Progresif: Jurnal Ilmiah Komputer
https://ojs.stmik-banjarbaru.ac.id/index.php/progresif/index
Jl. Ahmad Yani, K.M. 33,5 - Kampus STMIK Banjarbaru Loktabat – Banjarbaru,
e-mail: puslit.stmikbjb@gmail.com
e-ISSN: 2685-0877

Komparasi YOLOv8 Nano dan YOLO11 Nano untuk Klasifikasi Penyakit Daun Tomat pada Perangkat Edge

DOI: http://dx.doi.org/10.35889/progresif.v22i3.3929

Muhammad Nur[1*], Iwan Jaya[2], Haytsam Adzka Mawla[3]
Teknik Informatika, Universitas Bani Saleh, Bekasi, Indonesia
**e-mail Corresponding Author:* mnur@ubs.ac.id

Abstract

Tomato leaf diseases pose a significant threat to horticultural productivity, while field identification methods remain highly dependent on expert observation. This study compared the performance of YOLOv8 Nano and YOLO11 Nano for tomato leaf disease classification on an edge device. A dataset of 7,200 tomato leaf images across six disease classes was used; both models were trained under identical configurations with 20 independent seed replications, exported to ONNX format, and evaluated on a Raspberry Pi 5. YOLOv8 Nano achieved a mean accuracy of 88.32% while YOLO11 Nano achieved 87.81%. The Wilcoxon Signed-Rank test confirmed that the accuracy difference was not statistically significant (W = 75.5; p = 0.2706). Nevertheless, YOLOv8 Nano demonstrated shorter mean inference time (8.430 ms vs 8.502 ms; p = 0.0064), smaller model size (5.531 MB vs 5.895 MB), and lower peak memory usage (72.31 MB vs 72.72 MB). Both models proved comparable in classification accuracy, but YOLOv8 Nano is recommended for edge deployment owing to its statistically verified efficiency advantages.

Keywords: *YOLOv8 Nano; YOLO11 Nano; tomato leaf disease classification; edge device; Wilcoxon Signed-Rank*

Abstrak

Penyakit daun tomat menjadi ancaman serius bagi produktivitas hortikultura, sementara metode identifikasi visual di lapangan masih sangat bergantung pada keahlian pengamat. Penelitian ini membandingkan performa YOLOv8 Nano dan YOLO11 Nano untuk klasifikasi penyakit daun tomat pada perangkat edge. Dataset yang digunakan terdiri dari 7.200 citra daun tomat yang terbagi dalam enam kelas penyakit; kedua model dilatih menggunakan 20 variasi seed independen dengan konfigurasi identik, kemudian diekspor ke format ONNX dan diuji langsung pada Raspberry Pi 5. YOLOv8 Nano memperoleh akurasi rata-rata 88,32%, sedangkan YOLO11 Nano memperoleh 87,81%. Uji Wilcoxon Signed-Rank mengonfirmasi bahwa selisih akurasi tidak signifikan secara statistik (W = 75,5; p = 0,2706). Meski demikian, YOLOv8 Nano menunjukkan waktu inferensi yang lebih singkat (8,430 ms vs 8,502 ms; p = 0,0064), ukuran model lebih kecil (5,531 MB vs 5,895 MB), dan konsumsi memori lebih rendah (72,31 MB vs 72,72 MB). Kedua model terbukti setara dari sisi akurasi klasifikasi, namun YOLOv8 Nano lebih direkomendasikan untuk implementasi edge berkat keunggulan efisiensinya yang terukur.

Kata kunci: *YOLOv8 Nano; YOLO11 Nano; klasifikasi penyakit daun tomat; perangkat edge; Wilcoxon Signed-Rank*

1. Pendahuluan

Tanaman tomat (*Solanum lycopersicum*) termasuk dalam komoditas hortikultura yang paling banyak dibudidayakan, baik di tingkat global maupun di Indonesia, dengan nilai ekonomi yang cukup tinggi dan permintaan pasar yang konsisten sepanjang tahun. Sayangnya, potensi produktivitas tanaman ini kerap terhambat oleh berbagai penyakit daun yang disebabkan oleh agen patogen berbeda-beda, mulai dari jamur, bakteri, hingga virus. Beberapa penyakit yang paling umum dilaporkan adalah *early blight* akibat jamur *Alternaria solani*, *late blight* yang

disebabkan oleh *Phytophthora infestans*, *mold leaf* dari *Fulvia fulva*, *septoria leaf spot* akibat *Septoria lycopersici*, serta *yellow leaf curl virus* yang ditularkan kutu kebul. Jika tidak dikenali dan ditangani sejak dini, penyakit-penyakit ini dapat mengakibatkan penurunan hasil panen antara 20 hingga 80 persen sekaligus menimbulkan kerugian ekonomi yang substansial bagi para petani [1], [2].

Pada praktiknya, proses identifikasi penyakit daun tomat masih sangat mengandalkan pengamatan visual yang dilakukan oleh petani atau tenaga ahli pertanian. Pendekatan ini memiliki keterbatasan mendasar karena hasilnya sangat bergantung pada pengalaman dan konsistensi individu pengamat. Kerumitan bertambah ketika beberapa jenis penyakit menampilkan gejala visual yang mirip satu sama lain, seperti bercak, perubahan warna daun, dan nekrosis jaringan. Kesalahan dalam mengidentifikasi penyakit di lapangan berpotensi mengakibatkan tindakan penanganan yang tidak tepat sasaran, sehingga serangan penyakit justru terus berkembang dan kerugian semakin membesar. Kondisi inilah yang mendorong diperlukannya sebuah pendekatan otomatis berbasis citra yang mampu mengenali jenis penyakit daun tomat secara lebih cepat, konsisten, dan dapat diskalakan [1], [2].

Perkembangan *deep learning* dan *computer vision* membuka peluang besar dalam klasifikasi penyakit tanaman berbasis citra. Jackulin dan Murugavalli [3] serta Nyawose et al. [4] menunjukkan bahwa *machine learning* dan *deep learning* terbukti efektif untuk deteksi penyakit tanaman, meskipun pembahasannya masih bersifat umum pada domain penyakit tanaman. Untuk penyakit daun tomat secara spesifik, Jelali [1] meninjau berbagai pendekatan *deep learning* berbasis dataset lapangan, sedangkan Das et al. [2] menyurvei metode klasifikasi, deteksi, dan segmentasi dan menekankan bahwa tantangan utama tidak hanya pada pencapaian akurasi tinggi, tetapi juga kemampuan model diterapkan dalam kondisi nyata yang memiliki keterbatasan perangkat, variasi lingkungan, serta kebutuhan respons yang cepat. Dari sisi pengembangan model, Sun et al. [5]] mengusulkan model Eff-Swin yang menggabungkan EfficientNetV2 dengan Swin Transformer dengan akurasi 99,70% pada dataset tomat terkontrol, sementara He dan Tong [6] mengembangkan LT-YOLO berbasis YOLO yang ringan dengan fokus pada keseimbangan akurasi dan efisiensi komputasi. Penelitian yang lebih ketat oleh Ramos dan Sappa [7] membandingkan YOLOv8 hingga YOLOv12 pada dataset lapangan dan menemukan YOLO11 unggul dari sisi akurasi, sedangkan Majid dan Ariatmanto [8] membandingkan YOLO11 dan YOLOv8 untuk identifikasi penyakit daun tomat dan melaporkan akurasi YOLO11 sebesar 99,4% serta YOLOv8 sebesar 98,5%. Namun, kedua penelitian tersebut menggunakan *pipeline object detection*, bukan *image classification*, dan tidak mengevaluasi efisiensi inferensi secara langsung pada perangkat *edge* nyata. Perbandingan khusus antara YOLOv8 Nano dan YOLO11 Nano untuk tugas *image classification* dengan protokol multi-*seed* yang ketat dan evaluasi efisiensi inferensi langsung pada perangkat *edge* masih sangat terbatas.

Untuk mengatasi kesenjangan tersebut, penelitian ini membandingkan secara sistematis YOLOv8 Nano dan YOLO11 Nano untuk tugas *image classification* penyakit daun tomat pada perangkat *edge*. Komputasi *edge* dipilih karena memungkinkan proses inferensi dilakukan secara lokal tanpa ketergantungan pada konektivitas awan yang sering kali tidak tersedia di kawasan pertanian terpencil; Singh dan Gill [9] mengidentifikasi pengurangan latensi, perlindungan privasi data, dan kemampuan operasi tanpa jaringan sebagai pendorong utama adopsi *edge AI* dalam pertanian presisi, sedangkan Xu et al. [10] memaparkan strategi implementasi *deep learning* pada perangkat *edge* mencakup desain arsitektur ringan dan kompresi model. Majeed et al. [8] lebih jauh membuktikan kelayakan pendekatan ini melalui prototipe *edge AI* mandiri untuk deteksi penyakit tomat pada platform tertanam kompak. Raspberry Pi 5 dipilih sebagai platform *edge* yang representatif dan terjangkau. Untuk mengakomodasi variabilitas stokastik dalam pelatihan *deep learning*, eksperimen dilakukan dengan 20 *seed* acak independen, dan signifikansi statistik perbedaan dianalisis menggunakan uji Wilcoxon Signed-Rank yang sesuai untuk data berpasangan tanpa asumsi normalitas [11], [12]. Kebaruan (*novelty*) penelitian ini terletak pada evaluasi komparatif yang spesifik terhadap tugas *image classification* (bukan *object detection*), dengan protokol multi-*seed* yang ketat dan *profiling* efisiensi inferensi secara langsung pada perangkat *edge* nyata, sehingga mampu memberikan panduan yang dapat ditindaklanjuti oleh para praktisi dalam memilih model yang paling tepat untuk sistem pemantauan penyakit tanaman berbasis perangkat *edge*.

2. Metode Penelitian

2.1 Desain Penelitian

Penelitian ini menggunakan desain eksperimen komparatif kuantitatif. Variabel bebas adalah jenis arsitektur model, yaitu YOLOv8 Nano dan YOLO11 Nano. Variabel terikat mencakup akurasi, *macro-averaged precision*, *recall*, *F1-score*, waktu inferensi rata-rata, waktu total *pipeline*, FPS, ukuran file model, dan penggunaan memori puncak. Untuk mengakomodasi variasi stokastik dalam pelatihan, setiap model dilatih dengan 20 *seed* acak independen (*seed* 1 hingga 20), menghasilkan total 40 model terlatih. Signifikansi statistik perbedaan berpasangan dievaluasi menggunakan uji Wilcoxon Signed-Rank pada *alpha* = 0,05, yang diimplementasikan melalui pustaka SciPy dalam Python.

2.2 Perangkat Keras dan Perangkat Edge

Penelitian ini menggunakan dua perangkat keras utama yang memiliki peran berbeda dalam tahapan eksperimen. Pelatihan model dilakukan menggunakan Desktop PC dengan prosesor AMD Ryzen 5, RAM 16 GB, dan GPU NVIDIA RTX 4060 untuk akselerasi CUDA, sehingga pelatihan 40 model (20 *seed* × 2 arsitektur) dapat diselesaikan secara efisien.

Pengujian inferensi dilakukan pada Raspberry Pi 5 sebagai perangkat *edge* representatif. Raspberry Pi 5 menggunakan chip BCM2712 dengan prosesor ARM Cortex-A76 *quad-core* pada frekuensi 2,4 GHz, RAM 8 GB LPDDR4X, dan menjalankan sistem operasi Raspberry Pi OS 64-bit. Spesifikasi lengkap perangkat *edge* yang digunakan disajikan pada Tabel 1.

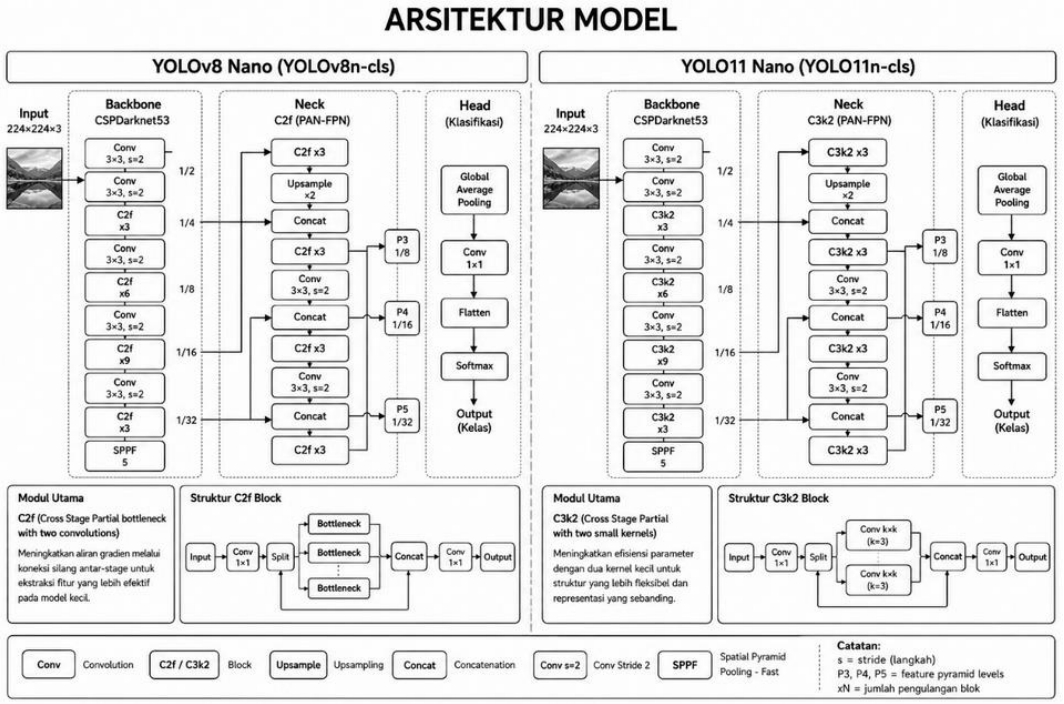
Tabel 1. Spesifikasi Perangkat Edge (Raspberry Pi 5)

Komponen	Spesifikasi
SoC	Broadcom BCM2712
CPU	ARM Cortex-A76 quad-core @ 2,4 GHz (64-bit)
RAM	8 GB LPDDR4X
Sistem Operasi	Raspberry Pi OS 64-bit (Debian Bookworm)
Runtime Inferensi	ONNX Runtime (CPUExecutionProvider)
Thread (intra/inter)	4 / 1

2.3 Arsitektur Model

YOLOv8 Nano (YOLOv8n-cls) dan YOLO11 Nano (YOLO11n-cls) merupakan varian terkecil dari keluarga arsitektur YOLO yang dirancang untuk perangkat dengan sumber daya komputasi terbatas. YOLOv8 Nano menggunakan *backbone* CSPDarknet53 yang dikombinasikan dengan modul *neck* C2f (*Cross Stage Partial bottleneck with two convolutions*). Modul C2f meningkatkan aliran gradien melalui koneksi silang antar-*stage*, sehingga memungkinkan ekstraksi fitur yang lebih efektif pada skala model kecil. YOLO11 Nano merupakan generasi penerus yang diperkenalkan Ultralytics pada tahun 2024 dengan memperkenalkan blok C3k2 (*Cross Stage Partial with two small kernels*) menggantikan modul C2f. Blok C3k2 meningkatkan efisiensi parameter sambil mempertahankan kapasitas representasi yang sebanding, menghasilkan ukuran model yang sedikit lebih besar namun dengan struktur yang lebih fleksibel [13], [14], [15].

Gambar 1 menunjukkan perbandingan arsitektur YOLOv8 Nano (YOLOv8n-cls) dan YOLO11 Nano (YOLO11n-cls) secara ringkas, mulai dari input, backbone, neck, hingga head klasifikasi. Kedua model menggunakan backbone CSPDarknet53 untuk melakukan ekstraksi fitur secara bertahap pada beberapa skala, sedangkan perbedaan utama terletak pada modul utama pada bagian neck, yaitu C2f pada YOLOv8 Nano dan C3k2 pada YOLO11 Nano. Fitur dari beberapa tingkat resolusi kemudian diproses melalui mekanisme upsampling dan concatenation untuk membentuk representasi fitur multiskala pada P3, P4, dan P5. Pada tahap akhir, fitur diproses melalui Global Average Pooling, convolution 1×1, flatten, dan Softmax untuk menghasilkan keluaran berupa kelas prediksi. Bagian bawah gambar juga memperlihatkan struktur internal modul C2f dan C3k2, sehingga perbedaan mekanisme pemrosesan kedua modul dapat terlihat dengan lebih jelas.



Gambar 1. Arsitektur Model Yolov8 Nano dan Yolov11 Nano

Perbandingan karakteristik arsitektur kedua model disajikan pada Tabel 2.

Tabel 2. Perbandingan Karakteristik Arsitektur YOLOv8 Nano dan YOLO11 Nano

Karakteristik	YOLOv8 Nano (n-cls)	YOLO11 Nano (n-cls)
Backbone	CSPDarknet53	CSP-based (diperbarui)
Modul Utama	C2f (Cross Stage Partial)	C3k2 (kernel ganda kecil)
Pre-training	ImageNet	ImageNet
Format Ekspor	ONNX	ONNX
Ukuran Model (ONNX)	5,531 MB	5,895 MB
Parameter (relatif)	Lebih sedikit	Lebih banyak

2.4 Dataset

Dataset yang digunakan adalah *Tomato Leaf Disease Classification Dataset in Pakistan* (Malik et al., 2026) [16] yang tersedia secara publik. Dataset ini terdiri dari 7.200 citra daun tomat yang dikumpulkan dari kondisi lapangan pertanian di Pakistan dengan beragam kondisi pencahayaan dan sudut pengambilan gambar, sehingga karakteristiknya lebih merepresentasikan kondisi nyata dibandingkan dataset berbasis latar terkontrol seperti PlantVillage. Citra dibagi ke dalam enam kelas seimbang, masing-masing berjumlah 1.200 citra per kelas: *Tomato Early Blight*, *Tomato Healthy*, *Tomato Leaf Late Blight*, *Tomato Leaf Yellow Curl Virus*, *Tomato Mold Leaf*, dan *Tomato Septoria Leaf Spot*. Pemilihan dataset lapangan ini konsisten dengan rekomendasi literatur yang menekankan pentingnya data dari kondisi nyata untuk pengujian sistem berbasis *edge* [17], [18].

Dataset dipartisi menjadi data latih (80%), data validasi (10%), dan data uji (10%) menggunakan pembagian bertingkat untuk mempertahankan keseimbangan kelas di seluruh partisi. Distribusi lengkap disajikan pada Tabel 3.

Table 3. Distribusi Dataset Penelitian berdasarkan Kelas dan Partisi

Kelas Penyakit	Latih (80%)	Validasi (10%)	Uji (10%)	Total
Tomato Early Blight	960	120	120	1.200
Tomato Healthy	960	120	120	1.200
Tomato Leaf Late Blight	960	120	120	1.200
Tomato Leaf Yellow Curl Virus	960	120	120	1.200
Tomato Mold Leaf	960	120	120	1.200
Tomato Septoria Leaf Spot	960	120	120	1.200
Total	5.760	720	720	7.200

Preprocessing yang diterapkan pada semua partisi meliputi *resize* ke ukuran 224 × 224 piksel dan normalisasi nilai piksel ke rentang [0, 1]. Augmentasi data diterapkan secara eksklusif pada data latih dan mencakup variasi HSV acak (h = 0,015; s = 0,7; v = 0,4), *horizontal flip* (p = 0,5), *random erasing* (p = 0,4), komposisi *mosaic*, dan RandAugment. Augmentasi ini diimplementasikan melalui *pipeline* pelatihan Ultralytics dan konsisten dengan praktik standar untuk tugas klasifikasi YOLO.

2.5 Konfigurasi Model dan Pelatihan

YOLOv8 Nano (YOLOv8n-cls) dan YOLO11 Nano (YOLO11n-cls) merupakan anggota terkecil dari keluarga arsitektur masing-masing. YOLOv8 menggunakan *backbone* CSPDarknet53 dengan *neck* C2f; YOLO11 memperkenalkan blok C3k2 yang meningkatkan efisiensi parameter sambil mempertahankan kapasitas representasi yang sebanding. Keduanya diinisialisasi dari bobot *pretrained* ImageNet yang tersedia secara publik (*transfer learning*). Konfigurasi hiperparameter yang identik diterapkan pada kedua model sebagaimana dirinci pada Tabel 4. Setiap *seed* menghasilkan satu *checkpoint* bobot terbaik yang dipilih berdasarkan akurasi validasi tertinggi, kemudian diekspor ke format ONNX menggunakan perintah *yolo export* dengan *imgsz=224* dan *simplify=True*.

Table 4. Konfigurasi Pelatihan dan Inferensi

Parameter	Nilai / Konfigurasi
Model dasar	YOLOv8n-cls.pt / YOLO11n-cls.pt (pretrained ImageNet)
Jumlah epoch	50
Ukuran citra	224 × 224 piksel
Batch size	16
Optimizer	Auto (SGD, lr0 = 0,01; weight decay = 0,0005)
Augmentasi data	RandAugment, Mosaic, Flip Horizontal, HSV Jitter
Format ekspor	ONNX (simplify=True, opset otomatis)
Runtime inferensi	ONNX Runtime – CPUExecutionProvider
Perangkat edge	Raspberry Pi 5, RAM 8 GB (ARM Cortex-A76 @ 2,4 GHz)
Thread (intra/inter)	4 / 1
Warm-up	10 citra per sesi pengujian
Variasi seed	20 seed independen (seed 1 hingga 20)

2.6 Protokol Pengujian pada Perangkat Edge

Inferensi dilakukan pada Raspberry Pi 5 (BCM2712, ARM Cortex-A76 × 4 @ 2,4 GHz, RAM 8 GB LPDDR4X) yang menjalankan sistem operasi Raspberry Pi OS 64-bit. ONNX Runtime digunakan dengan *CPUExecutionProvider* yang dikonfigurasi dengan empat *intra-operation thread* dan satu *inter-operation thread*. Skrip *benchmarking* melakukan *warm-up* sebanyak 10 citra sebelum merekam latensi per citra untuk seluruh 720 citra data uji. Untuk setiap citra, *pipeline* pengujian diuraikan menjadi tiga tahap: *preprocessing* (*decode* + *resize* + normalisasi), inferensi (*forward pass* melalui ONNX Runtime), dan *postprocessing* (*argmax* + pemetaan label). Waktu dinding (*wall-clock time*) diukur menggunakan *time.perf_counter()* dengan resolusi milidetik. Penggunaan memori RSS puncak disampling setiap 10 ms dalam *thread* latar menggunakan *psutil* selama sesi *benchmarking* berlangsung, dan nilai maksimum yang teramati dicatat sebagai metrik memori puncak.

2.7 Metrik Evaluasi

Performa klasifikasi dikuantifikasi menggunakan empat metrik utama yang dihitung dari 720 citra data uji untuk setiap *seed*. Akurasi merepresentasikan proporsi sampel yang diklasifikasikan dengan benar dari seluruh sampel yang diuji. *Macro-averaged precision*, *recall*, dan *F1-score* dihitung di seluruh enam kelas untuk memberikan bobot yang sama pada setiap kategori penyakit tanpa memandang jumlah sampel [11]. *Macro F1-score* berfungsi sebagai metrik komparatif utama karena secara bersamaan memperhitungkan kesalahan prediksi positif dan negatif di seluruh kelas. Persamaan yang digunakan adalah sebagai berikut.

Akurasi = Jumlah Prediksi Benar / Jumlah Seluruh Data Uji ……..…. (1)

Precisioni = TPi / (TPi + FPi) ……..…. (2)

Recalli = TPi / (TPi + FNi) ……..…. (3)

F1-scorei = 2 × (Precisioni × Recalli) / (Precisioni + Recalli) ……..…. (4)

Macro F1-score = (1/K) × Σ F1-scorei, i = 1..K ……..…. (5)

Uji Wilcoxon Signed-Rank diterapkan pada distribusi 20-pasang *seed* untuk akurasi, waktu inferensi rata-rata, waktu total *pipeline*, dan FPS; ukuran model dan RSS puncak dilaporkan secara deskriptif karena nilainya tidak berubah antarjalan *seed* [12].

2.8 Analisis Confusion Matrix

Confusion matrix kumulatif dibangun dengan mengagregatkan prediksi dari seluruh 20 *seed*, menghasilkan 20 × 720 = 14.400 evaluasi citra per model. Agregasi ini memberikan pandangan berbasis populasi sampel yang lebih besar terhadap pola kesalahan klasifikasi sistematis, sehingga memungkinkan identifikasi pasangan penyakit yang mudah tertukar secara visual dan berkontribusi secara tidak proporsional terhadap tingkat kesalahan keseluruhan model.

3. Hasil dan Pembahasan

3.1 Hasil Pelatihan Model

Pelatihan kedua model dilakukan selama 50 epoch dengan 20 variasi *seed* independen. Nilai *train loss* pada kedua model secara konsisten menurun sepanjang proses pelatihan, yang menunjukkan kemampuan model dalam mempelajari pola dari data latih. Nilai *validation loss* cenderung lebih fluktuatif dibandingkan *train loss*, yang mengindikasikan adanya variabilitas dalam generalisasi model terhadap data di luar data latih. Fluktuasi semacam ini wajar dalam pelatihan model *deep learning* dan menjadi salah satu alasan mengapa penggunaan 20 variasi *seed* diperlukan untuk mendapatkan gambaran performa yang lebih stabil dan representatif.

3.2 Contoh Hasil Klasifikasi

Gambar 2 menyajikan contoh hasil klasifikasi kedua model terhadap sampel citra daun tomat dari data uji, menampilkan prediksi kelas beserta skor kepercayaan (*confidence score*). Hasil ini menggambarkan kemampuan model dalam membedakan enam kelas penyakit berdasarkan karakteristik visual yang terekstraksi selama proses pelatihan. Kelas dengan ciri visual paling distingtif seperti *Healthy* dan *Yellow Curl Virus* cenderung menghasilkan skor kepercayaan yang lebih tinggi, sedangkan kelas dengan kemiripan morfologis seperti *Early Blight* dan *Mold Leaf* menunjukkan skor yang lebih rendah dan pola kebingungan yang konsisten dengan analisis *confusion matrix* pada Subbagian 3.5.

Gambar 2. *Contoh hasil klasifikasi YOLOv8 Nano dan YOLO11 Nano pada sampel data uji untuk enam kelas penyakit daun tomat. Nilai di setiap panel menunjukkan skor kepercayaan (confidence score) prediksi model.*

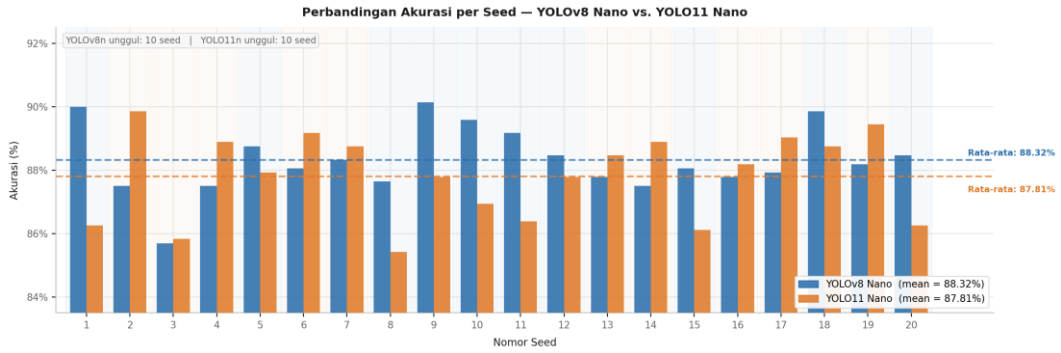
3.3 Perbandingan Akurasi Klasifikasi

Tabel 5 merangkum rata-rata metrik klasifikasi dari 20 replikasi *seed*. YOLOv8 Nano memperoleh rata-rata akurasi 88,32% (Macro F1 = 88,33%), sedangkan YOLO11 Nano memperoleh 87,81% (Macro F1 = 87,89%). Selisih akurasi absolut sebesar 0,51 poin persentase (pp) tergolong kecil dan analisis lebih lanjut mengonfirmasi bahwa perbedaan ini tidak signifikan secara statistik.

Tabel 5. *Rata-rata Hasil Pengujian Klasifikasi dari 20 Seed (Data Uji: 720 Citra)*

Model	Rata-rata Benar	Akurasi (%)	Macro Precision (%)	Macro Recall (%)
YOLOv8 Nano	635,90 / 720	88,32	88,40	88,32
YOLO11 Nano	632,20 / 720	87,81	88,06	87,81

Gambar 3 memvisualisasikan distribusi akurasi per *seed* untuk kedua model. Dari 20 perbandingan berpasangan, YOLOv8 Nano memperoleh akurasi lebih tinggi pada 10 *seed*, sementara YOLO11 Nano juga unggul pada 10 *seed* yang berbeda. Selisih per *seed* berkisar dari −2,36 pp hingga +3,75 pp. Distribusi yang seimbang ini menunjukkan adanya tumpang tindih substansial antara distribusi akurasi kedua model, tanpa ada kecenderungan arah yang konsisten.



Gambar 3. *Perbandingan akurasi per seed: YOLOv8 Nano (biru) dan YOLO11 Nano (oranye) dari 20 seed pelatihan independen. Garis putus-putus menunjukkan nilai rata-rata masing-masing model.*

Detail perbandingan akurasi pada setiap *seed* ditampilkan pada Tabel 6.

Table 6. Perbandingan Akurasi YOLOv8 Nano dan YOLO11 Nano pada Setiap Seed

Seed	YOLOv8n (%)	YOLO11n (%)	Seed	YOLOv8n (%)	YOLO11n (%)
1	90,00	86,25	11	89,17	86,39
2	87,50	89,86	12	88,47	87,78
3	85,69	85,83	13	87,78	88,47
4	87,50	88,89	14	87,50	88,89
5	88,75	87,92	15	88,06	86,11
6	88,06	89,17	16	87,78	88,19
7	88,33	88,75	17	87,92	89,03
8	87,64	85,42	18	89,86	88,75
9	90,14	87,78	19	88,19	89,44
10	89,58	86,94	20	88,47	86,25
Rata-rata	88,32		87,81		

Temuan ini bersesuaian dengan kesimpulan Ramos dan Sappa [7] dalam perbandingan *object detection* mereka pada dataset Tomato-Village, di mana YOLOv11 memang mengungguli YOLOv8 dalam akurasi namun selisihnya sensitif terhadap karakteristik dataset dan jenis tugas. Divergensi antara temuan mereka dan penelitian ini --- YOLO11 unggul dalam deteksi versus kesetaraan dalam klasifikasi --- memperkuat pentingnya evaluasi yang spesifik terhadap tugas. Majid dan Ariatmanto [8] melaporkan angka akurasi absolut yang lebih tinggi untuk kedua model (YOLO11: 99,4%; YOLOv8: 98,5%), kemungkinan besar karena dataset yang mereka gunakan dikumpulkan dalam kondisi yang lebih terkontrol sehingga mengurangi ambiguitas antarkelas.

Simetri kemenangan per *seed* (10 berbanding 10) dalam penelitian ini menunjukkan bahwa tidak ada arsitektur yang memiliki keunggulan struktural untuk tugas klasifikasi ini; perbedaan yang teramati lebih disebabkan oleh variabilitas stokastik pelatihan daripada superioritas arsitektur.

3.4 Evaluasi per Kelas

Evaluasi per kelas dihitung dari akumulasi prediksi seluruh 20 *seed*, menghasilkan total 14.400 data uji per model (2.400 per kelas). Tabel 7 dan Tabel 8 menyajikan hasil evaluasi YOLOv8 Nano dan YOLO11 Nano secara berurutan.

Table 7. Hasil Evaluasi per Kelas pada YOLOv8 Nano (Akumulasi 20 Seed)

Kelas	Precision (%)	Recall (%)	F1-score (%)	Support
Early Blight	79,36	85,25	82,20	2.400
Healthy	94,68	95,71	95,19	2.400
Yellow Curl Virus	95,89	95,38	95,63	2.400
Mold Leaf	85,81	81,42	83,56	2.400
Septoria Leaf Spot	86,64	86,17	86,40	2.400
Late Blight	88,02	86,00	87,00	2.400
Rata-rata Macro	88,40	88,32	88,33	14.400

Table 8. Hasil Evaluasi per Kelas pada YOLO11 Nano (Akumulasi 20 Seed)

Kelas	Precision (%)	Recall (%)	F1-score (%)	Support
Early Blight	77,25	84,04	80,50	2.400
Healthy	95,76	94,21	94,98	2.400
Yellow Curl Virus	95,43	90,50	92,90	2.400
Mold Leaf	83,12	84,13	83,62	2.400
Septoria Leaf Spot	88,63	86,42	87,51	2.400
Late Blight	88,17	87,54	87,85	2.400
Rata-rata Macro	88,06	87,81	87,89	14.400

YOLOv8 Nano menunjukkan performa lebih kuat pada kelas *Healthy* (F1 = 95,19%) dan *Yellow Curl Virus* (F1 = 95,63%), dua kelas dengan karakteristik visual paling distingtif dalam dataset. YOLO11 Nano memperoleh *F1-score* yang sedikit lebih tinggi pada kelas *Mold Leaf* (83,62% vs 83,56%), *Septoria Leaf Spot* (87,51% vs 86,40%), dan *Late Blight* (87,85% vs 87,00%). Kelas *Early Blight* menjadi kelas yang paling sulit diklasifikasikan oleh kedua model, dengan *F1-score* 82,20% (YOLOv8n) dan 80,50% (YOLO11n). Dari sudut pandang agronomis, kebingungan antara *Septoria Leaf Spot* yang dikelola terutama dengan fungisida dan *Late Blight* yang mungkin memerlukan perlakuan sistemik membawa konsekuensi ekonomi dan lingkungan yang tidak dapat diabaikan, sehingga presisi per kelas penyakit sama pentingnya dengan akurasi agregat untuk penerapan sistem ini di lapangan.

3.5 Analisis Confusion Matrix

Confusion matrix kumulatif dari 20 *seed* mengungkapkan pola kesalahan yang konsisten pada kedua model. Pada YOLOv8 Nano, pasangan kesalahan terbesar adalah: *Early Blight* diprediksi sebagai *Mold Leaf* (160 kali), *Mold Leaf* diprediksi sebagai *Early Blight* (159 kali), dan *Late Blight* diprediksi sebagai *Early Blight* (151 kali). Pada YOLO11 Nano, kesalahan dominan adalah: *Early Blight* diprediksi sebagai *Mold Leaf* (191 kali), *Late Blight* diprediksi sebagai *Early Blight* (176 kali), dan *Yellow Curl Virus* diprediksi sebagai *Early Blight* (152 kali).

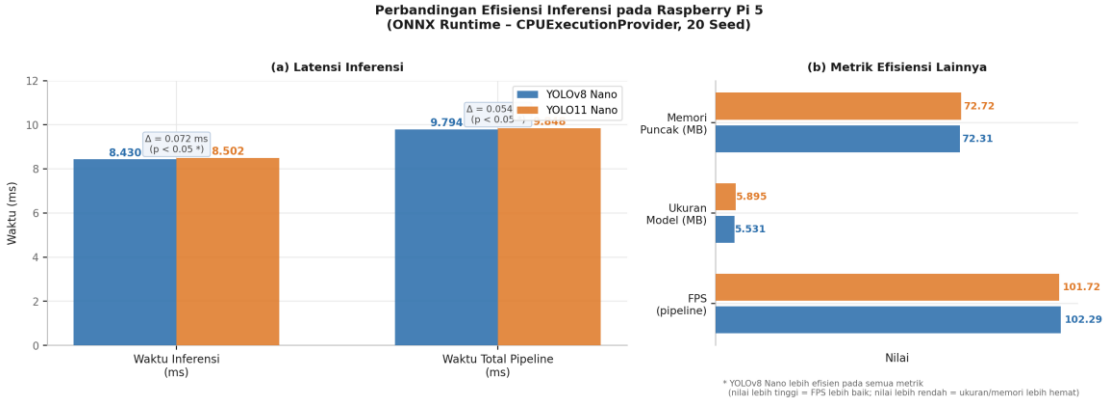
Pola ini konsisten dengan karakteristik visual penyakit daun tomat: kelas *Early Blight*, *Late Blight*, *Mold Leaf*, dan *Septoria Leaf Spot* sama-sama dapat menampilkan bercak atau perubahan jaringan pada permukaan daun yang secara visual terlihat serupa, terutama pada kondisi citra yang kurang tajam atau pada tahap awal perkembangan gejala. Das et al. [2] secara serupa mengidentifikasi kluster morfologis ini sebagai sumber utama kesalahan klasifikasi dalam ulasan mereka. Sebaliknya, kelas *Healthy* dan *Yellow Curl Virus* memiliki ciri visual yang lebih mudah dibedakan dari kelas penyakit berbercak, sehingga kedua model mampu mengklasifikasikannya dengan *F1-score* di atas 92%.

3.6 Efisiensi Inferensi pada Perangkat Edge

Tabel 9 melaporkan rata-rata metrik efisiensi dari 20 evaluasi *seed* pada Raspberry Pi 5. Gambar 4 menyajikan visualisasi perbandingan waktu inferensi dan total *pipeline*.

Table 9. Rata-rata Metrik Efisiensi Inferensi pada Raspberry Pi 5 (ONNX Runtime, CPU; n = 20 Seed)

Metrik	YOLOv8 Nano	YOLO11 Nano	Selisih (V8n−Y11n)	Wilcoxon p
Waktu inferensi (ms)	8,430	8,502	−0,072	0,0064 *
Waktu total pipeline (ms)	9,794	9,848	−0,054	0,0484 *
FPS (total pipeline)	102,29	101,72	+0,57	0,0484 *
Ukuran model (MB)	5,531	5,895	−0,364	N/A (statis)
Memori puncak (MB)	72,31	72,72	−0,41	N/A (statis)



Gambar 4. *Perbandingan waktu inferensi rata-rata dan waktu total pipeline YOLOv8 Nano (biru) dan YOLO11 Nano (oranye) pada Raspberry Pi 5. (a) Latensi inferensi dan total pipeline (ms); (b) FPS, ukuran model, dan penggunaan memori puncak.*

YOLOv8 Nano memperoleh waktu inferensi rata-rata lebih rendah (8,430 ms vs 8,502 ms), waktu total *pipeline* lebih rendah (9,794 ms vs 9,848 ms), dan FPS lebih tinggi (102,29 vs 101,72) secara konsisten pada 20 pasang *seed*. YOLOv8 Nano lebih cepat dari sisi waktu inferensi pada 14 dari 20 *seed* dan lebih cepat dari sisi total *pipeline* pada 13 dari 20 *seed*. Selain itu, YOLOv8 Nano juga memiliki ukuran model yang lebih kecil (5,531 MB vs 5,895 MB; selisih 6,2%) dan memori puncak yang lebih rendah (72,31 MB vs 72,72 MB).

Meskipun selisih absolut waktu inferensi sebesar 0,072 ms tergolong kecil untuk klasifikasi citra tunggal, relevansi praktisnya muncul pada skala besar: memproses 1.000 citra akan menghemat sekitar 72 ms, sebuah angka yang terasa dalam *pipeline* inspeksi otomatis berthroughput tinggi. Dimensi efisiensi yang lebih signifikan secara praktis adalah ukuran file model --- footprint 6,2% lebih kecil dari YOLOv8 Nano berdampak langsung pada penerapan di sistem tertanam yang dalam (*deep embedded systems*) di mana penyimpanan dibatasi hingga beberapa puluh megabyte. Penggunaan memori puncak yang lebih rendah (72,31 MB vs 72,72 MB) juga relevan untuk perangkat *edge* yang berbagi sumber daya dengan proses OS, driver sensor, dan *middleware* komunikasi, sebagaimana ditekankan oleh Singh dan Gill [9] serta Xu et al. [10] dalam ulasan mereka. Kedua model beroperasi jauh di bawah 100 ms per citra pada Raspberry Pi 5, mengonfirmasi kelayakan keduanya untuk aplikasi pemantauan lapangan secara *real-time*.

3.7 Hasil Uji Wilcoxon Signed-Rank

Tabel 10 menyajikan rangkuman lengkap hasil uji Wilcoxon Signed-Rank. Perbedaan akurasi antara kedua model tidak signifikan secara statistik (W = 75,5; p = 0,2706), mengonfirmasi bahwa tidak ada model yang dapat dinyatakan superior dari sisi klasifikasi berdasarkan eksperimen ini. Keunggulan efisiensi YOLOv8 Nano, sebaliknya, signifikan secara statistik: waktu inferensi rata-rata (W = 34,0; p = 0,0064) dan waktu total *pipeline* / FPS (W = 52,0; p = 0,0484) keduanya melewati ambang *alpha* = 0,05.

Table 10. Hasil Uji Wilcoxon Signed-Rank (Berpasangan, n = 20; Alpha = 0,05)

Metrik	Statistik W	p-value	Keputusan (alpha = 0,05)
Akurasi klasifikasi	75,5	0,2706	Tidak signifikan – H0 diterima
Waktu inferensi rata-rata	34,0	0,0064	Signifikan – YOLOv8n lebih cepat
Waktu total pipeline	52,0	0,0484	Signifikan – YOLOv8n lebih cepat
FPS total pipeline	52,0	0,0484	Signifikan – YOLOv8n lebih tinggi

Berdasarkan keseluruhan bukti yang terkumpul, YOLOv8 Nano direkomendasikan untuk klasifikasi penyakit daun tomat berbasis perangkat *edge* ketika akurasi dan efisiensi dipertimbangkan secara bersamaan. Rekomendasi ini tidak bersandar pada superioritas akurasi yang signifikan --- karena kedua model setara pada kriteria ini --- melainkan pada keunggulan konvergen dalam ukuran model yang lebih kecil, memori puncak lebih rendah, dan waktu inferensi yang secara statistik lebih singkat. Ketiganya secara kolektif mengurangi biaya operasional dan memperluas kompatibilitas perangkat keras ke spesifikasi yang lebih rendah. Sebaliknya, apabila sebuah *pipeline* yang ada sudah memanfaatkan fitur khusus YOLO11 seperti *head* deteksi atau segmentasi, mengadopsi YOLO11 Nano untuk komponen klasifikasi merupakan pilihan yang sah tanpa pengorbanan akurasi yang berarti.

3.8 Pembahasan

3.8.1 Interpretasi Hasil dan Jawaban atas Tujuan Penelitian

Penelitian ini menetapkan empat tujuan utama: membandingkan akurasi, menganalisis signifikansi statistik, mengevaluasi efisiensi pada perangkat *edge*, dan menentukan model yang lebih sesuai. Keempat tujuan tersebut terjawab secara terpadu. Dari sisi akurasi, YOLOv8 Nano (88,32%) memperoleh rata-rata yang sedikit lebih tinggi dibandingkan YOLO11 Nano (87,81%), namun perbedaan tidak signifikan secara statistik (p = 0,2706). Fakta bahwa masing-masing model unggul pada tepat 10 dari 20 *seed* memperkuat kesimpulan ini: tidak ada arsitektur yang menunjukkan dominasi struktural konsisten dalam tugas klasifikasi penyakit daun tomat. Dari sisi efisiensi, YOLOv8 Nano unggul secara statistik signifikan pada waktu inferensi dan total *pipeline*, serta memiliki ukuran model dan penggunaan memori yang lebih rendah secara deskriptif. Kombinasi performa akurasi yang setara dengan keunggulan efisiensi yang terverifikasi statistik menjadikan YOLOv8 Nano sebagai pilihan yang lebih direkomendasikan untuk implementasi pada perangkat *edge*.

3.8.2 Rasionalisasi Perbedaan Performa Berbasis Karakteristik Arsitektur

Perbedaan performa yang tipis dapat dijelaskan melalui karakteristik arsitektur. YOLOv8 Nano menggunakan modul C2f yang mengoptimalkan aliran gradien melalui koneksi silang antar-*stage*, sehingga mampu mengekstraksi fitur visual secara efisien pada skala model yang sangat kecil. YOLO11 Nano memperkenalkan blok C3k2 yang menggantikan C2f dengan mekanisme konvolusi *kernel* lebih kecil yang dimaksudkan untuk meningkatkan efisiensi parameter [13], [14], [15]. Meskipun perubahan arsitektur ini secara teoritis menjanjikan, pada skala nano dengan dataset penyakit daun tomat yang spesifik, manfaat tersebut tidak menghasilkan peningkatan akurasi yang signifikan. Sebaliknya, ukuran model YOLO11 Nano yang sedikit lebih besar (5,895 MB vs 5,531 MB) mengindikasikan kompleksitas arsitektur C3k2 yang lebih tinggi dan berkontribusi pada waktu inferensi yang sedikit lebih lambat pada perangkat CPU seperti Raspberry Pi 5, konsisten dengan temuan bahwa komputasi CPU tidak selalu memperlihatkan keunggulan yang sama seperti pada GPU untuk blok konvolusi yang lebih kompleks.

3.8.3 Perbandingan dengan Konsep Ilmiah Landasan Penelitian

Hasil penelitian ini sejalan dengan konsep teoretis yang mendasarinya. Prinsip klasifikasi citra berbasis CNN menyatakan bahwa model mempelajari hierarki fitur visual dari data latih [3], [4]. Kedua model berhasil mempelajari representasi tersebut, terbukti dari akurasi rata-rata di atas 87% dan *F1-score* tinggi untuk kelas distingtif seperti *Healthy* (F1 > 94%) dan *Yellow Curl Virus* (F1 > 92%). Adapun kesulitan pada kelas *Early Blight*, *Mold Leaf*, *Late Blight*, dan *Septoria Leaf Spot* konsisten dengan teori bahwa model CNN berbasis klasifikasi global citra akan kesulitan membedakan gejala pada kluster morfologis serupa karena jarak antar-kelas yang kecil dalam ruang fitur [2], [11]. Konsep komputasi *edge* yang menekankan latensi rendah dan kemandirian dari *cloud* [9], [10] juga terkonfirmasi: kedua model berjalan di atas 100 FPS pada

Raspberry Pi 5 dengan total *pipeline* di bawah 10 ms, memenuhi syarat kelayakan untuk respons *real-time* di lapangan pertanian.

3.8.4 Perbandingan dengan Penelitian Terdahulu

Temuan penelitian ini menunjukkan konvergensi sekaligus divergensi yang menarik dengan penelitian terdahulu. Ramos dan Sappa [7] melaporkan YOLO11 mengungguli YOLOv8 dalam akurasi *object detection* penyakit daun tomat, sedangkan penelitian ini menemukan kesetaraan statistik dalam akurasi *image classification*. Divergensi ini memperkuat argumen bahwa performa model tidak dapat digeneralisasi melampaui tugas, dataset, dan skenario pengujian yang spesifik. Majid dan Ariatmanto [8] melaporkan akurasi lebih tinggi untuk kedua model (YOLO11: 99,4%; YOLOv8: 98,5%), kemungkinan besar karena dataset yang mereka gunakan dikumpulkan dalam kondisi lebih terkontrol sehingga mengurangi ambiguitas antarkelas. Penelitian ini menggunakan dataset lapangan dari Pakistan [16] yang lebih menantang karena variasi pencahayaan dan sudut pengambilan gambar, menjelaskan akurasi absolut yang lebih rendah namun lebih representatif terhadap kondisi nyata [1], [17], [18]. Dalam konteks model ringan untuk *edge deployment*, penelitian ini mendukung temuan He dan Tong [6] serta Sun et al. [19] yang menekankan pentingnya evaluasi langsung pada perangkat target karena performa pada GPU tidak dapat diprediksi pada CPU, dan bahwa varian nano dari keluarga YOLO tetap kompetitif untuk skenario perangkat terbatas.

3.8.5 Kontribusi bagi Pengembangan Ilmu Pengetahuan

Penelitian ini memberikan beberapa kontribusi ilmiah yang saling melengkapi. Pertama, penelitian ini mengisi kesenjangan dalam literatur dengan menyediakan perbandingan langsung YOLOv8 Nano versus YOLO11 Nano yang spesifik untuk tugas *image classification* (bukan *object detection*) penyakit daun tomat. Kedua, protokol 20 *seed* yang ketat menghasilkan basis bukti statistik yang lebih kuat dibandingkan studi single-run, memberikan estimasi performa yang stabil dan andal. Ketiga, *profiling* efisiensi inferensi secara langsung pada perangkat *edge* nyata (Raspberry Pi 5) dengan format ONNX mengisi celah empiris yang belum dijembatani oleh penelitian terdahulu yang umumnya melaporkan kecepatan pada GPU atau *workstation*. Kontribusi metodologis juga terletak pada demonstrasi pentingnya pengujian statistik nonparametrik (Wilcoxon Signed-Rank) dalam evaluasi model *deep learning* pada skenario replikasi terbatas, di mana asumsi normalitas tidak dapat diverifikasi [11], [12].

3.8.6 Keterbatasan dan Peluang Penelitian Lebih Lanjut

Penelitian ini memiliki beberapa keterbatasan. Pertama, dataset yang digunakan berasal dari satu sumber (Pakistan) sehingga generalisasi ke kondisi pertanian lain memerlukan validasi tambahan [1], [17]. Kedua, pengujian efisiensi hanya dilakukan pada Raspberry Pi 5 dengan CPU; hasil inferensi dapat berbeda pada perangkat dengan NPU (*Neural Processing Unit*), GPU tersemat seperti NVIDIA Jetson, atau *runtime* alternatif seperti NCNN dan TensorFlow Lite [10], [20]. Ketiga, ekspor ke format ONNX tanpa kuantisasi INT8 belum dioptimalkan untuk latensi minimum. Keempat, penelitian ini berfokus pada *image classification* sehingga model tidak memberikan informasi lokasi area daun yang terinfeksi secara spasial. Berdasarkan keterbatasan tersebut, penelitian lanjutan disarankan untuk: (1) memperluas dataset ke berbagai wilayah geografis dan kondisi pertanian yang beragam; (2) membandingkan *runtime* inferensi alternatif seperti TensorRT pada NVIDIA Jetson dan NCNN pada platform *mobile*; (3) menerapkan kuantisasi INT8 pascapelatihan untuk kompresi model lebih lanjut; dan (4) memperluas *pipeline* ke arah *object detection* atau segmentasi instance agar sistem mampu melokalisasi wilayah daun yang terinfeksi secara spasial [19].

4. Simpulan

Penelitian ini membandingkan secara sistematis YOLOv8 Nano dan YOLO11 Nano untuk tugas klasifikasi citra penyakit daun tomat dalam protokol eksperimen 20 *seed* yang ketat, dengan evaluasi performa inferensi langsung pada platform *edge* Raspberry Pi 5. Kedua model berhasil mencapai rata-rata akurasi di atas 87% pada *benchmark* enam kelas, dengan YOLOv8 Nano di angka 88,32% dan YOLO11 Nano di angka 87,81%. Secara kritis, uji Wilcoxon Signed-Rank mengonfirmasi bahwa selisih akurasi 0,51 pp tidak signifikan secara statistik ($W = 75,5$; $p = 0,2706$), yang menetapkan kedua model sebagai fungsional setara dari sisi performa klasifikasi pada dataset yang digunakan. Pola kesalahan dominan yang terpusat pada kluster *Early Blight*, *Late Blight*, *Mold Leaf*, dan *Septoria Leaf Spot* yang secara visual serupa terjadi pada kedua

model, menunjukkan bahwa keterbatasan ini lebih merupakan fungsi kemiripan antarkelas pada tingkat dataset daripada kapasitas representasi yang spesifik pada salah satu arsitektur tertentu. Dari sisi efisiensi, YOLOv8 Nano menunjukkan keunggulan yang signifikan secara statistik dalam waktu inferensi rata-rata (8,430 ms vs 8,502 ms; p = 0,0064) dan metrik total *pipeline* (p = 0,0484), serta keunggulan deskriptif dalam ukuran file model (5,531 MB vs 5,895 MB) dan penggunaan memori puncak (72,31 MB vs 72,72 MB). Kedua model beroperasi di atas 100 FPS pada Raspberry Pi 5, mengonfirmasi kelayakannya untuk pemantauan lapangan secara *real-time*.

Secara keseluruhan, YOLOv8 Nano direkomendasikan sebagai pilihan yang lebih optimal untuk klasifikasi penyakit daun tomat berbasis perangkat *edge* ketika efisiensi, penghematan memori, dan portabilitas model menjadi pertimbangan utama, sekaligus mengakui bahwa YOLO11 Nano tetap merupakan alternatif yang dapat dipertanggungjawabkan apabila hanya akurasi semata yang menjadi kriteria seleksi. Penelitian selanjutnya disarankan mengatasi keterbatasan yang telah diidentifikasi: perluasan dataset ke berbagai wilayah geografis dan perangkat pencitraan yang beragam, *benchmarking* runtime inferensi alternatif seperti TensorRT pada NVIDIA Jetson dan NCNN pada platform *mobile*, penerapan kuantisasi INT8 pascapelatihan untuk kompresi model lebih lanjut, serta perluasan *pipeline* ke arah *object detection* atau segmentasi instance agar sistem mampu melokalisasi wilayah daun yang terinfeksi secara spasial dalam satu inferensi tunggal, sehingga nilai dukungan keputusan sistem pemantauan penyakit tomat secara otomatis dapat ditingkatkan secara substansial di lapangan.

Daftar Referensi

[1] M. Jelali, “Deep learning networks-based tomato disease and pest detection: A first review of research studies using real field datasets,” *Front. Plant Sci.*, vol. 15, p. 1493322, 2024, doi: 10.3389/fpls.2024.1493322.

[2] A. Das, F. Pathan, J. R. Jim, M. M. Kabir, and M. F. Mridha, “Deep learning-based classification, detection, and segmentation of tomato leaf diseases: A state-of-the-art review,” *Artificial Intelligence in Agriculture*, vol. 15, no. 2, pp. 192–220, 2025, doi: 10.1016/j.aiia.2025.02.006.

[3] C. Jackulin and S. Murugavalli, “A comprehensive review on detection of plant disease using machine learning and deep learning approaches,” *Measurement: Sensors*, vol. 24, p. 100441, 2022, doi: 10.1016/j.measen.2022.100441.

[4] T. Nyawose, R. C. Maswanganyi, and P. Khumalo, “A review on the detection of plant disease using machine learning and deep learning approaches,” *J. Imaging*, vol. 11, no. 10, p. 326, 2025, doi: 10.3390/jimaging11100326.

[5] Y. Sun, L. Ning, B. Zhao, and J. Yan, “Tomato leaf disease classification by combining EfficientNetV2 and a Swin Transformer,” *Applied Sciences*, vol. 14, no. 17, p. 7472, 2024, doi: 10.3390/app14177472.

[6] Z. He and M. Tong, “LT-YOLO: A lightweight network for detecting tomato leaf diseases,” *Computers, Materials & Continua*, vol. 82, no. 3, pp. 4301–4317, 2025, doi: 10.32604/cmc.2025.060550.

[7] L. T. Ramos and A. D. Sappa, “A comprehensive analysis of YOLO architectures for tomato leaf disease identification,” *Sci. Rep.*, vol. 15, p. 26890, 2025, doi: 10.1038/s41598-025-11064-0.

[8] M. A. K. Majid and D. Ariatmanto, “Comparative analysis of the performance of YOLO11 and YOLOv8 models in identification of diseases in tomato leaves,” *Bangkit Indonesia*, vol. 14, no. 2, pp. 63–72, 2025.

[9] R. Singh and S. S. Gill, “Edge AI: A survey,” *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 71–92, 2023, doi: 10.1016/j.iotcps.2023.02.004.

[10] Y. Xu, T. M. Khan, Y. Song, and E. Meijering, “Edge deep learning in computer vision and medical diagnostics: A comprehensive survey,” *Artif. Intell. Rev.*, vol. 58, no. 3, p. 93, 2025, doi: 10.1007/s10462-024-11033-5.

[11] M. Grandini, E. Bagli, and G. Visani, “Metrics for multi-class classification: An overview,” *arXiv preprint*, p. 2008.05756, 2020, doi: 10.48550/arXiv.2008.05756.

[12] O. Rainio, J. Teuho, and R. Klen, “Evaluation metrics and statistical tests for machine learning,” *Sci. Rep.*, vol. 14, p. 6086, 2024, doi: 10.1038/s41598-024-56706-x.

[13] J. Terven, D.-M. Cordova-Esparza, and J.-A. Romero-Gonzalez, “A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS,” *Mach. Learn. Knowl. Extr.*, vol. 5, no. 4, pp. 1680–1716, 2023, doi: 10.3390/make5040083.

[14] M. L. Ali and Z. Zhang, “The YOLO framework: A comprehensive review of evolution, applications, and benchmarks in object detection,” *Computers*, vol. 13, no. 12, p. 336, 2024, doi: 10.3390/computers13120336.
[15] 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, 2025, doi: 10.1016/j.jestch.2025.102161.
[16] M. N. Malik, S. H. Ali, and H. Ali, “Tomato Leaf Disease Classification Dataset in Pakistan,” 2026, *Mendeley Data, Pakistan*.
[17] A. Imtiaz, F. B. I. Swapnil, S. R. Masud, and D. Karmaker, “Tomato leaf dataset: A dataset for multiclass disease detection and classification,” *Data Brief*, vol. 60, p. 111520, 2025, doi: 10.1016/j.dib.2025.111520.
[18] M. K. Oni and T. T. Prama, “A comprehensive dataset of tomato leaf images for disease analysis in Bangladesh,” *Data Brief*, vol. 59, p. 111327, 2025, doi: 10.1016/j.dib.2025.111327.
[19] J. Sun, Z. Feng, J. Han, F. Xu, H. Zhang, and Y. Guo, “YOLO-PLNet: a lightweight real-time detection model for peanut leaf diseases based on edge deployment,” *Front. Plant Sci.*, vol. 16, Nov. 2025, doi: 10.3389/fpls.2025.1707501.
[20] Y. Majeed, M. O. Ojo, and A. Zahid, “Standalone edge AI-based solution for tomato diseases detection,” *Smart Agricultural Technology*, vol. 9, p. 100547, 2024, doi: 10.1016/j.atech.2024.100547.