Journal of Science and Social Research ISSN 2615 – 4307 (Print)
Oct 2026, IX (5): 8963 – 8970 ISSN 2615 – 3262 (Online)
Available online at http://jurnal.goretanpena.com/index.php/JSSR

DETEKSI OBJEK WAKTU NYATA PADA JETSON ORIN NANO UNTUK PEMANTAUAN KAPASITAS ORANG

Fitria Priyulida[1], Fitria Nova Hulu[2], David Nababan[3], Natha Niel Mendrofa[4]
Universitas Sari Mutiara Indonesia, Medan
Politeknik Negeri Medan, Sumatera Utara
e-mail: fpriyulida27@gmail.com

Abstract: *Real-time object detection on edge devices forces a trade-off between image quality and inference speed, yet both are often reported as a single performance figure. This study examines that trade-off on an NVIDIA Jetson Orin Nano paired with a Logitech C922 USB camera for person-capacity monitoring. The first stage tested two resolutions, three lighting conditions, and three distances to select a camera configuration. The second stage built a detectNet program with TensorRT acceleration that counts persons and bottles, raises a capacity-exceeded status when three or more persons appear, and logs every frame to CSV across six test scenarios. A resolution of 1280 x 720, normal lighting, and a 1 m distance gave the best visual quality (4 of 5), while low light and backlight reduced it to 2. At 640 x 480 the program reached 137-176 FPS with 5.7-7.3 ms latency, roughly five times the camera limit of 30 fps, which shows that inference rate and capture rate must be reported separately. Raising the confidence threshold from 0.50 to 0.70 removed low-confidence detections of 51-63%. Timing at 1280 x 720 failed (FPS reported as Inf) and temperature read 0.0 C, so these findings remain preliminary until wall-clock timing and sensor readings are corrected.*

Keywords: *edge AI, Jetson Orin Nano, real-time object detection, confidence threshold, person counting.*

Abstrak: *Deteksi objek waktu nyata di perangkat edge selalu menukar kualitas citra dengan kecepatan inferensi, tetapi keduanya sering dilaporkan sebagai satu angka performa. Penelitian ini menguji pertukaran tersebut pada NVIDIA Jetson Orin Nano dengan kamera USB Logitech C922 untuk pemantauan kapasitas orang. Tahap pertama menguji dua resolusi, tiga kondisi cahaya, dan tiga jarak untuk memilih konfigurasi kamera. Tahap kedua membangun program berbasis detectNet dan TensorRT yang menghitung orang dan botol, memberi status KAPASITAS TERLAMPAUI bila terdapat tiga orang atau lebih, dan mencatat setiap bingkai ke CSV pada enam skenario uji. Resolusi 1280 x 720, cahaya normal, dan jarak 1 m menghasilkan kualitas visual terbaik (4 dari 5), sedangkan cahaya rendah dan backlight menurunkannya ke nilai 2. Pada 640 x 480 program mencapai 137-176 FPS dengan latency 5,7-7,3 ms, sekitar lima kali batas kamera 30 fps, sehingga laju inferensi dan laju tangkap kamera harus dilaporkan terpisah. Kenaikan threshold dari 0,50 ke 0,70 menyaring deteksi berkeyakinan rendah 51-63%. Pengukuran pada 1280 x 720 gagal (FPS terbaca Inf) dan suhu terbaca 0,0 C, sehingga temuan ini masih bersifat awal sampai pengukuran waktu dinding dan pembacaan sensor diperbaiki.*

Kata kunci: *edge AI, Jetson Orin Nano, deteksi objek waktu nyata, threshold confidence, penghitungan orang.*

PENDAHULUAN

Ruang tunggu, laboratorium, dan halte punya satu masalah yang sama: jumlah orang di dalamnya perlu diketahui saat itu juga, bukan setelah rekaman diperiksa. Mengirim setiap bingkai video ke server memang membuka akses ke

model yang besar, tetapi membawa jeda jaringan dan membuat citra wajah meninggalkan lokasi. *Edge AI* memindahkan inferensi ke perangkat di dekat kamera sehingga latency turun dan data tidak perlu dikirim keluar, dengan harga berupa memori, daya komputasi, dan anggaran energi yang terbatas (Wang et al., 2025; "Edge AI in Practice," 2025; Al-Mimi et al., 2026).

Keluarga Jetson dari NVIDIA sudah cukup sering diuji untuk kebutuhan ini. Shalash et al. (2026) melaporkan bahwa inferensi YOLO hanya dengan CPU pada Jetson Orin Nano tertahan di bawah 3 FPS, sedangkan GPU menaikkan kecepatan lebih dari lima kali dan menekan beban CPU di bawah 10 persen. Pada Jetson Orin NX, TensorRT lebih cepat 17,7 persen daripada PyTorch pada batch 2, tetapi gagal pada batch besar karena memori tidak cukup (Aljami et al., 2026). Pilihan arsitektur juga menggeser angka secara tajam: YOLOX mencapai 107 FPS dan YOLOv12 45 FPS pada perangkat yang sama (Kim et al., 2025), dan cara implementasi, native atau hasil generasi kode, ikut mengubah kualitas deteksi serta latency (Otero et al., 2026). Orin Nano bahkan dinilai layak untuk deteksi waktu nyata dengan daya di bawah 2 W (Mela & Sánchez, 2025). Pertanyaan "apakah perangkat ini mampu" pada dasarnya sudah terjawab.

Pertanyaan yang lebih jarang dijawab berada di sisi masukan. Resolusi yang lebih besar menambah jumlah piksel kira-kira secara kuadratik, sehingga latency dan memori naik sementara detail objek kecil membaik (Burkus et al., 2026). Cahaya rendah menurunkan kontras, melemahkan tekstur, dan menambah derau, dan ketahanan detektor pada kondisi ini masih terbatas (Su & Lu, 2026; Yang et al., 2026; Yao et al., 2026). Jarak menentukan seberapa besar objek terlihat; estimasi jarak dengan YOLOv8 dan OpenCV mencapai akurasi rata-rata 91 persen dengan galat sekitar 3,6 persen (Syahrudin et al., 2025). Untuk penghitungan orang, deteksi YOLO dengan pelacakan ringan sudah dipakai di stasiun trem berbasis Jetson Nano (Diaz-Santos et al., 2025) dan pada robot bergerak (Gomulka & Wozniak, 2026). Di atas semua itu, threshold *confidence* menentukan deteksi mana yang dianggap ada, dan threshold rendah cenderung menambah *false positive* (Wenkel et al., 2021).

Celahnya terletak pada cara angka-angka tersebut dihasilkan. Sebagian besar *benchmark* mengukur model pada dataset atau berkas video standar, sehingga kamera terpisah dari inferensi. Di lapangan, operator harus memutuskan resolusi, posisi, pencahayaan, dan threshold sebelum melihat satu pun angka FPS. Ada pula kekeliruan yang mudah terjadi: FPS inferensi dibaca sebagai kemampuan kamera, padahal kamera USB biasa dibatasi 30 fps. Kegagalan pengukuran, seperti pewaktu CUDA yang gagal atau sensor suhu yang tidak terbaca, juga jarang dilaporkan, padahal kegagalan semacam itu dapat membuat angka performa tampak lebih baik dari kenyataan.

Penelitian ini menelusuri rantai lengkap dari kamera sampai keputusan status pada Jetson Orin Nano dengan pustaka jetson-inference (Franklin, n.d.). Kontribusinya meliputi: (1) prosedur pemilihan konfigurasi kamera melalui uji resolusi, cahaya, jarak, dan sudut; (2) program pemantauan kapasitas dengan aturan tiga orang, pencatatan CSV, dan penyimpanan gambar peringatan; (3) analisis performa pada enam skenario yang memisahkan laju tangkap kamera dari laju inferensi; dan (4) dokumentasi kegagalan pengukuran beserta perbaikan yang diperlukan sebelum data dapat dipakai sebagai bukti. Karena sebagian pengukuran belum valid, hasilnya diposisikan sebagai studi awal.

METODE

Desain Penelitian

Penelitian menggunakan desain eksperimental terapan dalam dua tahap. Tahap pertama (17 September 2026)

Journal of Science and Social Research ISSN 2615 – 4307 (Print)
Oct 2026, IX (5): 1 – 6 ISSN 2615 – 3262 (Online)
Available online at http://jurnal.goretanpena.com/index.php/JSSR

mengarakterisasi kamera untuk memilih satu konfigurasi terbaik. Tahap kedua (23-24 September 2026) mengembangkan program deteksi secara bertahap, lalu mengujinya pada enam skenario. Kelas objek yang dihitung adalah *person* dan *bottle*; kelas lain yang muncul, seperti kursi, dicatat sebagai objek tambahan.

Perangkat dan Konfigurasi

Seluruh inferensi berjalan di perangkat, tanpa server eksternal. Spesifikasi perangkat keras dan perangkat lunak dirangkum pada Tabel 1.

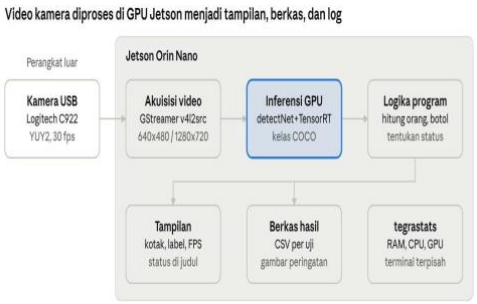
Tabel 1 Perangkat dan Perangkat Lunak

Komponen	Spesifikasi
Perangkat edge	NVIDIA Jetson Orin Nano, RAM 7.337 MB, swap 9.812 MB
Kamera	Logitech C922 USB/UVC, maks. 1920 x 1080, 30 fps
Sistem operasi	Ubuntu (Jetson Linux)
Lingkungan	Conda env AIoT, VS Code
Pustaka	jetson-inference dan jetson-utils (detectNet, videoSource, videoOutput), CUDA, TensorRT
Model	Model deteksi pralatih COCO bawaan jetson-inference
Akuisisi video	GStreamer v4l2src, YUY2, 30/1 fps
Pemantau sistem	tegrastats, interval 1000 ms

Arsitektur Sistem

Kamera mengirim video YUY2 ke Jetson melalui GStreamer v4l2src. Setiap bingkai diproses oleh detectNet yang dipercepat TensorRT, lalu logika program menghitung orang dan botol serta menentukan status.

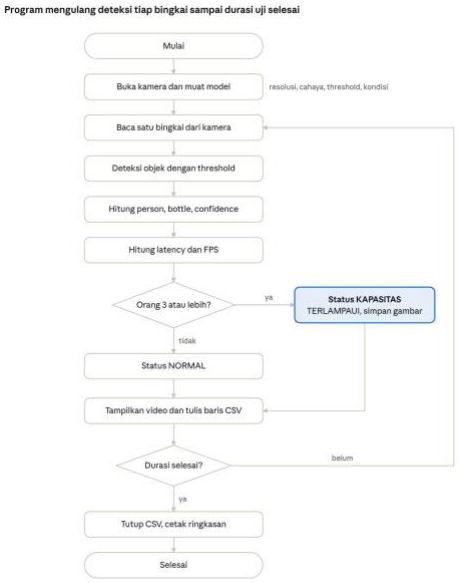
Hasilnya diteruskan ke tiga keluaran: tampilan berisi kotak deteksi, label, FPS, dan status pada judul jendela; berkas CSV per uji beserta gambar peringatan; dan log tegrastats yang berjalan di terminal terpisah untuk memantau RAM, CPU, dan GPU (Gambar 1).



Gambar 1 Diagram Sistem Alur

Program dan Aturan Status

Program membuka kamera dan memuat model dengan parameter resolusi, kondisi cahaya, threshold, dan label kondisi uji. Pada setiap iterasi, satu bingkai dibaca, dideteksi dengan threshold yang ditetapkan, lalu jumlah *person*, *bottle*, dan nilai *confidence* dicatat. Bila terdapat tiga orang atau lebih, status menjadi KAPASITAS TERLAMPAUI dan bingkai disimpan sebagai gambar bukti; bila tidak, status NORMAL. Video ditampilkan, satu baris ditulis ke CSV, dan siklus berulang sampai durasi uji selesai (Gambar 2).



Gambar 2 Flowchart Program

Latency diukur sebagai waktu pemrosesan satu bingkai dalam milidetik, dan FPS inferensi diturunkan darinya:

$$FPS = \frac{1000}{t_{lat}} \quad (1)$$

dengan t_{lat} adalah latency per bingkai (ms). FPS ini menggambarkan kecepatan model, bukan laju kamera menghasilkan gambar, sehingga nilainya dapat melampaui 30 fps.

Skenario Pengujian dan Analisis Data

Pada tahap pertama, kamera diuji pada resolusi 640 x 480 dan 1280 x 720; cahaya normal, rendah, dan backlight; serta jarak 0,5 m, 1 m, dan 2 m pada posisi tengah, kiri, dan kanan. Kualitas citra dinilai secara visual pada skala 1-5. Tahap kedua menjalankan enam skenario pada Tabel 2. Setiap uji berlangsung minimal 30 detik dengan tegrastats aktif. Nilai performa dibaca dari log terminal dan judul jendela, sehingga dilaporkan sebagai rentang, bukan rata-rata dari CSV.

Tabel 2 Rancangan Enam Skenario Uji

Uji	Resolusi	Cahaya	Threshold	Kondisi
1	640 x 480	Normal	0,50	1 orang
2	640 x 480	Normal	0,50	3 orang
3	1280 x 720	Normal	0,50	3 orang
4	640 x 480	Redah	0,50	Objek tambahan
5	640 x 480	Normal	0,70	Objek tambahan
6	640 x 480	Normal	0,50	Uji peringatan penuh

HASIL DAN PEMBAHASAN

Karakterisasi Kamera

Kamera terdeteksi aktif dengan keluaran maksimum 1920 x 1080 pada 30 fps tanpa kendala awal. Tabel 3 merangkum hasil penilaian kualitas citra.

Tabel 3 Hasil Uji Kamera (skala 1-5)

Pengujian	Konfigurasi	Kualitas	FPS kamera
Resolusi	640 x 480	3	30
Resolusi	1280 x 720	4	30
Cahaya	Normal	4	30
Cahaya	Rendah	2	30
Cahaya	Backlight	2	30
Jarak	0,5 m	4	30
Jarak	1 m	4	30
Jarak	2 m	3	30

Cahaya ternyata faktor paling berpengaruh. Resolusi dan jarak masing-masing hanya menggeser kualitas satu tingkat, sedangkan cahaya rendah dan backlight menjatuhkannya dua tingkat; pada backlight, objek utama menjadi siluet karena latar jauh lebih terang. Kenaikan resolusi ke 1280 x 720 tidak mengubah FPS karena kamera mengunci keluaran pada 30 fps di kedua resolusi. Jarak 0,5 m dan 1 m sama-sama bernilai 4, tetapi 1 m dipilih karena cakupannya lebih luas, hal yang menentukan ketika tujuan sistem adalah menghitung beberapa orang sekaligus. Konfigurasi terpilih adalah 1280 x 720, posisi tengah dengan sudut 30 derajat, jarak 1 m, cahaya normal, tanpa objek tambahan.

Pengembangan Program Bertahap

Program dikembangkan dalam tujuh langkah, masing-masing menambah satu kemampuan. Langkah 4 menghitung orang dan menampilkan status serta FPS (107 FPS pada 640 x 480 dengan tiga orang). Langkah 5 menambah hitungan botol dan *confidence* tertinggi (0,97; 140 FPS). Langkah 6 mengganti status menjadi KAPASITAS TERLAMPAUI untuk tiga orang atau lebih (137-157 FPS). Langkah 7 menambah pencatatan CSV dan penyimpanan gambar peringatan (sekitar 175 FPS). Langkah 8 menambah hitungan objek lain dan latency (5,7-7,5 ms; 132-178 FPS). Langkah 9 menjadikan durasi, resolusi, cahaya, threshold, dan kondisi sebagai parameter sehingga keenam uji menghasilkan berkas CSV

terpisah, dan langkah 10 menganalisis performa, threshold, serta suhu secara otomatis dari CSV.

Performa pada Enam Skenario

Tabel 4 menunjukkan hasil keenam skenario. FPS berada pada rentang 137-176 di semua uji yang terukur. Hanya Uji 3 (1280 x 720) yang gagal terukur: FPS tercatat Inf dan latency 0,0 ms.

Tabel 4 Hasil Enam Skenario Pengujian

Uji	Resolusi	Cahaya	Thr.	Kondisi	Person / Bottle	Confidence	FPS	Latency (ms)	Status
1	640 x 480	Normal	0,50	1 orang	1-2 / 1	0,91-0,97	140-176	5,7-7,2	NORMAL
2	640 x 480	Normal	0,50	3 orang	2-3 / 1	0,94-0,99	174-175	5,70-5,74	KT
3	1280 x 720	Normal	0,50	3 orang	3 / 1	0,90-0,94	Inf*	0,0*	KT
4	640 x 480	Rendah	0,50	Objek tambahan	1 / 1	0,92-0,97	139-140	7,16-7,20	NORMAL
5	640 x 480	Normal	0,70	Objek tambahan	2 / 1	0,96	137	7,3	NORMAL
6	640 x 480	Normal	0,50	Uji peringatan penuh	2 / 1	0,93-0,95	140-141	7,1-7,14	NORMAL

KT = KAPASITAS TERLAMPAUI; * tidak terukur akibat galat pewaktu CUDA.

Ada tiga catatan yang membatasi pembacaan tabel ini. Pertama, Uji 4-6 dirancang memakai resolusi terpilih, tetapi judul jendela pada tangkapan layar menunjukkan 640 x 480, sehingga ketiganya kemungkinan berjalan dengan konfigurasi berbeda dari rencana. Kedua, citra pada Uji 4 tampak cukup terang, sehingga kondisi cahaya rendah perlu dipastikan ulang. Ketiga, Uji 6 menampilkan status NORMAL dengan dua orang pada cuplikan yang terlihat; status penuh hanya tampak pada Uji 2 dan 3.

Beban Sistem

Log tegrastats menunjukkan RAM terpakai 4.551-4.948 MB dari 7.337 MB (62-67 persen) dan swap 1.164 MB dari 9.812 MB. Inti CPU bekerja pada 1.510 MHz dengan beban per inti hingga sekitar 78 persen. Kolom GPU dan suhu tidak terbaca utuh, dan keluaran langkah 10 mencetak suhu awal dan akhir 0,0 C untuk keenam uji.

Pembahasan

Temuan yang paling mudah disalahpahami adalah selisih antara 30 fps dan 137-176 FPS. Angka pertama adalah batas kamera, angka kedua adalah seberapa cepat model menyelesaikan satu bingkai. Sistem sebagai satu kesatuan tetap menerima paling banyak 30 bingkai baru per detik, sehingga GPU menganggur sebagian besar waktunya. Ruang sisa sekitar lima kali ini justru informasi yang berguna: ia menunjukkan bahwa model yang lebih berat atau resolusi yang lebih tinggi masih dapat ditampung tanpa menurunkan laju efektif di bawah laju kamera. Pola ini sejalan dengan Shalash et al. (2026), yang menemukan bahwa akselerasi GPU pada Orin Nano memberi lompatan kecepatan beberapa kali lipat dibanding CPU.

Soal resolusi, tahap pertama dan tahap kedua belum bisa disambungkan. Tahap pertama menyatakan 1280 x 720 lebih baik secara visual, sementara dampaknya pada laju inferensi tidak terukur karena Uji 3 gagal dan Uji 4-6 tampaknya berjalan pada 640 x 480. Resolusi 1280 x 720 memuat tiga kali

piksel 640 x 480, dan beban komputasi tumbuh kira-kira kuadratik terhadap dimensi input (Burkus et al., 2026). Pada resolusi itu pula kursi hanya terdeteksi dengan *confidence* 56-63 persen, sehingga detail tambahan belum otomatis berarti objek kecil lebih mudah dikenali. Keputusan resolusi akhir karenanya ditunda sampai pengukuran diulang.

Pada Uji 4, *confidence* tetap 0,92-0,97, setara dengan cahaya normal, padahal tahap pertama menilai cahaya rendah sebagai kondisi terburuk. Ada dua kemungkinan: kondisinya tidak cukup gelap, atau *confidence* objek yang terdeteksi memang tidak peka terhadap pencahayaan. Kemungkinan kedua lebih mengkhawatirkan, karena *confidence* hanya menceritakan objek yang berhasil ditemukan dan diam tentang objek yang terlewat (Wenkel et al., 2021). Literatur cahaya rendah melaporkan bahwa hilangnya kontras dan tekstur menurunkan akurasi deteksi (Su & Lu, 2026; Yao et al., 2026), dan penurunan akurasi semacam itu bisa muncul sebagai objek yang tidak terdeteksi tanpa meninggalkan jejak pada nilai *confidence*. Evaluasi kondisi cahaya seharusnya menghitung jumlah objek yang terlewat, bukan hanya membaca *confidence*.

Threshold 0,70 pada Uji 5 menyisakan deteksi di atas 88 persen, sedangkan pada 0,50 muncul kursi 56-63 persen dan botol 51-55 persen. Bagi pemantauan kapasitas, kedua jenis galat punya harga berbeda. Deteksi orang palsu memicu alarm yang tidak perlu, sedangkan orang yang terlewat membuat ruangan tampak lebih longgar dari kenyataan. Masalahnya, saat tiga orang dan botol berdiri berdekatan, *confidence* turun hingga 51-56 persen. Threshold 0,70 akan membuang deteksi tersebut tepat pada situasi ketika alarm kapasitas paling dibutuhkan. Threshold tinggi aman untuk ruangan yang lengang, tetapi berisiko pada kerumunan, sehingga nilai antara 0,50 dan 0,70 perlu ditentukan berdasarkan kepadatan yang diharapkan, bukan semata kebersihan deteksi.

Latency naik dari sekitar 5,7 ms pada uji awal menjadi sekitar 7,1 ms pada uji berikutnya. Kenaikan 1,4 ms ini kemungkinan berasal dari beban lain yang berjalan bersamaan, seperti peramban terbuka serta penulisan CSV dan gambar. RAM 62-67 persen dan penutupan paksa VS Code (*killed, code 9*) pada 23 September menguatkan dugaan bahwa ruang memori sudah sempit. Untuk penerapan, program sebaiknya berjalan tanpa antarmuka pengembangan. Hubungan beban dan suhu belum dapat disimpulkan karena suhu terbaca 0,0 C, padahal perilaku termal dan daya Jetson berubah pada beban berkelanjutan (Krišlauks et al., 2026).

Kegagalan pengukuran pada Uji 3 berasal dari galat CUDA berulang, yaitu *cudaEventElapsedTime, device not ready* (error 600) di tensorNet.h:709. Pewaktu berbasis CUDA event membaca waktu sebelum GPU selesai, sehingga latency tercatat 0,0 ms dan FPS menjadi tak hingga. Program tetap berjalan, tetapi angka performanya tidak dapat dipercaya. Perbaikannya sederhana: mengukur waktu dinding dengan time.perf_counter() di sekitar fungsi Detect(), lalu menurunkan FPS melalui persamaan (1).

SIMPULAN

Penelitian ini menguji rantai kamera hingga deteksi objek waktu nyata pada Jetson Orin Nano untuk pemantauan kapasitas orang. Hasil awal menunjukkan:

1. Konfigurasi kamera terbaik adalah 1280 x 720, cahaya normal, jarak 1 m, dan sudut 30 derajat dengan kualitas 4 dari 5; cahaya rendah dan backlight menurunkan kualitas ke 2, dan jarak 2 m ke 3.
2. Program deteksi orang dan botol berjalan waktu nyata dengan latency 5,7-7,3 ms (137-176 FPS) pada 640 x 480, sekitar lima kali laju kamera 30 fps, sehingga keduanya harus dilaporkan terpisah.
3. Threshold 0,70 menghasilkan deteksi di atas 88 persen, tetapi berisiko membuang orang yang saling

berdekatan dengan confidence 51-56 persen; pilihan threshold perlu disesuaikan dengan kepadatan ruangan.

4. Data pada 1280 x 720 dan data suhu belum valid karena FPS terbaca Inf dan suhu 0,0 C.

Penelitian selanjutnya perlu mengukur latency dengan waktu dinding, mengulang Uji 3-6 pada 1280 x 720 dengan kondisi cahaya terkendali, memperbaiki pembacaan suhu dari tegrastats, menghitung rata-rata dan simpangan langsung dari CSV, serta mencoba model yang lebih ringan atau presisi FP16/INT8 untuk menurunkan beban memori.

DAFTAR PUSTAKA

Aljami, H. M., Alrowais, N. A., AlAwajy, A. M., Alhrgan, S. M., Aldwaani, R. A., Alsawadi, M. S., Saqib, N. U., Alam, S. S., & Alsubaie, R. (2026). Benchmarking YOLOv8 variants for object detection efficiency on Jetson Orin NX for edge computing applications. *Computers*, *15*(2), 74. https://doi.org/10.3390/computers15020074

Al-Mimi, H., Al-Dahoud, A., Al-Dahoud, A., & Fezari, M. (2026). Edge AI: Embedded intelligence, a review on hardware and applications. *WSEAS Transactions on Information Science and Applications*, *23*.

Burkus, E., Lestyán, B., Dénes-Fazakas, L., & Eigner, G. (2026). Quantized vs. full-precision YOLO models on edge devices: A performance benchmark for real-time license plate detection in smart parking systems. *Sensors*, *26*(16), 5034. https://doi.org/10.3390/s26165034

Diaz-Santos, S., Caballero-Gil, P., & Caballero-Gil, C. (2025). Real-time passenger flow analysis in tram stations using YOLO-based computer vision and edge AI on Jetson Nano. *Computers*, *14*(11), 476. https://doi.org/10.3390/computers14110476

Edge AI in practice: A survey and deployment framework for neural networks on embedded systems. (2025). *Electronics*, *14*(24), 4877. https://doi.org/10.3390/electronics14244877

Franklin, D. (n.d.). *Hello AI World: jetson-inference* [GitHub repository]. https://github.com/dusty-nv/jetson-inference

Gomulka, K., & Wozniak, P. (2026). People counting using YOLO-based detection and clustering for a mobile robot. *Sensors*, *26*(15), 4664. https://doi.org/10.3390/s26154664

Kim, J., Kim, G., Yoshitoshi, R., & Tokuda, K. (2025). Real-time object detection for edge computing-based agricultural automation: A case study comparing the YOLOX and YOLOv12 architectures and their performance in potato harvesting systems. *Sensors*, *25*(15), 4586. https://doi.org/10.3390/s25154586

Krišlauks, R., Tiscenko, D., Medvedevs, V., Ormanis, J., & Judvaitis, J. (2026). Ambient temperature impact on the thermal behavior and power consumption of the NVIDIA Jetson AGX Orin in an outdoor enclosure. *Electronics*, *15*(11), 2467. https://doi.org/10.3390/electronics15112467

Mela, J. L., & Sánchez, C. G. (2025). Yolo-based power-efficient object detection on edge devices for USVs. *Journal of Real-Time Image Processing*, *22*, 108. https://doi.org/10.1007/s11554-025-01682-2

Otero, P. M., Tellaeche, A., & Melero, M. H. (2026). Performance analysis of the YOLO object detection algorithm in embedded systems: Generated code vs. native implementation. *Computation*, *14*(3), 67. https://doi.org/10.3390/computation14030067

Shalash, O., Khatab, E., El-Agamy, A.,

Elmokadem, L., Abouelsaad, Y., Zaki, J., El-Sayed, M., & Said, H. (2026). Benchmarking YOLOv8-YOLOv12 for real-time object detection on single-board computers. *Machine Learning and Knowledge Extraction*, *8*(7), 204. https://doi.org/10.3390/make8070204

Su, Y., & Lu, M. (2026). Exposure-aware training for low-light object detection without target-domain data. *Journal of Imaging*, *12*(6), 245. https://doi.org/10.3390/jimaging12060245

Syahrudin, E., Utami, E., & Hartanto, A. D. (2025). Object detection with YOLOv8 and enhanced distance estimation using OpenCV for visually impaired accessibility. *JOIV: International Journal on Informatics Visualization*, *9*(2), 575-584. https://doi.org/10.62527/joiv.9.2.282

Wang, X., Tang, Z., Guo, J., Meng, T., Wang, C., Wang, T., & Jia, W. (2025). Empowering edge intelligence: A comprehensive survey on on-device AI models. *ACM Computing Surveys*, *57*(9), 228. https://doi.org/10.1145/3724420

Wenkel, S., Alhazmi, K., Liiv, T., Alrshoud, S., & Simon, M. (2021). Confidence score: The forgotten dimension of object detection performance evaluation. *Sensors*, *21*(13), 4350. https://doi.org/10.3390/s21134350

Yang, P., Jiang, J., & Zhang, Y. (2026). A discriminative enhancement and selective fusion method for low-light cross-spectral object detection. *Sensors*, *26*(9), 2684. https://doi.org/10.3390/s26092684

Yao, Z., Xu, H., Zhang, H., Tu, X., & Yin, J. (2026). Improved CenterNet-based multimodal object detection for low-light and complex environments. *Sensors*, *26*(12), 3735. https://doi.org/10.3390/s26123735