAP@50:95

Gambar 3. Hubungan akurasi dan resolusi

Secara keseluruhan, konfigurasi E8 mencapai akurasi tertinggi (mAP@50=0,9247) dan diikuti oleh E16 (mAP@50=0,9221) dan E7 (mAP@50=0,9210). Peningkatan *batch size* dari 8 ke 16 cenderung memperbaiki akurasi pada resolusi 320, 640, dan 720 piksel, namun tidak berlaku universal. Hal ini terlihat pada resolusi 480 piksel di mana akurasi justru mengalami penurunan marginal pada E2 (mAP@50=0,9114) dan E6 (mAP@50=0,9089).

Namun, peningkatan *batch size* menjadi 32 dan 64 memicu ketidakstabilan pelatihan. Hal ini terlihat pada tiga konfigurasi spesifik: E11 (*batch size*=32, resolusi = 640 px), E13 (*batch size* =64, resolusi = 320 px), dan E15 (*batch size* =64, resolusi = 640 px). Sebaliknya, *batch size* 32 dan 64 pada resolusi 720 px menunjukkan stabilitas *training* normal pada E12 (mAP@50 = 0,9208) dan E16 (mAP@50 = 0,9221).

Eksperimen ini memperlihatkan bahwa peningkatan *batch size* tidak selalu diikuti oleh peningkatan akurasi yang konsisten pada setiap resolusi *input*. Ketidakstabilan kritis terjadi pada *batch size* 32 dan 64, di mana sebagian konfigurasi menunjukkan ketidakstabilan pelatihan *(early stopping)*, sementara lainnya tetap stabil. Temuan ini menunjukkan adanya interaksi sensitif antara *batch size* dan resolusi *input* dalam menentukan keberhasilan pelatihan model.

Untuk menjaga analisis tetap terarah dan sistematis, penelitian ini memilih satu model terbaik pada setiap level resolusi berdasarkan dua kriteria utama, yaitu: (1) akurasi tertinggi (mAP@50) dalam kelompok resolusi yang sama, dan (2) stabilitas selama proses pelatihan. Dengan kriteria tersebut, empat model yang dipilih untuk tahap inferensi *edge* adalah E5 pada resolusi 320 piksel, E14 pada resolusi 480 piksel, E7 pada resolusi 640 piksel, dan E8 pada resolusi 720 piksel.

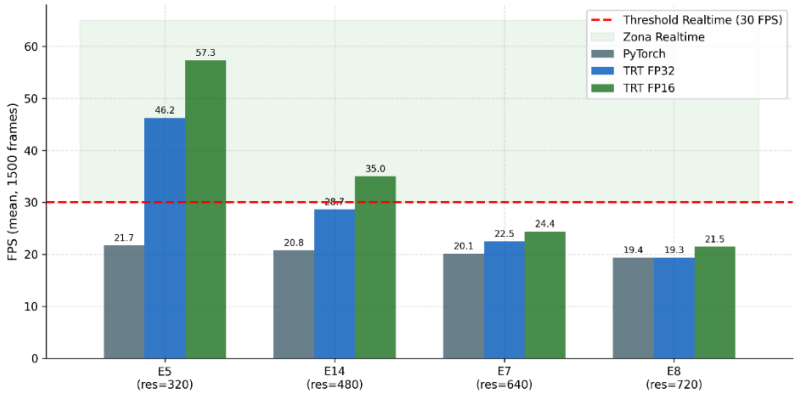
3.2 Analisis *Benchmark* pada Perangkat *Jetson Orin Nano*

Evaluasi performa inferensi dilakukan terhadap empat konfigurasi eksperimen terpilih (E5, E7, E8, dan E14) yang mewakili variasi resolusi *input* 320×320, 640×640, 720×720, dan 480×480 piksel. Setiap konfigurasi diuji dalam tiga format inferensi, yaitu *PyTorch native* (.pt), *TensorRT FP32*, dan *TensorRT FP16*, sehingga diperoleh total 12 skenario pengujian dengan masing-masing skenario dilakukan 3 kali pengujian. Metrik utama yang dievaluasi mencakup *throughput* rata-rata (FPS Mean) dan latensi maksimum (Lat Max), serta konsumsi memori GPU. Ambang batas *real-time* ditetapkan pada ≥ 30 FPS sesuai standar pemrosesan video, dengan hasil rangkuman disajikan pada tabel 5 dan gambar 4.

Tabel 5. Benchmark Inference - *YOLO12n* pada *Jetson Orin Nano*

Eksp ID*	Format	Image size (px)	FPS Mean	Max Latency (ms)	*Real-time* ≥30 FPS
E5	PyTorch (.pt)	320	21,7213	49,8468	✗
E5	**TRT FP32**	**320**	**46,2458**	**25,9575**	✓
E5	**TRT FP16**	**320**	**57,2790**	**21,1023**	✓
E7	PyTorch (.pt)	640	20,1291	52,0900	✗
E7	TRT FP32	640	22,4929	226,8376	✗
E7	TRT FP16	640	24,3952	42,7573	✗
E8	PyTorch (.pt)	720	19,3525	55,7353	✗
E8	TRT FP32	720	19,3460	60,3773	✗
E8	TRT FP16	720	21,4565	51,1572	✗
E14	PyTorch (.pt)	480	20,7919	52,1265	✗
E14	TRT FP32	480	28,6565	36,9312	✗
E14	**TRT FP16**	**480**	**35,0109**	**32,3367**	✓

Exp ID : Experiment ID, TRT : TensorRT, FPS : Frame per second, RT : Real-time



Gambar 4. FPS tiap format model (*PyTorch, TensorRT FP32,* dan *TensorRT FP16)*

Perbandingan antar format inferensi secara konsisten menunjukkan bahwa konversi ke *TensorRT* memberikan peningkatan *throughput* yang signifikan dibanding *native PyTorch* pada seluruh skenario resolusi. Konversi ke *TensorRT* FP32 memberikan peningkatan *throughput* yang bervariasi bergantung resolusi: peningkatan terjadi pada resolusi 320 px (E5 FP32: 46,25 FPS, naik 113,0% dari baseline PyTorch) dan 480 px (E14 FP32: 28,66 FPS, naik 37,8%), namun tidak memberi peningkatan pada resolusi 720 px (E8 FP32: 19,35 FPS).

Hasil evaluasi latensi menunjukkan bahwa konfigurasi FP16 menghasilkan performa inferensi yang lebih konsisten dibandingkan FP32. E5 TRT FP16 (320 px) dan E14 TRT FP16 (480 px) memperlihatkan distribusi latensi, masing-masing dengan latensi maksimum 21,10 ms dan 32,34 ms. Sebaliknya, E7 TRT FP32 (640 px) memperlihatkan lonjakan latensi maksimum hingga 226,84 ms. Temuan ini menunjukkan bahwa rata-rata latensi tidak cukup merepresentasikan reliabilitas inferensi karena berisiko menurunkan deteksi dan menyebabkan kehilangan observasi objek pada frame tertentu. Oleh karena itu, konfigurasi dengan latensi rata-rata rendah tetapi variabilitas tinggi tidak dapat dianggap optimal untuk deployment sistem ITS *real-time*.

3.3 Analisis Inferensi pada *Jetson Orin Nano*

Selain *throughput* dan latensi, evaluasi deployment juga mempertimbangkan perubahan performa deteksi setelah model diekspor ke format *TensorRT* serta efisiensi penggunaan memori GPU. Pada penelitian ini, inferensi diuji pada empat resolusi *input*, yaitu 320 piksel, 480 piksel, 640 piksel, dan 720 piksel, menggunakan tiga format model, yaitu *PyTorch* (.pt), *TensorRT* FP32, dan *TensorRT FP16.* Evaluasi dilakukan dengan mempertimbangkan metrik akurasi validasi, yaitu *precision, recall,* mAP@50, dan mAP@50:95, serta kebutuhan memori GPU TensorRT, dan ukuran *engine*. Hasil inferensi dan alokasi GPU untuk model *YOLO12n* dapat dilihat pada tabel 6 dan 7.

Tabel 6. Performa inferensi tiap format model pada *Jetson Orin Nano*

Identitas Model		Image size	Akurasi (val set)			
Exp ID*	**Format**	**(px)**	**Precision**	**Recall**	**mAP@50**	**mAP@50:95**
E5	PyTorch (.pt)	320	0,8737	0,8602	0,9015	0,7655
E5	TRT FP32	320	0,8687	0,8475	0,8999	0,7621
E5	TRT FP16	320	0,8689	0,8476	0,9001	0,7621
E14	PyTorch (.pt)	480	0,8782	0,8598	0,9122	0,7975
E14	TRT FP32	480	0,8833	0,8529	0,9108	0,7934
E14	TRT FP16	480	0,8835	0,8519	0,9109	0,7933
E7	PyTorch (.pt)	640	0,8845	0,8691	0,9224	0,8151
E7	TRT FP32	640	0,8861	0,8561	0,9157	0,8044
E7	TRT FP16	640	0,8859	0,8564	0,9153	0,8041
E8	PyTorch (.pt)	720	0,8773	0,8856	0,9259	0,8208
E8	TRT FP32	720	0,8819	0,8685	0,9191	0,8095
E8	TRT FP16	720	0,8814	0,8687	0,9189	0,8094

Tabel 7. Alokasi GPU format *TensorRT* pada *Jetson Orin Nano*

Identitas Model		TRT GPU Mem (MiB)		Engine Size (MiB)
Exp ID*	**Format**	**GPU Alloc (MiB)**	**GPU Total (MiB)**	
E5	TRT FP32	10	14	13
E5	TRT FP16	5	7	8
E14	TRT FP32	9	20	13
E14	TRT FP16	5	10	9
E7	TRT FP32	9	32	14
E7	TRT FP16	5	16	9
E8	TRT FP32	9	42	14
E8	TRT FP16	5	21	8

Exp ID : Experiment ID

Hasil pengujian menunjukkan trade-off yang jelas antara akurasi dan kecepatan inferensi: *PyTorch* unggul dalam akurasi, *TensorRT FP16* unggul dalam kecepatan, sementara *TensorRT FP32* secara umum

berada di antara keduanya. Dari sisi akurasi, resolusi *input* yang lebih tinggi secara konsisten meningkatkan performa deteksi. Pada model *PyTorch,* mAP@50 meningkat dari 0,9015 (320 px) hingga 0,9259 (720 px), dengan pola serupa pada mAP@50:95 dari 0,7655 menjadi 0,8208. Resolusi lebih tinggi memungkinkan model mempertahankan detail spasial yang lebih baik untuk lokalisasi objek yang lebih presisi. Namun, peningkatan akurasi mulai melandai di resolusi tinggi, hal ini dapat dilihat pada peningkatan resolusi 640 px ke 720 px yang relatif kecil.

Konversi ke *TensorRT* menimbulkan penurunan akurasi yang sangat kecil dan dapat diabaikan. Pada 320 px, mAP@50 hanya turun dari 0,9015 (*PyTorch*) menjadi 0,8999 (FP32) dan 0,9001 (FP16). Pola serupa terjadi di seluruh resolusi. Yang lebih penting, performa *TensorRT* FP16 hampir identik dengan FP32 dari sisi akurasi, sehingga kuantisasi ke FP16 tidak merugikan kualitas deteksi secara berarti. Sebaliknya, perbedaan kecepatan inferensi sangat signifikan. Pada 320 px, *TensorRT* FP32 mencapai 46,25 FPS dan FP16 mencapai 57,28 FPS, jauh melampaui PyTorch yang hanya 21,72 FPS. Namun, pada resolusi 640 px ke atas, seluruh format gagal memenuhi threshold *real-time* 30 FPS. Temuan ini menunjukkan bahwa resolusi tinggi dapat meningkatkan akurasi dengan proses komputasi yang signifikan.

Dari seluruh konfigurasi, hanya tiga yang memenuhi ambang *real-time* (≥ 30 FPS): TRT FP32 dan FP16 pada 320 px, serta TRT FP16 pada 480 px. Di antara ketiganya, kompromi terbaik antara akurasi dan kecepatan dalam eksperimen ini adalah TRT FP16 480 px dengan FPS=35,01, mAP@50=0,9109 dan mAP@50:95=0,7933. Dari sisi efisiensi memori, TRT FP16 secara konsisten menggunakan sekitar separuh memori GPU dibanding FP32 dengan ukuran engine yang juga lebih kecil. Hal ini menjadikannya pilihan yang tepat untuk diterapkan pada perangkat *edge*.

Secara keseluruhan, terdapat tiga simpulan yang dapat dirumuskan. Pertama, model paling akurat tidak otomatis menjadi model paling layak *deploy*, seperti eksperimen pada resolusi 720 piksel menghasilkan mAP tinggi tetapi gagal memenuhi kebutuhan *real-time*. Kedua, dalam skenario eksperimen ini, *TensorRT* FP16 menunjukkan efisiensi terbaik dari sisi *throughput* dan penggunaan memori GPU. Ketiga, resolusi 480 piksel dengan *TensorRT* FP16 memberikan keseimbangan terbaik antara akurasi, FPS, latensi, dan efisiensi *resource* pada *Jetson Orin Nano*. Penelitian ini menunjukkan bahwa keberhasilan *deployment model* pada perangkat *edge* tidak hanya ditentukan oleh akurasi *training*, tetapi juga oleh strategi optimasi inferensi dan pemilihan resolusi yang tepat. Untuk sistem transportasi cerdas berbasis *edge*, pemilihan konfigurasi model harus mempertimbangkan *throughput*, latensi, dan efisiensi memori. Dalam konteks ini, *TensorRT* FP16 pada resolusi 480 piksel merupakan pendekatan paling rasional.

4. Kesimpulan

Penelitian ini mengevaluasi pengaruh kombinasi *batch size* (8, 16, 32, 64) dan resolusi *input* (320, 480, 640, 720 piksel) terhadap performa *YOLO12n* untuk deteksi kendaraan berbasis PKJI pada *Jetson Orin Nano* melalui 16 konfigurasi eksperimen dan tiga format inferensi. Terdapat empat temuan utama dari penelitian ini. Pertama, interaksi *batch size* dan resolusi bersifat non-linier: tiga konfigurasi (E11, E13, E15) menunjukkan ketidakstabilan pelatihan, sementara *batch size* besar pada resolusi 720 piksel justru stabil. Kedua, resolusi tinggi meningkatkan akurasi namun peningkatannya semakin tidak signifikan seiring kenaikan resolusi. E8 (*batch size* 16, 720 piksel) mencatat mAP@50 tertinggi (0,9247) tetapi gagal memenuhi ambang *real-time*. Ketiga, *TensorRT* FP16 terbukti sebagai format inferensi paling efisien, mempertahankan akurasi hampir identik dengan FP32 (selisih <0,002 mAP@50) dengan konsumsi memori separuhnya. Keempat, dari 12 skenario inferensi, hanya tiga yang memenuhi ≥30 FPS, dan konfigurasi *E14 TRT FP16* (480 piksel, FPS = 35,01, mAP@50 = 0,9109) menjadi konfigurasi terbaik dalam eksperimen ini untuk deployment ITS *real-time*. Penelitian ini menunjukkan bahwa *YOLO12n* dengan bobot *pretrained COCO* mampu memberikan performa yang baik pada *dataset* PKJI, dan pemilihan

konfigurasi pelatihan serta strategi optimasi inferensi terbukti penting dalam konteks *edge* deployment. Penelitian ini berkontribusi sebagai evaluasi empiris pemilihan konfigurasi *YOLO12n* berbasis klasifikasi PKJI untuk sistem ITS *edge* di Indonesia.

Penelitian lanjutan disarankan untuk memperluas ruang eksplorasi dengan menyertakan hyperparameter tambahan seperti *learning rate, optimizer*, dan strategi augmentasi data serta investigasi lebih lanjut terhadap fenomena ketidakstabilan pelatihan pada konfigurasi *batch size* besar dengan resolusi menengah. Peningkatan kualitas dan kuantitas data pelatihan, penambahan skenario kondisi lingkungan yang lebih beragam serta penyeimbangan distribusi antar kelas kendaraan diharapkan dapat meningkatkan kemampuan generalisasi model secara signifikan. Secara praktis, pengembangan pipeline deteksi *end-to-end* berbasis model optimal yang mencakup modul tracking dan antarmuka analitik lalu lintas *real-time* pada perangkat *edge* merupakan arah pengembangan lanjutan yang relevan.

Ucapan Terimakasih

Penelitian ini didukung oleh PPAPT Kementerian Pendidikan Tinggi, Sains, dan Teknologi Republik Indonesia (Kemdiktisaintek).

Daftar Pustaka

[1] M. Chaman, A. El Maliki, H. Dahou, and A. Hadjoudja, "Benchmarking YOLO-based deep learning models for *real-time* object detection in hybrid ADAS and intelligent transportation systems," Results in Engineering, vol. 29, no. 16, p. 108942, Mar. 2026, doi: 10.1016/j.rineng.2025.108942.

[2] M. A. Berwo et al., "Deep Learning Techniques for Vehicle Detection and Classification from Images/Videos: A Survey," Sensors 2023, Vol. 23, vol. 23, no. 10, May 2023, doi: 10.3390/s23104832.

[3] J. L. Mela and C. G. Sánchez, "Yolo-based power-efficient object detection on *edge* devices for USVs," Journal of *Real-time* Image Processing 2025 22:3, vol. 22, no. 3, pp. 108-, May 2025, doi: 10.1007/s11554-025-01682-2.

[4] H. Feng, G. Mu, S. Zhong, P. Zhang, and T. Yuan, "Benchmark Analysis of YOLO Performance on *Edge* Intelligence Devices," Cryptography 2022, Vol. 6, vol. 6, no. 2, Apr. 2022, doi: 10.3390/cryptography6020016.

[5] F. Z. Guerrouj, S. R. Florez, A. El Ouardi, M. Abouzahir, and M. Ramzi, "Quantized Object Detection for *Real-time* Inference on Embedded GPU Architectures," International Journal of Advanced Computer Science and Applications, vol. 16, no. 5, p. 2025, 2025, doi: 10.14569/IJACSA.2025.0160503.

[6] O. G. Ajayi, P. O. Ibrahim, and O. S. Adegboyega, "Effect of Hyperparameter Tuning on the Performance of YOLOv8 for Multi Crop Classification on UAV Images," Applied Sciences 2024, Vol. 14, vol. 14, no. 13, Jun. 2024, doi: 10.3390/app14135708.

[7] P. Mittal, "A comprehensive survey of deep learning-based lightweight object detection models for *edge* devices," Artificial Intelligence Review 2024 57:9, vol. 57, no. 9, pp. 242-, Aug. 2024, doi: 10.1007/s10462-024-10877-1.

[8] Y. Tian, Q. Ye, and D. Doermann, "*YOLOv12*: Attention-Centric *Real-time* Object Detectors," Feb. 2025, doi: 10.48550/arXiv.2502.12524.

[9] Q. Chen, "Traffic Object Detection Using *YOLOv12*," Open Access Library Journal, vol. 12, no. 8, pp. 1–15, Aug. 2025, doi: 10.4236/oalib.1113991.

[10] T. Ge, B. Ning, and Y. Xie, "YOLO-AFR: An Improved *YOLOv12*-Based Model for Accurate and *Real-time* Dangerous Driving Behavior Detection," Applied Sciences 2025, Vol. 15, vol. 15, no. 11, May 2025, doi: 10.3390/app15116090.

[11] X. Li, L. Chen, T. Huang, A. Yang, and W. Liu, "YOLO-*edge*: *real-time* vehicle detection for *edge* devices," Cluster Computing 2024 28:5, vol. 28, no. 5, pp. 289-, Apr. 2025, doi: 10.1007/s10586-024-04963-w.

[12] M. He, C. Li, J. Yang, and X. Ning, "*Real-time* vehicle detection methods based on an improved YOLO-Lite approach in *edge* computing scenarios," Discover Artificial Intelligence 2026, Feb. 2026, doi: 10.1007/s44163-026-00885-1.

[13] M. M. Hasan et al., "BDNet: A Lightweight *YOLOv12*-Based Vehicle Detection Framework for Smart Urban Traffic Monitoring," Smart Cities 2026, Vol. 9, vol. 9, no. 2, Feb. 2026, doi: 10.3390/smartcities9020033.

[14] Z. Song, X. Zhang, and P. Tan, "YOLO-Fast: a lightweight object detection model for *edge* devices," The Journal of Supercomputing 2025 81:5, vol. 81, no. 5, pp. 724-, Apr. 2025, doi: 10.1007/s11227-025-07172-3.

[15] Y. Li et al., "YOLO-*EDGE*: an object detection algorithm for traffic scenarios," The Journal of Supercomputing 2025 81:7, vol. 81, no. 7, pp. 840-, May 2025, doi: 10.1007/s11227-025-07275-x.

[16] A. A. Murat and M. S. Kiran, "A comprehensive review on YOLO versions for object detection," Engineering Science and Technology, an International Journal, vol. 70, p. 102161, Oct. 2025, doi: 10.1016/j.jestch.2025.102161.