ANALISIS KOMPARATIF EFISIENSI KOMPUTASI MODEL XGBOOST PADA INFRASTRUKTUR DOCKER CONTAINER DAN NATIVE OS UNTUK MENDUKUNG KEANDALAN LAYANAN PEMBELAJARAN DIGITAL BERBASIS KECERDASAN BUATAN

A Fadilah[1], Achmad Noercholis[2], Fransisca Sisilia Mukti[3]
Fakultas Teknologi dan Desain, Institut Teknologi dan Bisnis Asia Malang
[1]thariqdanzo99@gmail.com, [2]anoercholis@asia.ac.id, [3]ms.frans@asia.ac.id

ABSTRACT

The reliability of AI-based digital learning services depends heavily on the appropriate selection of machine learning model deployment infrastructure. Educational institutions commonly face computational resource constraints, choosing between a native OS and Docker Container, a critical factor affecting the responsiveness and availability of digital services that underpin academic activities. This study empirically compares the computational efficiency (CPU and RAM) and accuracy of the Extreme Gradient Boosting (XGBoost) algorithm in these two execution environments, using the CIC-IDS 2017 dataset from the Canadian Institute for Cybersecurity, comprising approximately 2.8 million records across 15 classes as a representative large-scale computational workload. A quantitative experimental method with real-time resource monitoring was applied to identical hardware. Results show that the Native OS environment achieved an average end-to-end (E2E) execution time of 209.72 seconds, approximately 9.4 times faster than Docker Container, which required an average of 1,965.83 seconds due to a 2-core CPU restriction. Native OS averaged 90.60% CPU and 403.01 MB RAM, while Docker averaged 24.41% CPU and 683.76 MB RAM with a +280.75 MB memory overhead. Critically, all model quality metrics Accuracy, Precision, Recall, and F1-Score were identical at 99.89% in both environments, confirming that the execution environment does not affect AI model quality. This study concludes that to support the reliability of AI-based digital learning services requiring real-time responses, using a native OS or providing unrestricted resource allocation in a Docker Container is the recommended deployment architecture for educational institutions.

Keywords: XGBoost, Docker Container, Native OS, Computational Efficiency, Digital Learning, Artificial Intelligence in Education, CIC-IDS 2017

ABSTRAK

Keandalan layanan pembelajaran digital berbasis kecerdasan buatan (AI) sangat bergantung pada pemilihan infrastruktur deployment model machine learning yang tepat. Institusi pendidikan umumnya menghadapi keterbatasan sumber daya komputasi, sehingga keputusan penggunaan Native OS atau Docker Container sebagai lingkungan eksekusi model AI berdampak langsung pada responsivitas dan ketersediaan layanan digital yang menopang aktivitas akademik. Penelitian ini bertujuan membandingkan secara empiris efisiensi komputasi (CPU dan RAM) serta akurasi algoritma Extreme Gradient Boosting (XGBoost) pada dua lingkungan eksekusi tersebut, menggunakan dataset CIC-IDS 2017 yang terdiri dari sekitar 2,8 juta rekaman dalam 15 kelas sebagai representasi beban kerja komputasi berskala besar. Metode eksperimen kuantitatif dengan pemantauan sumber daya secara

real-time diterapkan pada perangkat keras identik. Hasil penelitian menunjukkan bahwa Native OS mencatatkan waktu eksekusi end-to-end (E2E) rata-rata 209,72 detik sekitar 9,4 kali lebih cepat dibandingkan Docker Container yang memerlukan rata-rata 1.965,83 detik akibat pembatasan 2 core CPU. Native OS menggunakan rata-rata 90,60% CPU dan 403,01 MB RAM, sementara Docker menggunakan 24,41% CPU dan 683,76 MB RAM dengan overhead memori +280,75 MB. Seluruh metrik kualitas model, akurasi, presisi, recall, dan F1-Score, identik pada angka 99,89% di kedua lingkungan. Penelitian ini menyimpulkan bahwa untuk mendukung keandalan layanan pembelajaran digital berbasis AI yang membutuhkan respons real-time, penggunaan Native OS atau alokasi sumber daya penuh tanpa pembatasan pada Docker Container merupakan rekomendasi arsitektur deployment terbaik bagi institusi pendidikan.

Kata Kunci: XGBoost, Docker Container, Native OS, Efisiensi Komputasi, Pembelajaran Digital, Kecerdasan Buatan dalam Pendidikan, CIC-IDS 2017

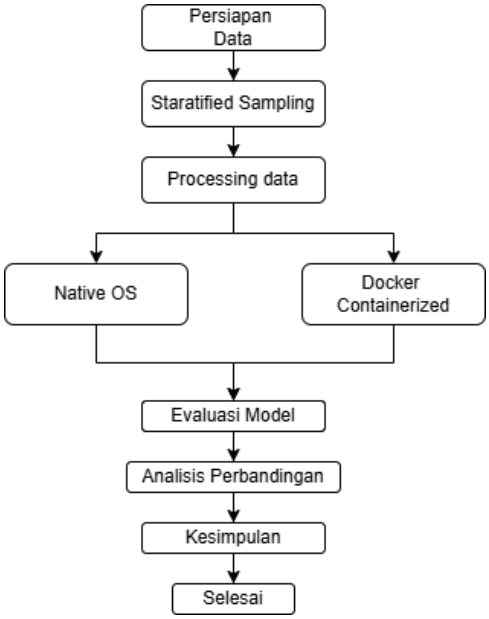
A. Pendahuluan

Perkembangan layanan digital di lingkungan pendidikan telah meningkatkan ketergantungan institusi pendidikan pada infrastruktur teknologi informasi. Berbagai platform *e-learning*, sistem analitik pembelajaran, dan layanan akademik digital kini menjadi tulang punggung aktivitas pendidikan modern. Menurut Adelabu et al. (2014), ketersediaan dan pemanfaatan infrastruktur *e-learning* merupakan faktor kunci dalam mendukung proses belajar mengajar yang efektif di perguruan tinggi, sekaligus menjadi tantangan utama bagi institusi di negara berkembang yang sering dihadapkan pada keterbatasan anggaran dan fasilitas teknologi informasi. Keandalan infrastruktur teknologi informasi merupakan fondasi keberhasilan sistem *e-learning*. Alsabawy et al. (2013) membuktikan secara empiris bahwa layanan infrastruktur TI yang mencakup aspek komunikasi, keamanan basis data, dan teknologi web memiliki pengaruh signifikan terhadap kesuksesan sistem *e-learning*. Temuan ini menegaskan bahwa pemilihan lingkungan *deployment* yang tepat merupakan faktor strategis yang secara langsung memengaruhi kualitas layanan digital pendidikan yang diterima pengguna akhir. Dalam ekosistem layanan digital pendidikan berbasis kecerdasan buatan (AI), algoritma *machine learning* memainkan peran yang semakin krusial. Wijaya & Yuniarto (2024) melalui tinjauan literaturnya menegaskan bahwa *machine learning* mampu menghasilkan sistem rekomendasi yang akurat melalui pemrosesan data dan identifikasi pola yang efisien menggunakan berbagai algoritma klasifikasi. Pendekatan komparatif antaralgoritma seperti yang dilakukan oleh Rofiq & Noercholis (2026) dalam mengevaluasi arsitektur CNN untuk klasifikasi citra menunjukkan bahwa metodologi studi komparatif dapat menghasilkan panduan teknis yang konkret bagi pengembang sistem berbasis AI.

Salah satu algoritma *machine learning* yang terbukti memiliki performa mutakhir (*state-of-the-art*) dalam klasifikasi data berskala besar adalah Extreme Gradient Boosting (XGBoost) (Chen & Guestrin, 2016). Noercholis & Riska (2025) dalam studi komparatif arsitektur model klasifikasi pada konferensi internasional mengonfirmasi bahwa evaluasi perbandingan lingkungan dan konfigurasi eksekusi model merupakan langkah penting sebelum melakukan *deployment* pada sistem nyata. Performa komputasi XGBoost sangat dipengaruhi oleh lingkungan eksekusi tempat model tersebut di-*deploy*, yang pada akhirnya menentukan responsivitas layanan digital pendidikan (Mahendra & Mukti, 2022). Teknologi kontainerisasi seperti Docker Container menjadi pilihan populer dalam pengembangan infrastruktur layanan digital pendidikan berbasis *cloud* karena menawarkan isolasi pustaka sistem yang fleksibel, portabilitas tinggi, serta efisiensi *overhead* yang secara teoritis sangat minim (Wen et al., 2023). Mukti & Junikhah (2019) dalam studi komparatif model propagasi jaringan nirkabel kampus menunjukkan bahwa pemilihan konfigurasi infrastruktur jaringan secara langsung memengaruhi kualitas layanan digital di lingkungan akademik, sebuah prinsip yang berlaku pula dalam konteks pemilihan lingkungan *deployment* model AI. Namun, pembatasan alokasi *core* fisik yang lazim diterapkan institusi pendidikan akibat keterbatasan kapasitas server dapat memicu penurunan efisiensi waktu eksekusi yang signifikan (Fredella & Rahman, 2025). Penelitian ini bertujuan melakukan analisis komparatif secara empiris mengenai efisiensi komputasi (CPU dan memori) serta akurasi algoritma XGBoost pada dua lingkungan eksekusi Docker Container dengan pembatasan 2 core CPU dan Native OS (Windows) menggunakan dataset CIC-IDS 2017 sebagai representasi beban kerja komputasi berskala besar. Melalui penelitian ini diharapkan dapat diberikan rekomendasi teknis berbasis data empiris bagi institusi pendidikan dan pengembang platform layanan digital pendidikan dalam menentukan arsitektur *model deployment* AI yang optimal guna memastikan keandalan dan responsivitas layanan pembelajaran digital pada infrastruktur dengan keterbatasan sumber daya komputasi.

B. Metode Penelitian

Penelitian ini menggunakan metode eksperimen kuantitatif dengan cara menguji dua lingkungan eksekusi secara langsung menggunakan perangkat keras yang identik. Variabel bebas adalah jenis lingkungan eksekusi (Native OS dan Docker), sedangkan variabel terikat mencakup tingkat akurasi model, total waktu pelatihan, serta rata-rata dan puncak penggunaan CPU maupun RAM. Pendekatan studi komparatif ini mengadopsi prinsip metodologi yang digunakan oleh Rofiq & Noercholis (2026) dan Noercholis & Riska (2025) dalam mengevaluasi performa model secara terstruktur pada kondisi eksekusi yang berbeda.



Gambar 1 Alur Penelitian Analisis Docker dan Native OS

1. Docker dan Kontainerisasi

Pembagian lingkungan eksperimen ke dalam dua arsitektur yang berbeda dilakukan untuk menguji sejauh mana fleksibilitas penyebaran model memengaruhi efisiensi penggunaan sumber daya fisik, yang merupakan pertimbangan kunci dalam *deployment* infrastruktur layanan digital pendidikan. Wen et al. (2023) membuktikan bahwa Docker secara teoritis menghasilkan *overhead* yang sangat minimal, berkisar 0–5% dibandingkan dengan sistem *bare-metal*. Namun, restriksi alokasi prosesor yang lazim diterapkan pada server layanan digital pendidikan berkapasitas terbatas dapat memicu pembengkakan waktu eksekusi. Prayoga & Latifah (2026) menguatkan hal ini melalui analisis kinerja CPU dan RAM pada lingkungan IaaS lokal, sementara Fredella & Rahman (2025) menjelaskan dampak aktivasi *virtual memory* terhadap latensi komputasi akibat keterbatasan RAM fisik.

2. XGBoost

XGBoost diperkenalkan oleh Chen & Guestrin (2016) sebagai sistem *tree boosting* yang terukur dan efisien, dipilih karena kemampuannya dalam menangani klasifikasi data berskala besar secara *scalable* melalui arsitektur pohon keputusan yang paralel (Gunawan et al., 2022). Relevansinya terhadap layanan digital pendidikan terletak pada karakteristik pemrosesan data yang merepresentasikan beban kerja sistem AI pendidikan secara *real-time*. AlZoman & Alenazi (2021) mengonfirmasi keunggulan algoritma klasifikasi berbasis ensemble pada pemrosesan dataset berskala besar, sementara Moeslim et al. (2025) mengonfirmasi keunggulan XGBoost dalam tugas klasifikasi pada lingkungan komputasi terbatas. Guna memaksimalkan akurasi, konfigurasi model diperkuat melalui *hyperparameter tuning* (Azizah et al., 2026; Firmansyah et al., 2026).

3. Dataset CIC-IDS 2017

CIC-IDS 2017 dari Canadian Institute for Cybersecurity digunakan sebagai representasi beban kerja komputasi berskala besar yang karakteristiknya serupa dengan pemrosesan *big data* pada layanan AI pendidikan: 2,8 juta rekaman, 15 kelas, dan membutuhkan klasifikasi cepat (Cherif & Kortebi, 2019). Kurniabudi et al. (2020) mengonfirmasi efektivitas dataset ini sebagai acuan evaluasi algoritma klasifikasi melalui analisis fitur berbasis *Information Gain*. Arqane et al. (2022) dan Wei & Shangguan

(2023) menegaskan keluasan penggunaan dataset ini dalam riset deteksi anomali berbasis *machine learning*. Pemilihan dataset CIC-IDS 2017 dalam penelitian ini didasarkan pada pertimbangan karakteristik komputasi yang merepresentasikan beban kerja sistem AI pendidikan, bukan pada relevansi domain kontennya. Fokus utama penelitian ini adalah mengukur efisiensi infrastruktur eksekusi model machine learning, sehingga parameter kritis yang dipertimbangkan adalah volume data, kompleksitas klasifikasi multi-kelas, dan distribusi kelas yang tidak seimbang, karakteristik yang identik dengan data log aktivitas pembelajaran digital skala besar. Evaluasi terhadap dataset pendidikan yang tersedia secara publik menunjukkan keterbatasan signifikan untuk keperluan benchmark infrastruktur komputasi. Dataset Open University Learning Analytics Dataset (OULAD) hanya memiliki 32.593 rekaman mahasiswa yang siap diklasifikasikan, Student Performance Dataset (UCI) memuat 650 rekaman, dan xAPI Educational Dataset hanya 480 rekaman seluruhnya jauh di bawah ambang batas yang memadai untuk menghasilkan perbedaan performa infrastruktur yang signifikan dan terukur. Oleh karena itu, CIC-IDS 2017 dipilih sebagai instrumen benchmark yang paling mendekati karakteristik beban kerja komputasi sistem AI pendidikan skala besar. Penggunaan stratified random sampling sebanyak 100.000 rekaman dilakukan untuk mempertahankan proporsi distribusi 15 kelas secara representatif, sekaligus mensimulasikan volume data yang realistis bagi sistem AI pendidikan pada institusi pendidikan tinggi skala menengah di Indonesia dalam satu periode akademik.

4. Spesifikasi Lingkungan

Pengujian dilakukan pada laptop berprosesor Intel Core i7-1165G7 generasi ke-11 dengan RAM 16 GB. Lingkungan pertama: skrip Python dieksekusi langsung pada Windows 11 Home versi 25H2 (Native OS). Lingkungan kedua: Docker Container dengan image Python:3.10-slim yang dibatasi melalui konfigurasi *docker-compose.yml*: maksimal 4 *core* CPU aktif 2 *core* saat eksekusi, dan 4 GB RAM merepresentasikan tipikal server layanan digital pendidikan dengan keterbatasan sumber daya, sebagaimana dikaji oleh Mukti & Junikhah (2019) dalam konteks infrastruktur jaringan kampus. Pembatasan alokasi 2 core CPU pada lingkungan Docker Container dipilih secara sengaja untuk merepresentasikan kondisi tipikal server layanan digital pendidikan di institusi dengan keterbatasan anggaran infrastruktur. Kondisi ini mencerminkan praktik umum perguruan tinggi skala menengah di Indonesia yang umumnya menggunakan layanan Virtual Private Server (VPS) dengan spesifikasi terbatas atau menerapkan resource sharing antar layanan digital dalam satu server fisik. Dengan demikian, konfigurasi ini bukan sekadar pembatasan teknis eksperimental, melainkan simulasi kondisi deployment nyata yang dihadapi mayoritas institusi pendidikan dalam mengoperasikan layanan AI berbasis cloud.

5. Preprocessing dan Konfigurasi Model

Tahap pra-pemrosesan mencakup pembersihan nilai *infinite* dan NaN, dilanjutkan dengan reduksi data melalui *stratified random sampling* sebanyak 100.000 rekaman untuk

mempertahankan distribusi 15 kelas (Kurniabudi et al., 2020). Model XGBoost dikonfigurasi dengan: *n_estimators*=150, *max_depth*=8, *learning_rate*=0,1, dan *tree_method*='hist'. Konfigurasi identik diterapkan pada kedua lingkungan untuk memastikan validitas komparasi (Azizah et al., 2026; Firmansyah et al., 2026).

Tabel 1. Distribusi Kelas Dataset CIC-IDS 2017

No	Label	Rekaman	%
1	BENIGN	2.359.286	83,44
2	DoS Hulk	231.073	8,18
3	PortScan	158.930	5,62
4	DDoS	128.027	4,53
5	DoS GoldenEye	10.293	0,36
6	FTP-Patator	7.938	0,28
7	SSH-Patator	5.897	0,21
8	DoS slowloris	5.796	0,21
9	DoS Slowhttptest	5.499	0,19
10	Bot	1.966	0,07
11	Web Attack – BF	1.507	0,05
12	Web Attack – XSS	652	0,02
13	Infiltration	36	0,001
14	Web Attack – SQLi	21	0,001
15	Heartbleed	11	0,000

Tabel 2. Konfigurasi Parameter Model XGBoost

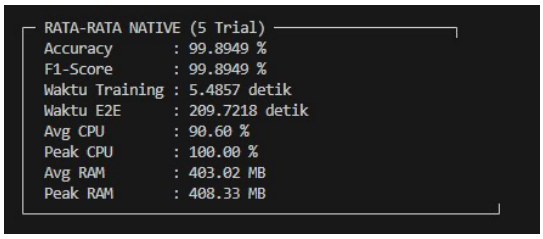
No	Parameter	Nilai	Fungsi
1	n_estimators	150	Jumlah pohon keputusan yang dibangun
2	max_depth	8	Kedalaman maksimum tiap pohon
3	learning_rate	0,1	Laju pembelajaran bertahap
4	tree_method	hist	Histogram untuk percepatan komputasi

C. Hasil Penelitian dan Pembahasan

Tahap awal penelitian berfokus pada pembersihan data dan penanganan *missing values* pada dataset CIC-IDS 2017 (Azizah et al., 2026). Proses rekayasa fitur kemudian diterapkan untuk mentransformasikan data mentah ke format yang siap dieksekusi oleh algoritma. Fitur redundan dieliminasi menggunakan seleksi berbasis *Information Gain* (Kurniabudi et al., 2020). Model XGBoost dilatih dengan *hyperparameter tuning* melalui *GridSearch Cross-Validation* (GSCV) untuk mencari kombinasi parameter terbaik dari nilai *learning rate*, *max depth*, dan *n_estimators*. Melalui konfigurasi parameter yang presisi, model mampu meminimalkan nilai *error* dan menghasilkan performa klasifikasi yang mutakhir (Firmansyah et al., 2026; Siswanto et al., 2023).

1. Hasil Pengujian Native OS (Windows)

Pengujian pada lingkungan Native OS dilakukan sebanyak 5 kali percobaan (*trial*) secara berurutan untuk memastikan konsistensi dan keandalan hasil. Setiap percobaan menjalankan proses *training* model XGBoost dari awal dengan data dan konfigurasi yang sama.



Gambar 2. Rata-rata Hasil Pengujian Native OS (5 Trial)

Berdasarkan hasil pada Gambar 2, model XGBoost yang dijalankan pada Native OS berhasil mencapai akurasi dan F1-Score sebesar 99,89%. Waktu *training* rata-rata tercatat 5,49 detik, sedangkan total waktu *end-to-end* (E2E) mencapai 209,72 detik. CPU rata-rata terpakai 90,60% dengan puncak 100%, sementara RAM rata-rata 403,02 MB. Waktu respons yang singkat ini mengindikasikan kemampuan Native OS mendukung keandalan layanan digital pendidikan berbasis AI secara *real-time*, sebagaimana ditekankan oleh Alsabawy et al. (2013) bahwa responsivitas infrastruktur TI merupakan faktor kritis bagi keberhasilan sistem e-learning.

Tabel 3. Detail Hasil Percobaan Native OS

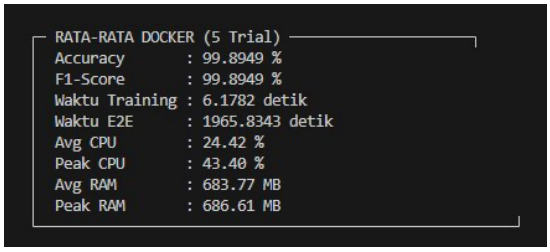
No	T.Train	E2E(det)	CPU	RAM	Akurasi
1	5,55 d	241,5	91,16 %	395 MB	99,89 %
2	5,25 d	214,9	90,63 %	441 MB	99,89 %
3	5,81 d	194,5	90,90 %	395 MB	99,89 %
4	5,47 d	190,8	90,12 %	387 MB	99,89 %
5	5,35 d	206,7	90,21 %	394 MB	99,89 %
Rata	**5,49 d**	**209,7**	**90,60 %**	**403 MB**	**99,89 %**

Tabel 3 memperlihatkan hasil yang sangat konsisten antarpercobaan. Variasi waktu E2E (190,8–241,5 detik) kemungkinan disebabkan oleh fluktuasi beban sistem operasi di *background*. Nilai akurasi dan F1-Score tetap stabil di 99,89% pada setiap percobaan, menguatkan temuan Ahda & Zainuddin (2019) bahwa model deterministik menghasilkan kualitas prediksi yang konsisten pada berbagai kondisi eksekusi.

2. Hasil Pengujian Docker Container

Pengujian Docker container dilakukan dengan konfigurasi identik dalam 5 percobaan. Pembatasan 2 *core* CPU merepresentasikan tipikal server layanan digital pendidikan berkapasitas terbatas. Kusbianto et al.

(2025) menjelaskan bahwa pembatasan jumlah *thread* pada sistem *multicore* langsung memengaruhi efisiensi CPU dan *throughput* komputasi secara keseluruhan.



Gambar 3. Rata-rata Hasil Pengujian Docker Container (5 Trial)

Hasil pada Gambar 3 menunjukkan bahwa model XGBoost tetap mencapai akurasi 99,89%, identik dengan Native OS, namun terdapat perbedaan dramatis pada waktu E2E, di mana Docker membutuhkan rata-rata 1.965,83 detik, hampir 9,4 kali lebih lambat. Konsumsi RAM rata-rata juga lebih tinggi, yaitu 683,77 MB. Dalam konteks keandalan layanan digital pendidikan, latensi ini berpotensi mengganggu ketersediaan dan responsivitas layanan yang bergantung pada pemrosesan AI secara *real-time*, sebagaimana ditekankan Adelabu et al. (2014) bahwa keandalan infrastruktur *e-learning* sangat menentukan efektivitas proses pembelajaran.

Tabel 4. Detail Hasil Percobaan Docker Container

Trial	Akurasi	T.Train	E2E (det)	CPU %	RAM(MB)
1	99,89 %	6,28 d	563,3	23,77	679,2
2	99,89 %	5,41 d	2166,6	24,92	713,8
3	99,89 %	6,29 d	2270,4	22,50	674,4
4	99,89 %	6,26 d	1399,3	24,92	675,4
5	99,89 %	6,62 d	3429,7	29,97	676,0

Hasil Tabel 4 menunjukkan bahwa model XGBoost tetap mencapai akurasi 99,89% identik dengan Native OS, namun terdapat variasi waktu E2E yang sangat besar (563,33–429,7 detik). Variasi ini disebabkan oleh *overhead* I/O sistem berkas kontainer dan ketidakstabilan alokasi sumber daya *host* (Wen et al., 2023). Konsumsi RAM rata-rata Docker (683,77 MB) lebih tinggi +280,75 MB daripada Native OS sebagai *overhead* lapisan *runtime* kontainer (Prayoga & Latifah, 2026).

3. Perbandingan Native OS vs Docker Container

Tabel 5. Perbandingan Metrik Evaluasi

Metrik	Native OS	Docker	Selisih
Akurasi / F1 (%)	99,89	99,89	0
Waktu Training (det)	5,49	6,17	+0,69
Waktu E2E (det)	209,72	1965,83	×9,4
CPU rata-rata (%)	90,60	24,41	-66,19
RAM rata-rata (MB)	403,01	683,76	+280,75

3.1 Akurasi dan Kualitas Model. Baik Native OS maupun Docker Container menghasilkan nilai Accuracy, Precision, Recall, dan F1-Score yang identik di angka 99,89%. Hal ini terjadi karena XGBoost bersifat deterministic dengan data dan parameter yang sama, model yang dihasilkan selalu identik (Ahda & Zainuddin, 2019). Wijaya & Yuniarto (2024) mengonfirmasi bahwa konsistensi kualitas model klasifikasi *machine learning* tidak dipengaruhi oleh lingkungan eksekusi, melainkan oleh kualitas data dan konfigurasi algoritma. Implikasi krusial bagi layanan digital pendidikan: kualitas prediksi AI tidak akan terdegradasi meskipun model di-*deploy* dalam lingkungan Docker yang dibatasi sumber dayanya.

3.2 Waktu Training. Perbedaan waktu *training* antara Native OS (5,49 detik) dan Docker (6,18 detik) hanya sebesar 0,69 detik relatif kecil dan dapat dikaitkan dengan *overhead* proses *startup* kontainer serta *marshalling* data antara *filesystem* kontainer dan *host*. Konsistensi ini juga terlihat pada studi Rofiq & Noercholis (2026) di mana waktu *training* antar-arsitektur yang dibandingkan tidak menunjukkan perbedaan signifikan meskipun dijalankan pada konfigurasi berbeda.

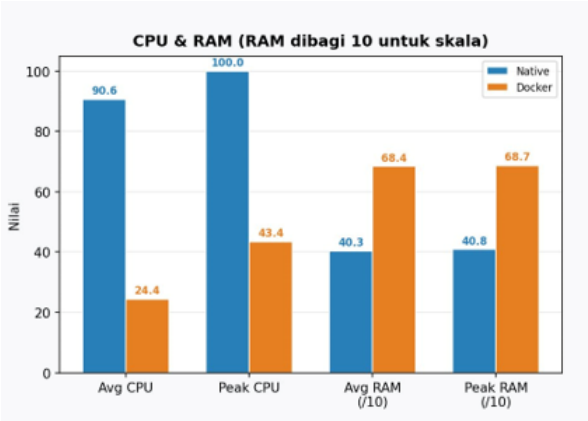
3.3 Waktu End-to-End dan Keandalan Layanan. Perbedaan paling dramatis terletak pada waktu E2E, di mana Docker membutuhkan 1.965,83 detik dibandingkan Native 209,72 detik, selisih 9,4 kali lebih lambat. Fredella & Rahman (2025) menjelaskan bahwa aktivasi mekanisme *virtual memory* seperti *paging* akibat keterbatasan RAM turut berkontribusi pada penurunan respons komputasi. Kusbianto et al. (2025) mengonfirmasi bahwa pembatasan *core* CPU secara langsung mengurangi *throughput* komputasi. Dalam konteks keandalan layanan digital pendidikan, latensi ini tidak dapat ditoleransi untuk fitur pembelajaran berbasis AI yang membutuhkan respons *real-time*.

3.4 Konsumsi CPU dan RAM. OS mencatatkan rata-rata CPU 90,60%, sedangkan Docker hanya 24,41% karena pembatasan *core*. Docker membutuhkan +280,75 MB lebih banyak dari Native OS sebagai *overhead* lapisan *runtime* kontainer. Mukti & Junikhah (2019) menegaskan bahwa efisiensi konfigurasi infrastruktur jaringan kampus berdampak langsung pada kualitas layanan digital civitas akademika. Bagi institusi pendidikan dengan server RAM terbatas, *overhead* ini perlu diperhitungkan dalam perencanaan kapasitas infrastruktur (Prayoga & Latifah, 2026; Pratama & Ahda, 2025).

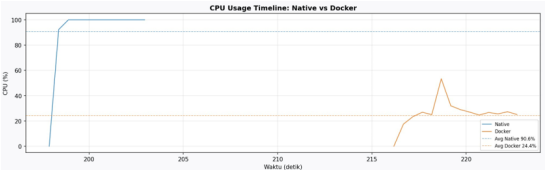
4. VIsualisasi Hasil Perbandingan

Bagian ini menjabarkan hasil visualisasi dari laporan perbandingan yang dihasilkan secara otomatis oleh sistem. Setiap grafik memiliki pesan tersendiri yang saling melengkapi dan menunjukkan adanya *trade-off* performa yang kentara antara lingkungan Native dan Docker, yang memiliki implikasi strategis bagi pemilihan arsitektur infrastruktur layanan digital pendidikan berbasis AI. Temuan ini konsisten dengan

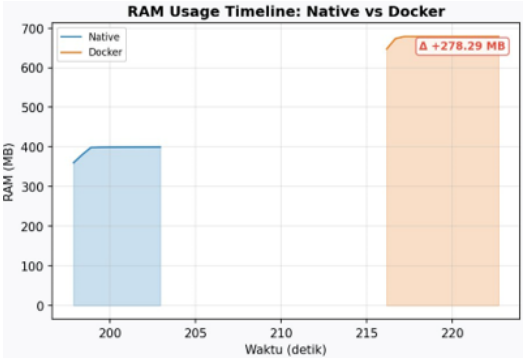
pernyataan Alsabawy et al. (2013) dan Adelabu et al. (2014) bahwa keandalan infrastruktur TI merupakan prasyarat keberhasilan layanan digital pendidikan.



Gambar 4. Grafik CPU & RAM pada Kedua Lingkungan



Gambar 5. CPU Usage Timeline: Native vs Docker



Gambar 6. RAM Usage Native vs Docker

Dari segi komputasi prosesor (Gambar 4 dan 5), Docker unggul karena mampu meredam penggunaan rata-rata CPU hingga hanya 24,4% (berbanding 90,6% pada Native) dan menjaga kestabilan grafik *timeline* tanpa mengalami *bottleneck* 100%. Namun efisiensi CPU pada Docker harus dibayar dengan konsumsi memori yang lebih boros (+280,75 MB) sebagaimana terlihat pada Gambar 6, serta waktu eksekusi yang jauh lebih panjang. Ketimpangan waktu eksekusi inilah yang menjadi pertimbangan utama dalam menentukan arsitektur infrastruktur yang tepat untuk mendukung keandalan layanan digital pendidikan berbasis AI (Wijaya & Yuniarto, 2024; Noercholis & Riska, 2025).

D. Kesimpulan

Penelitian ini telah berhasil menganalisis secara empiris dampak pemilihan lingkungan eksekusi terhadap efisiensi komputasi dan akurasi model XGBoost sebagai landasan rekomendasi teknis bagi pengembangan infrastruktur layanan digital pendidikan berbasis AI. Berdasarkan data empiris, lingkungan eksekusi terbukti tidak memberikan pengaruh negatif terhadap kualitas model. Nilai akurasi, presisi, recall, dan F1-Score teruji konsisten di angka 99,89% pada kedua lingkungan (Ahda & Zainuddin, 2019), memastikan bahwa kualitas layanan AI pendidikan tidak akan terdegradasi akibat pilihan infrastruktur *deployment*. Hal ini selaras dengan temuan Noercholis & Riska (2025) bahwa kualitas model klasifikasi bersifat stabil lintas konfigurasi eksekusi selama parameter dan data yang digunakan identik. Namun demikian, ketimpangan signifikan ditemukan pada dimensi efisiensi waktu pemrosesan dan manajemen memori. Native OS mencatatkan waktu E2E rata-rata 209,72 detik dengan konsumsi RAM efisien di angka 403,01 MB. Sebaliknya, Docker Container membutuhkan waktu

operasional hingga 1.965,83 detik dan alokasi memori yang melonjak menjadi 683,76 MB akibat pembatasan 2 *core* CPU (Kusbianto et al., 2025; Fredella & Rahman, 2025). Perbedaan waktu eksekusi sebesar 9,4 kali ini secara langsung berimplikasi pada responsivitas dan keandalan layanan digital pendidikan berbasis AI bagi pengguna akhir. Penelitian ini menyimpulkan bahwa untuk mendukung keandalan infrastruktur layanan pembelajaran digital berbasis AI yang membutuhkan respons *real-time*, penggunaan Native OS atau alokasi sumber daya penuh tanpa pembatasan pada Docker Container merupakan rekomendasi arsitektur *deployment* terbaik bagi institusi pendidikan (Wen et al., 2023). Sebagaimana ditegaskan oleh Adelabu et al. (2014) dan Alsabawy et al. (2013) bahwa ketersediaan dan keandalan infrastruktur merupakan kunci efektivitas sistem *e-learning*, dan dikuatkan oleh Mukti & Junikhah (2019) bahwa konfigurasi infrastruktur jaringan kampus secara langsung memengaruhi kualitas layanan digital akademik, maka restriksi sumber daya yang terlalu ketat harus dihindari agar tidak mengorbankan keandalan layanan AI yang menopang aktivitas akademik.

DAFTAR PUSTAKA

Adelabu, O. A., Adu, E. O., & Adjorgri, S. J. (2014). The Availability and Utilization of E-Learning Infrastructures for Teaching and Learning. Mediterranean Journal of Social Sciences, 5(23), 1348–1355.

Ahda, F. A., & Zainuddin, M. (2019). Prediksi Kepuasan Pelayanan Perpustakaan Menggunakan Algoritma Decision Tree (C4.5). Jurnal PDP, 10(2), 142–149.

Alsabawy, A. Y., Cater-Steel, A., & Soar, J. (2013). IT Infrastructure Services as a Requirement for E-Learning System Success. Computers & Education, 69, 431–451. https://doi.org/10.1016/j.compedu.2013.07.035

AlZoman, R. M., & Alenazi, M. J. F. (2021). A Comparative Study of Traffic Classification Techniques for Smart City Networks. Sensors, 21(14), 4677.

Arqane, A., Boutkhoum, O., Boukhriss, H., & El Moutaouakkil, A. (2022). Intrusion Detection System using Ensemble Learning Approaches: A Systematic Literature Review. IJOE, 18(13), 1–16.

Azizah, N. N., Utami, W. D., & Fanani, A. (2026). Algoritma XGBoost Untuk Analisis Prediksi Permintaan Kalibrasi di PT PAL Indonesia. JIKO (Jurnal Informatika dan Komputer), 10(1), 14–27.

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794). ACM.

Cherif, I. L., & Kortebi, A. (2019). On using eXtreme Gradient Boosting (XGBoost) Machine Learning Algorithm for Home Network Traffic Classification. In IEEE Wireless Days (WD), Manchester, UK.

Firmansyah, M., dkk. (2026). Penerapan Algoritma Xgboost dan Random Forest dalam Prediksi Uraian Resiko Proyek Pada Dinas XYZ. Jurnal Tekno Kompak, 20(2), 614–627.

Fredella, F., & Rahman, U. (2025). Penerapan Virtual Memory terhadap Kinerja CPU, GPU, dan Respons Multitasking pada Windows 10. Mars: Jurnal Teknik Mesin, Industri, Elektro dan Ilmu Komputer, 3(5), 168–178.

Gunawan, R., Handika, E. S., & Ismanto, E. (2022). Pendekatan Machine Learning Dengan Menggunakan Algoritma Xgboost (Extreme Gradient Boosting) Untuk Peningkatan Kinerja Klasifikasi

Serangan Syn. CoSciTech, 3(3), 453–463.

Kurniabudi, Sholikah, M., Maksum, A., Darmawidjadja, M. I., & Hanifah, N. (2020). CICIDS-2017 Dataset Feature Analysis With Information Gain for Anomaly Detection. IEEE Access, 8, 132911–132921.

Kusbianto, M. K., Al Akmal, M. A. K., & Yaqin, M. A. (2025). Pengaruh Jumlah Thread terhadap Efisiensi CPU pada Sistem Multicore. Jurnal Pustaka Data, 5(2), 412–417.

Mahendra, D. D., & Mukti, F. S. (2022). Sistem Deteksi dan Pengendalian Serangan Denial of Service pada Server Berbasis Snort dan Telegram-API. Techno.COM, 21(3), 511–522.

Moeslim, A. G. S., Firmansyah, E., & Sutara, B. (2025). Perbandingan Kinerja XGBoost dan IndoBERT untuk Klasifikasi Teks Kesehatan Bahasa Indonesia. DSI: Jurnal Data Science Indonesia, 5(2), 1–8.

Mukti, F. S., & Junikhah, A. (2019). Studi Komparatif Empat Model Propagasi Empiris dalam Ruangan untuk Jaringan Nirkabel Kampus. Jurnal Teknologi dan Sistem Komputer, 7(4), 154–160. https://doi.org/10.14710/jtsiskom.7.4.2019.154-160

Mukti, F. S., Junikhah, A., Putra, P. M. A., Soetedjo, A., & Krismanto, A. U. (2022). A Clustering Optimization for Energy Consumption Problems in Wireless Sensor Networks using Modified K-Means++ Algorithm. IJIES, 15(3), 355–364.

Noercholis, A., & Riska, S. Y. (2025). Comparative Study of NASNetLarge and EfficientNetV2L Architectures for Brain Tumor Classification Using CNN Methods. In 2025 9th International Conference on Electrical, Electronics and Information Engineering (ICEEIE) (pp. 1–6). IEEE. https://doi.org/10.1109/ICEEIE66203.2025.11253976

Pratama, W., & Ahda, F. A. (2025). Inovasi Agen AI Dalam Sistem Pencatatan Struk Digital Otomatis Berbasis n8n. JURNAL FASILKOM, 15(3), 551–561.

Prayoga, H. R., & Latifah, N. (2026). Analisis Kinerja CPU dan RAM pada VirtualBox untuk Validasi Lingkungan IaaS Lokal. Jurnal Sistem Informasi Galuh, 4(1), 107–119.

Rofiq, F., & Noercholis, A. (2026). Analisis Komparatif Arsitektur CNN untuk Klasifikasi Penyakit Daun Tebu Berbasis Transfer Learning. Jutisi: Jurnal Ilmiah Teknik Informatika dan Sistem Informasi, 15(2). https://doi.org/10.35889/jutisi.v15i2.3431

Siswanto, S., Dewi, M. U., Kholifah, S., Widhiati, G., & Aryani, W. (2023). Penggunaan Model Deep Learning Untuk Meningkatkan Efisiensi Dalam Aplikasi Machine Learning. JPSI, 1(4), 215–238.

Wei, Y., & Shangguan, M. (2023). A Review of Deep Learning Based Intrusion Detection Systems. Highlights in Science, Engineering and Technology, 56, 1–8.

Wen, L., Rickert, M., Pan, F., Lin, J., & Knoll, A. (2023). Bare-Metal vs. Hypervisors and Containers: Performance Evaluation of Virtualization Technologies for Software-Defined Vehicles. In IEEE Intelligent Vehicles Symposium (IV).

Wijaya, G. F., & Yuniarto, D. (2024). Tinjauan Penerapan Machine Learning pada Sistem Rekomendasi Menggunakan Model Klasifikasi. Populer: Jurnal Penelitian Mahasiswa, 3(4), 144–153. https://doi.org/10.58192/populer.v3i4.2798