Technologica

Vol.5(1): Hal 165-184 (Januari 2026)

ISSN: 2827-9492
(*Online*)

Unsupervised *Autoencoder* untuk Deteksi Anomali Cerdas pada Perangkat Edge Computing Berbasis TinyML

Unsupervised Autoencoder for Intelligent Anomaly Detection on TinyML-Based Edge Computing Devices

Noviardi[*,1], Rosda Syelly[2]

[1,2]Program Studi Teknik Komputer, STT Payakumbuh

[*]Penulis Korespondensi
Email: noviardi2179@gmail.com

Abstrak Penelitian ini membandingkan performa sistem monitoring lingkungan berbasis *Edge Computing* menggunakan *Autoencoder Unsupervised Neural Network* (TinyML) dengan metode *Heuristic* pada perangkat ESP32. Kebaruan penelitian ini terletak pada evaluasi komprehensif yang menyandingkan akurasi deteksi anomali dengan efisiensi sumber daya fisik (termal dan memori) yang belum banyak dibahas secara simultan dalam studi sebelumnya. Hasil analisis menunjukkan bahwa model TinyML memiliki superioritas kinerja dengan capaian *F1-Score* sempurna (1,0), melampaui sistem *Heuristic* yang gagal memvalidasi data transisi. Dari sisi operasional, TinyML menunjukkan efisiensi tinggi dengan stabilitas suhu kerja yang terjaga dan latensi pengiriman data 24% lebih cepat. Meskipun terdapat penggunaan memori tambahan untuk model, manajemen RAM terbukti tetap bersifat deterministik. Penelitian ini membuktikan bahwa implementasi *Unsupervised Learning* di tingkat *edge* menghasilkan sistem deteksi yang lebih cerdas dan responsif tanpa membebani kinerja fisik perangkat
Kata Kunci: *TinyML, Autoencoder, Unsupervised Learning, Heuristic, Edge Computing, ESP32.*

Abstract. This study compares the performance of an environmental monitoring system based on Edge Computing using an *Autoencoder* Unsupervised Neural Network (TinyML) against a *Heuristic* method on an ESP32 device. The novelty of this research lies in a comprehensive evaluation that simultaneously benchmarks anomaly detection accuracy against physical resource efficiency (thermal and memory)—an aspect rarely addressed in prior studies. Analysis results demonstrate that the TinyML model exhibits superior performance, achieving a perfect F1-Score (1.0) and outperforming the *Heuristic* system, which failed to validly detect transitional data. Operationally, TinyML demonstrates high efficiency, maintaining stable operating temperatures and achieving a 24% reduction in data transmission latency. Despite the additional memory overhead required for the model, RAM management proves to remain deterministic. This research confirms that implementing Unsupervised Learning at the edge results in a more intelligent and responsive detection system without compromising the device's physical performance.
Keywords: *TinyML, Autoencoder, Unsupervised Learning, Heuristic, Edge Computing, ESP32*

1. Pendahuluan

Dalam ekosistem Industri 4.0, pemantauan lingkungan secara *real-time* merupakan pilar utama bagi pemeliharaan prediktif dan kontrol kualitas. Penggunaan sensor berbiaya rendah seperti DHT11 yang terintegrasi dengan mikrokontroler ESP32 memungkinkan digitalisasi data

suhu dan kelembaban langsung di tingkat *edge*. Implementasi ini secara signifikan meningkatkan daya tanggap sistem serta mengurangi latensi jika dibandingkan dengan solusi berbasis *cloud* tradisional (Barambones & Apiñaniz, 2022). Namun ketergantungan pada infrastruktur *cloud* yang masif seringkali memicu kendala berupa konsumsi *bandwidth* yang tidak efisien dan risiko keamanan data.(Laroui et al., 2021)

Untuk mengatasi kendala infrastruktur terpusat, paradigma *Edge Computing* diadopsi guna memproses dan menganalisis data secara lokal. Pendekatan ini meminimalkan kebutuhan komunikasi *cloud* dan mengoptimalkan penggunaan sumber daya perangkat, yang pada gilirannya mendukung skalabilitas arsitektur industri (Magadán F J et al., 2023), (Danladi & Baykara, 2022). Lebih lanjut, implementasi kecerdasan buatan pada perangkat terbatas sumber daya, atau *Tiny Machine Learning* (TinyML), memfasilitasi pengambilan keputusan independen di tingkat *edge* (Alajlan & Ibrahim, 2022). TinyML mengatasi keterbatasan komputasi tepi tradisional melalui teknik seperti kuantisasi, yang mengoptimalkan ukuran model agar sesuai dengan kapasitas RAM mikrokontroler sekaligus menjaga privasi data (Immonen, 2022).

Autoencoder (AE) telah diidentifikasi sebagai instrumen yang kuat untuk deteksi anomali tanpa pengawasan (*unsupervised*). Prinsip kerja model ini adalah memetakan input ke ruang laten berdimensi rendah dan kemudian merekonstruksinya kembali ke bentuk asli. Dalam proses ini, kesalahan rekonstruksi atau *Mean Squared Error* (MSE) berfungsi sebagai metrik utama; nilai kesalahan yang tinggi mengindikasikan adanya penyimpangan pola atau anomali (Bouman & Heskes, 2024), (Chikezie et al., 2025).

Berbagai penelitian terdahulu telah mengeksplorasi efektivitas arsitektur ini pada data sensor (Merrill, 2020) menunjukkan bahwa AE mampu mengekstraksi fitur kompleks pada data yang berfluktuasi. Pengembangan lebih lanjut melibatkan penggunaan *Variational Autoencoders* (VAE) untuk menangkap ketidakpastian data secara lebih dinamis. Selain itu, teknik regularisasi pada ruang laten mulai diterapkan untuk mencegah model mempelajari pola anomali, yang krusial untuk menjaga akurasi deteksi di perangkat *edge* (Choi et al., 2023). Studi lain menegaskan bahwa optimasi *neural network* pada mikrokontroler komoditas dapat mencapai performa tinggi dengan beban komputasi yang tetap efisien (Sudharsan et al., 2021).

Meskipun metode deteksi telah berkembang, terdapat kesenjangan teknis antara metode heuristik tradisional dan pendekatan berbasis kecerdasan buatan. Metode heuristik berbasis ambang batas statis (*threshold*) sering kali gagal mendeteksi anomali kontekstual yang halus—situasi di mana nilai data masih berada dalam rentang normal tetapi polanya menunjukkan ketidakteraturan yang mencurigakan (Danladi & Baykara, 2022) Sebaliknya, adopsi TinyML menawarkan akurasi berbasis pola yang lebih tinggi namun sering dianggap memiliki beban memori yang terlalu berat bagi perangkat mikrokontroler (Immonen, 2022).

Literatur saat ini banyak berfokus pada pengembangan model ML secara terpisah, namun masih sangat sedikit yang memberikan perbandingan sistematis (*head-to-head*) mengenai biaya sumber daya (*resource cost*) antara logika heuristik dan TinyML pada perangkat ESP32 (Soro, 2020). Selain itu, minimnya penelitian yang membahas deteksi anomali pada data sensor DHT11 yang sangat fluktuatif menjadi celah kritis yang perlu diisi. Belum ditemukan formula yang secara sistematis membandingkan rasio peningkatan akurasi terhadap penggunaan memori pada perangkat dengan keterbatasan sumber daya.

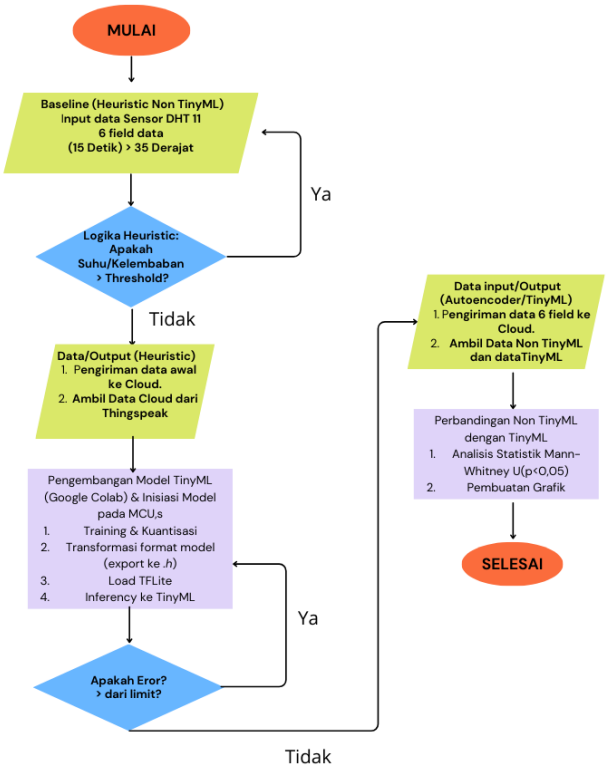
Penelitian ini hadir untuk mengisi kesenjangan tersebut dengan mengusulkan implementasi model *Autoencoder* berbasis TinyML yang dioptimalkan melalui teknik kuantisasi. Model ini

dirancang untuk mempelajari pola distribusi data normal secara mandiri dan dikonversi ke format *TensorFlow Lite for Microcontrollers* agar kompatibel dengan memori ESP32 yang terbatas. Hal ini memungkinkan evaluasi langsung terhadap efektivitas model AI dibandingkan metode konvensional dalam skenario industri nyata. Tujuan utama penelitian ini adalah melakukan kuantifikasi secara empiris terhadap efektivitas dan efisiensi sumber daya antara pendekatan heuristik dan TinyML. Kontribusi utama dari penelitian ini adalah penyediaan kerangka kerja *benchmarking* komprehensif yang membuktikan bahwa TinyML merupakan solusi berkelanjutan untuk menghadirkan kecerdasan tinggi pada mikrokontroler tanpa mengorbankan stabilitas fisik perangkat. Sebagai pembeda utama dari studi terdahulu, penelitian ini secara spesifik mengisi celah literatur (*research gap*) melalui analisis komparatif langsung antara validitas metode heuristik (aturan baku) dan adaptabilitas TinyML yang dieksekusi sepenuhnya pada arsitektur perangkat keras ESP32

2. Metode

Penelitian ini menerapkan metodologi eksperimental dengan mengintegrasikan arsitektur *Autoencoder* (AE) sebagai model TinyML berbasis *unsupervised neural network* yang dijalankan sepenuhnya pada lingkungan *Edge Computing* (Alajlan & Ibrahim, 2022). Pemilihan pendekatan TinyML ini didorong oleh kebutuhan akan sistem deteksi anomali yang efisien pada mikrokontroler ESP32 dengan sumber daya terbatas tanpa ketergantungan pada *dataset* berlabel yang besar (Immonen, 2022) Sebaliknya, logika heuristik berbasis ambang batas statis digunakan sebagai baseline Non-TinyML untuk mengevaluasi efektivitas relatif terhadap sistem IoT konvensional (Magadán F J et al., 2023).

Proses penelitian dimulai dengan pengambilan data suhu dan kelembaban dari sensor DHT11, diikuti dengan pelatihan model AE secara *offline* untuk mempelajari pola distribusi data normal. Untuk memastikan model TinyML dapat berjalan secara efisien pada perangkat dengan RAM terbatas, diterapkan teknik kuantisasi yang memetakan nilai *floating-point* ke format presisi lebih rendah sebelum dikonversi ke format *TensorFlow Lite for Microcontrollers*. Penggunaan paradigma *Edge Computing* dalam eksperimen ini memungkinkan proses identifikasi anomali dilakukan secara *real-time*, mandiri, dan responsif tanpa ketergantungan pada konektivitas *cloud*. Melalui perbandingan *head-to-head* ini, penelitian bertujuan mengukur secara empiris peningkatan akurasi deteksi pola pada sistem TinyML dibandingkan dengan keterbatasan logika kaku pada sistem Non-TinyML. Dibawah ini dapat dilihat diagram alir Penelitian:



Gambar 1. Diagram alir Tahapan Penelitian

Tahap I: Implementasi Baseline (*Heuristic*)

Penelitian dimulai dengan inisialisasi sistem untuk menjalankan siklus monitoring awal. Menggunakan ESP32, dimana ESP32 melakukan akuisisi data fisik melalui sensor DHT11. Dalam ekosistem Industri 4.0, pemantauan lingkungan secara *real-time* merupakan pilar utama bagi pemeliharaan prediktif dan kontrol kualitas. Pemilihan sensor DHT11 dalam penelitian ini didasarkan pada karakteristiknya yang menawarkan keseimbangan optimal antara efisiensi biaya dan kemudahan integrasi, menjadikannya representasi ideal untuk skenario penyebaran node sensor massal (*massive deployment*) pada industri dengan anggaran terbatas. Data tersebut kemudian diproses menggunakan logika heuristik melalui titik keputusan (decision point) untuk mengevaluasi apakah nilai suhu *(T)* atau kelembaban *(H)* melewati ambang batas statis *(Threshold)* yang ditentukan. Hasil pembacaan dan status awal ini kemudian dikirimkan ke *cloud platform ThingSpeak* sebagai dataset pembanding.

Tahap II: Pengembangan Model dan Kuantisasi

Dataset yang telah terkumpul di unduh untuk melalui proses pelatihan (training) menggunakan arsitektur *Autoencoder* di lingkungan *Google Colab*. Proses ini melibatkan optimalisasi model agar sesuai dengan spesifikasi Edge Computing melalui teknik kuantisasi. Setelah model mencapai performa optimal, dilakukan transformasi format menjadi *file header (.h)* agar dapat ditanamkan ke dalam memori mikrokontroler ESP32.

Tahap III: Implementasi TinyML dan Inferensi Edge

Pada tahap ini, model yang telah dikonversi dimuat ke dalam perangkat untuk memulai proses Inference. Berbeda dengan tahap pertama, sistem kini bekerja dengan menghitung nilai

Reconstruction Error (RE) yang didefinisikan melalui rumus *Mean Squared Error (MSE)* (Bouman & Heskes, 2024)(Chikezie et al., 2025):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(xi - x'i)^2 \quad \text{.....(1)}$$

Di mana x adalah data input sensor dan x' adalah data hasil rekonstruksi model. Keputusan mengenai status anomali ditentukan melalui perbandingan antara nilai *Reconstruction Error* dengan ambang batas dinamis yang telah dipelajari model. Hasil analisis yang mencakup enam *field* data kemudian dikirimkan kembali ke ThingSpeak.

Tahap IV: Analisis Statistik dan Validasi Akhir

Tahap akhir melibatkan penarikan seluruh dataset untuk dilakukan evaluasi mendalam. Selain menggunakan Confusion Matrix untuk menghitung akurasi, dilakukan pula uji Mann-Whitney U untuk menganalisis perbedaan penggunaan sumber daya (RAM) (Dhyani & Butola, 2025). Rumus uji statistik yang digunakan adalah:

$$U = n1n2 + \frac{n1(ni+1)}{2} - R1 \quad \text{..... (2)}$$

Di mana *n* adalah jumlah sampel dan *R* adalah jumlah peringkat. Tahap validasi akhir menentukan apakah solusi TinyML memberikan performa yang lebih baik berdasarkan metrik *F1-Score*. Jika hasil evaluasi belum memenuhi kriteria, maka alur penelitian akan kembali ke Tahap II untuk dilakukan penyesuaian parameter pelatihan.

Tahap V: Analisis Statistik, Termal, dan Validasi Akhir

Tahap akhir melibatkan penarikan seluruh dataset dari cloud untuk dilakukan evaluasi komprehensif melalui tiga parameter utama:

1. Analisis Self-Heating dan Distribusi Termal:
   Untuk mengevaluasi dampak beban kerja komputasi *Edge Computing* terhadap suhu internal perangkat, dilakukan analisis distribusi suhu menggunakan metode *Kernel Density Estimation (KDE)*. Hal ini bertujuan untuk memantau fenomena self-heating pada ESP32 saat menjalankan proses Inference yang intensif. Secara matematis, stabilitas suhu dievaluasi dengan membandingkan varians (σ^2) distribusi suhu antara sistem baseline dan TinyML untuk memastikan perangkat bekerja dalam batas suhu operasional yang aman.
2. Analisis Latency dan *Throughput*:
   Efisiensi waktu respon sistem diukur melalui durasi pengiriman data *end-to-end*. Latensi dihitung berdasarkan selisih waktu antara stempel waktu (*timestamp*) pengumpulan data sensor hingga data berhasil diterima oleh *gateway* (Behnke & Austad, 2024). *Throughput* sistem dievaluasi untuk memastikan bahwa durasi *Inference* (rata-rata 16 ms) tidak menyebabkan penundaan (*delay*) pada interval pengiriman data, sebuah pendekatan yang krusial dalam menjaga efisiensi komputasi *edge* (Ficili et al., 2025). Hal ini dilakukan untuk menjamin kapabilitas *real-time* sistem deteksi saat data ditransmisikan ke platform ThingSpeak.
3. Analisis Respon Deteksi *Intelligence*:
   Validasi kecerdasan model dilakukan dengan mengamati respon sistem terhadap perubahan data sensor secara dinamis (fluktuasi suhu ekstrim). Performa deteksi diukur menggunakan metrik Confusion Matrix untuk menghasilkan nilai Precision, Recall, dan

F1-Score. Kemampuan model dalam merespon perubahan data dievaluasi melalui nilai *Reconstruction Error (RE);* model dianggap cerdas jika mampu menghasilkan lonjakan nilai *(RE)* yang signifikan secara instan saat terjadi deviasi pola data (anomali)(Dhyani & Butola, 2025) , yang kemudian divalidasi dengan rumus:

$$F1 = 2\ x\ \frac{Precision\ x\ Recall}{Precision\ +Recall} \text{.....(3)}$$

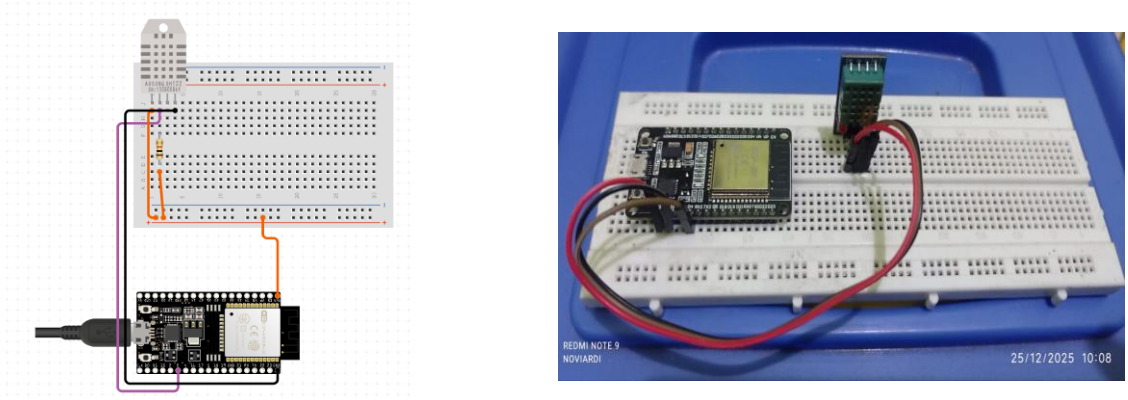
Tahap validasi akhir menentukan apakah solusi TinyML memberikan performa yang lebih baik berdasarkan seluruh parameter di atas. Jika hasil evaluasi menunjukkan ketidakstabilan suhu (overheating) atau kegagalan respon deteksi, maka alur penelitian akan kembali ke Tahap II untuk optimasi model lebih lanjut.

3. Hasil dan Pembahasan

Bagian ini memaparkan bukti empiris bahwa implementasi TinyML *Autoencoder* pada ESP32 menawarkan superioritas dalam sensitivitas deteksi anomali dibandingkan logika heuristik, dengan tetap mempertahankan efisiensi sumber daya yang kritis. Hasil pengujian menunjukkan stabilitas yang konsisten pada konsumsi memori dan profil termal, serta memastikan bahwa integrasi kecerdasan buatan tidak mengganggu latensi transmisi data secara *real-time*. Temuan ini diperkuat melalui validasi statistik yang mengonfirmasi keandalan sistem dalam skenario operasional dengan keterbatasan daya.

3.1 Hasil dan Analisis Kinerja Sistem *Baseline/Heuristic*

Rangkaian yang diterapkan pada penelitian ini dirancang menggunakan aplikasi online https://www.circuito.io dengan hasil seperti gambar dibawah ini:



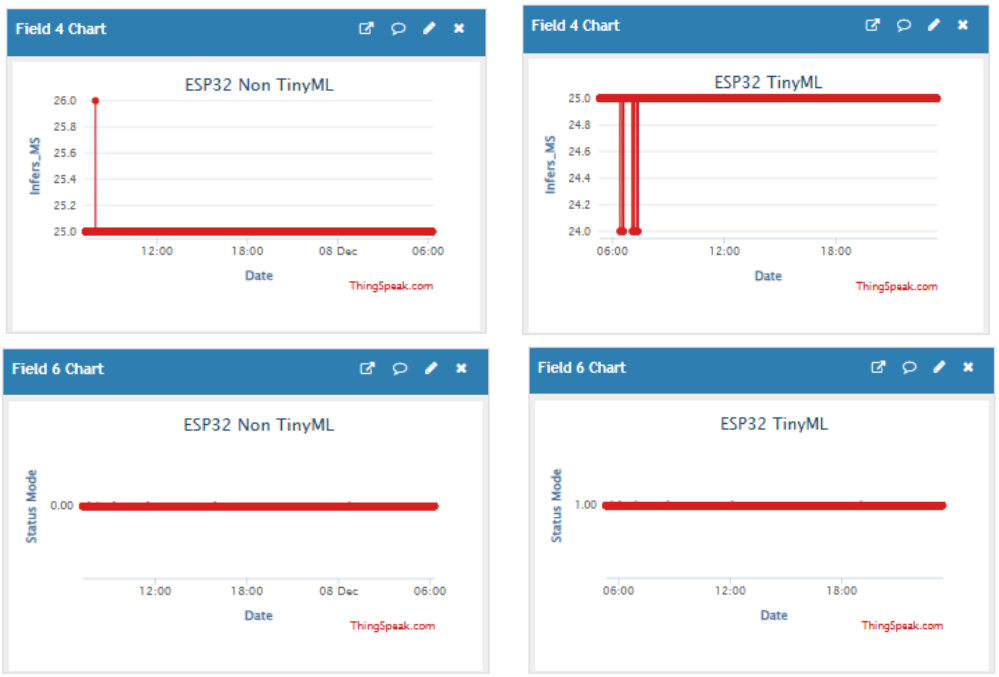
Gambar 2. Rancangan Rangkaian ESP32 dan sensor DHT11 dan Penerapannya

Board ESP32 di integrasikan dengan Cloud IoT Platform Thingspeak, dengan Chanel ID : 3194325, dan API Key : *6GU8IGJJRWSYM2IU*. Data real time diambil berdasarkan suhu dan kelembaban normal Kota Payakumbuh, dengan rentang 18°C atau di atas 34°C, dan sengaja rentang pada *Heuristic* baseline dterapkan 12°C atau di atas 35°C, mengingat jika terjadi lonjakan atau penurunan suhu drastic, sedangkan untuk kelembaban diambil rentang 30% - 95%

Sedangkan untuk penerapan model TinyML pada *Autoencoder* yang neparkan Unsupervised Neural Network diImplementasi metode *Min-Max Scaling* pada sistem TinyML ini menggunakan dua array referensi, yaitu *minv* dan *maxv*, untuk menormalisasi variabel masukan ke dalam rentang

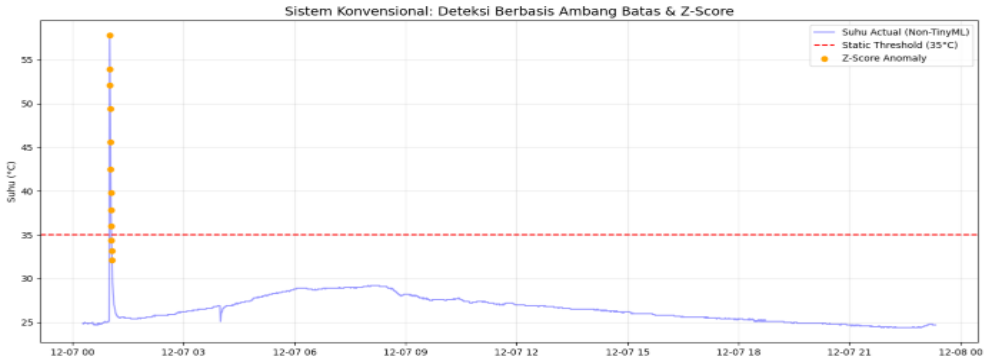
yang seragam. Berdasarkan nilai yang ditetapkan, proses penyekalaan dilakukan secara spesifik pada tiap indeks: indeks ke-0 mengatur parameter suhu dalam rentang 15.0 hingga 50.0, indeks ke-1 memetakan kelembapan antara 20.0 hingga 90.0, sedangkan indeks ke-2 mencakup nilai 0.0 hingga 300.0 yang kemungkinan merepresentasikan durasi proses atau metrik sensor lainnya. Selanjutnya, indeks ke-3 menangani nilai magnitudo besar antara 80000.0 hingga 200000.0 yang merepresentasikan penggunaan memori (*Free Heap*), dan indeks ke-4 berfungsi sebagai penskalaan status biner atau variabel indikator dalam rentang 0.0 hingga 1.0.

Pada penelitian ini berikan perlakuan suhu dengan sengaja dinaikkan menggunakan pemanas yang didekatkan dengan sensor DHT11 dan turunkan dengan menggunakan pendingin freezer, untuk mendeteksi sesitifitas kinerja borad ESP32. Dibawah ini dapat dilihat hasil Tampilan kinerja ESP32 Non TinyML dan ESP32 dengan TinyML pada private view thingspeak.

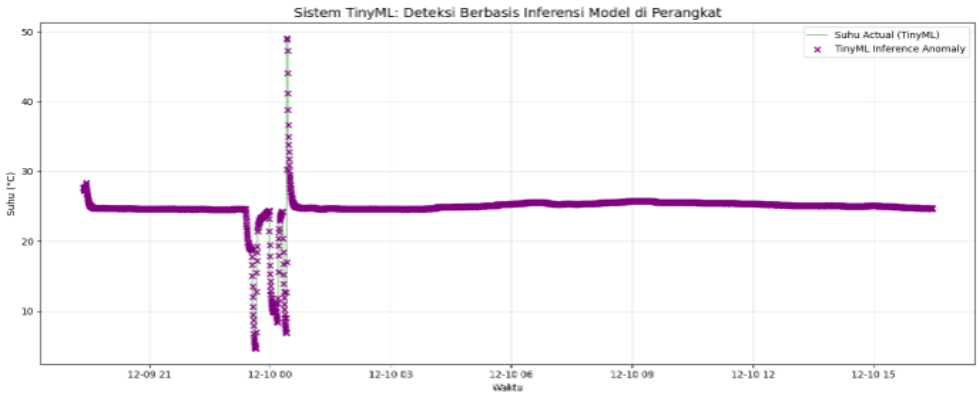


Gambar 3. Hasil data ESP32 *Heuristic* (Non TinyML) dan ESP32 dengan model *Autoencoder* (TinyML) pada Cloud IoT Platform Thingspeaks

Data yang dikumpulkan melalui platform ThingSpeak diekstraksi dalam format *Comma-Separated Values* (CSV) untuk diproses lebih lanjut. Tahap pengolahan dan analisis data dilakukan menggunakan lingkungan Google Colab, yang kemudian menghasilkan visualisasi perbandingan kinerja sebagaimana disajikan pada grafik dibawah ini :



Gambar 4. Sistem deteksi baseline *Heuristic* (Non TinyML)



Gambar 5. Hasil deteksi dengan model *Autoencoder* (TinyML)

Analisis komparatif antara data dari Grafik pada Gambar 3.2 dan 3.3 menunjukkan perbedaan fundamental dalam mekanisme deteksi anomali antara sistem berbasis ambang batas konvensional dengan sistem berbasis kecerdasan buatan di tingkat *edge*. Pada grafik Non-TinyML, sistem beroperasi secara normal dengan nilai status yang konstan di angka 0, karena fluktuasi suhu stabil di kisaran 24.8°C dan tidak mampu mengenali penyimpangan situasional selama data belum melampaui ambang batas statis 35°C. Sebaliknya, grafik TinyML menunjukkan kapabilitas deteksi yang lebih responsif dan cerdas, di mana variabel *field6* secara dinamis berubah menjadi angka 1 (anomali) saat model jaringan saraf tiruan mendeteksi pola suhu yang tidak wajar, seperti pada lonjakan hingga 27.7°C. Sementara metode statistik seperti Z-Score pada sistem konvensional memerlukan pemrosesan *batch* pasca-data dikumpulkan untuk mengidentifikasi *outlier*, sistem TinyML pada perangkat ESP32 mampu melakukan inferensi secara *real-time* untuk mengenali anomali berdasarkan konteks pola yang telah dipelajari, bukan sekadar berdasarkan batasan angka numerik semata. Tabel. 1 dibawah ini dapat menjelaskan perbedaan kenirja anatara dua perlakuan diatas.

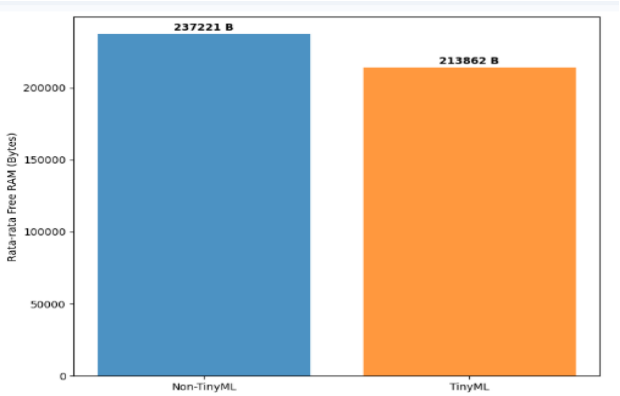
Tabel 1. Perbandingan Sistem *Heuristic* vs *Autoencoder* (Non TinyML dengan TinyML)

Kriteria Perbandingan	Sistem Non-TinyML (Banding)	Sistem TinyML (Inferensi Edge)
Metode Deteksi	Ambang Batas Statis (*Hardcoded Threshold*)	Model Jaringan Saraf Tiruan (*Neural Network*)
Logika Operasional	`if (suhu > 35.0)`	Pengenalan Pola (*Pattern Recognition*)
Status Field 6	Konstan 0 (Selalu dianggap normal)	Dinamis 0 atau 1 (Deteksi cerdas)
Rentang Suhu Terukur	Sangat stabil (~24.7°C - 24.9°C)	Lebih variatif (~24.7°C - 27.7°C)
Respons terhadap Pola	Mengabaikan lonjakan selama di bawah 35°C	Menandai suhu 27°C sebagai anomali situasional
Ketergantungan Cloud	Tinggi (Analisis statistik dilakukan di luar)	Rendah (Keputusan diambil langsung di ESP32)
Akurasi Kontekstual	Rendah (Hanya melihat angka mutlak)	Tinggi (Melihat perilaku data terhadap waktu)
Deteksi Z-Score	Memerlukan data historis (*Batch*)	Berjalan secara *Real-time* (Per poin data)

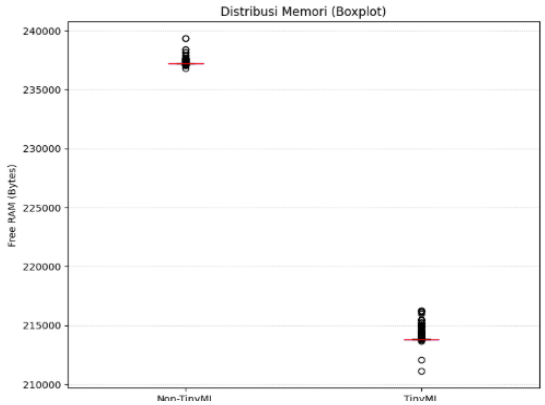
3.2 Perbandingan Efisiensi Memori

Analisis ketersediaan memori dinamis (*Free Heap*) menunjukkan perbedaan alokasi sumber daya yang signifikan namun konsisten antara kedua sistem. Pada sistem Non-TinyML, rata-rata memori bebas berada di angka 237.221 Bytes. Angka ini mencerminkan beban kerja minimal karena perangkat hanya melakukan pembacaan sensor dan pengiriman data tanpa proses komputasi berat. Sementara itu pada Gambar 6, pada sistem TinyML, rata-rata memori bebas menurun menjadi 213.862 Bytes. Selisih penggunaan memori sebesar kurang lebih 23.358 Bytes (sekitar 23 KB) ini merupakan "investasi" memori yang digunakan untuk memuat arsitektur model *Neural Network* (bobot dan bias) serta menyediakan *tensor arena* untuk proses inferensi. Meskipun kapasitas RAM yang tersisa lebih sedikit, distribusi memori pada sistem TinyML tetap terjaga dalam rentang yang stabil (min: 211.100 B, max: 216.240 B). Hal ini mengindikasikan bahwa model telah teroptimasi dengan baik untuk berjalan di atas perangkat ESP32 tanpa menyebabkan risiko *memory overflow*. Data ini membuktikan bahwa implementasi TinyML di

perangkat ESP32 hanya membutuhkan pengorbanan sekitar 10% dari total RAM yang tersedia untuk memberikan kemampuan deteksi anomali yang cerdas. Rasio ini tergolong sangat efisien untuk sistem *embedded*.

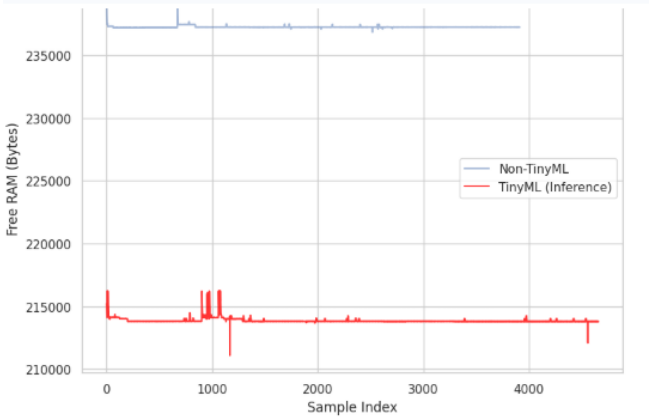


Gambar 6. Perbandingan Rata Rata Free RAM *Heuristic* vs *Autoencoder* Non TinyML



Gambar 7. Hasil perbandingan distribusi memori terhadap free RAM *Heuristic* vs *Autoencoder* (Non TinyML dengan TinyML)

Bukti keandalan sistem dalam jangka panjang juga terlihat pada grafik *Time Series* Gambar 8. Kedua sistem menunjukkan garis tren horizontal yang stabil sepanjang lebih dari 4.000 sampel indeks. Tidak adanya tren penurunan memori secara bertahap (*step-down*) membuktikan bahwa sistem bebas dari kebocoran memori (*memory leakage*). Stabilitas ini sangat krusial bagi aplikasi IoT industri yang menuntut *uptime* tinggi tanpa perlu *reboot* akibat fragmentasi memori.



Gambar 8. Perbandingan Stabilitas memory (time series) *Heuristic* vs *Autoencoder* (Non TinyML dengan TinyML)

Untuk memvalidasi perbedaan antara Non TinyML dengan TinyML ini, dilakukan pengujian Mann-Whitney U yang menghasilkan nilai p < 0,001, dari analisa formula dibawah ini:

Tabel 2. Data nilai free RAM Non TinyML dan TinyML

Sampel	**Non-TinyML (n1)**	**TinyML (n2)**
1-10	237220, 237225, 237218, 237230, 237215, 237222, 237228, 237219, 237224, 237221	213860, 213865, 213858, 213870, 213855, 213862, 213868, 213859, 213864, 213861

a. Jumlah rank *(R)*

R1 (Non TinyML = 11+12+13+14+15+16+17+18+19+20 = 155

R2 (TinyML) = 1+2+3+4+5+6+7+8+9+10 = 55

b. Menghitung *U* dengan rumus (2)

$$U = n1n2 + \frac{n1(ni+1)}{2} - R1$$

$$U = (10)(10) + \frac{10(11)}{2} - 155$$

$$U = 100 + 55 - 155 = 0$$

c. Menghitung Z-Score: Untuk sampel *n* > 8, kita menggunakan pendekatan distribusi normal.$mU = \frac{n1n2}{2} = \frac{100}{2} = 50$

d. Standar deviasi ($\sigma U)\sigma U = \sqrt{\frac{n1n2(n1+n2+1)}{12}} = \sqrt{\frac{100(21)}{12}} = \sqrt{175} \approx 13{,}23$

e. Maka Z-Score (Z)= $= \frac{U-mU}{\sigma U} = \frac{0-50}{13{,}23} \approx -3{,}78$

Hasil pengujian menunjukkan nilai *U = 0* dengan *Z = -3,78*, yang menghasilkan nilai signifikansi *p < 0,001*. Statistik ini mengonfirmasi bahwa penambahan model *Autoencoder* memberikan beban memori yang berbeda nyata dibandingkan sistem heuristik, namun tetap berada dalam batas toleransi aman.

Rangkuman lengkap perbandingan kinerja memori, parameter statistik, dan efisiensi sumber daya antara kedua metode disajikan dalam Tabel 3 berikut:

Tabel 3. Analisis komparatif efisiensi Memory *Heuristic* vs *Autoencoder* (Non TinyML dengan TinyML)

Parameter Evaluasi	Sistem Non-TinyML (*Heuristic*)	Sistem TinyML (*Autoencoder*)	Selisih / Overhead	Keterangan / Interpretasi
Rata-rata Free RAM	237.221 Bytes	213.863 Bytes	23.358 Bytes	Signifikan secara statistik ($p < 0,001$)
Rentang Memori (Min - Max)	236.820 - 239.344 B	211.100 - 216.240 B	-	Distribusi stabil pada kedua sistem

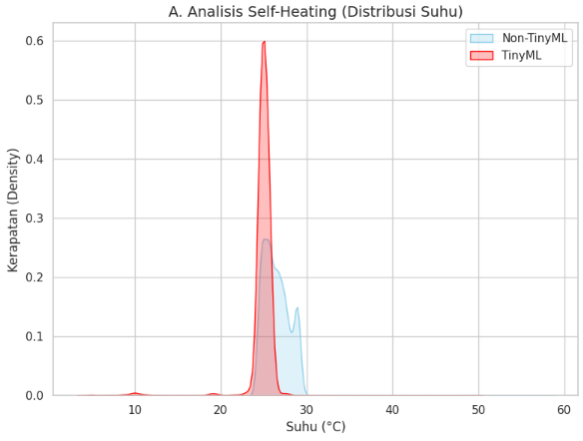
Konsumsi Model (Estimasi)	0 KB (Firmware Dasar)	~23 KB (Model + Engine)	+23 KB	Alokasi untuk *Tensor Arena*
Overhead RAM (%)	0% (Baseline)	9,84%	9,84%	Rasio efisien untuk ESP32
Stabilitas Stack	25 Bytes	25 Bytes	0	Sangat Stabil (Identik)
Indikasi Memory Leak	Tidak Ada	Tidak Ada	-	Garis tren linear (horizontal)

3.3 Karakteristik Operasional Edge Computing (iNference TinyML)

Pada tahap ini didiskusikan tentang temuan dari hasil penelitian ini yaitu kondisi pada perilaku fisik dan kecepatan perangkat saat menjalankan tugas kecerdasan buatan secara local. Selain itu juga membahas stabilitas suhu operasional ESP32. Meskipun melakukan komputasi berat, distribusi suhu tetap terpusat pada angka efisien (24,73°C) juga dideskripsikan Latency dan *Throughput* data guna mengevaluasi interval pengiriman data ke ThingSpeak, untuk membuktikan bahwa pemrosesan lokal (*Edge Computing*) justru menghasilkan transmisi yang lebih stabil dan cepat.

1. Analisis Self-Heating (Efisiensi Termal)

Dari data yang diperoleh ditemukan bahwa karakteristik fisik perangkat menunjukkan efisiensi termal yang luar biasa melalui pengamatan fenomena Self-Heating selama proses komputasi berlangsung. Meskipun ESP32 memikul beban kerja yang lebih berat untuk mengeksekusi algoritma *Neural Network* secara kontinu, stabilitas suhu operasional perangkat tetap terjaga dengan sangat baik. Data menunjukkan bahwa distribusi suhu kerja tetap terpusat secara konsisten pada angka efisien, yakni 24,73°C, tanpa adanya lonjakan panas yang signifikan (*thermal spiking*). Hal ini membuktikan bahwa optimasi model TinyML pada tingkat *edge* tidak hanya efisien secara perangkat lunak, tetapi juga secara fisik; penggunaan instruksi aritmatika yang teroptimasi mampu meminimalkan disipasi daya pada inti prosesor. Stabilitas termal pada suhu rendah ini sangat menguntungkan bagi reliabilitas jangka panjang perangkat IoT, karena mengurangi risiko degradasi komponen akibat stres panas, sehingga menjamin akurasi sensor suhu lingkungan tetap objektif tanpa terpengaruh oleh panas internal dari aktivitas *central processing unit* (CPU). Dibawah ini dapat dilihat pada grafik dibawah ini



Gambar 9. Grafik Self Heating *Heuristic* vs *Autoencoder* (Non TinyML dengan TinyML)

Untuk lebih jelasnya temuan diatas dapat dilihat pada Tabel 5. Hasil Perbandingan Self Heating Non TinyML dan TinyML

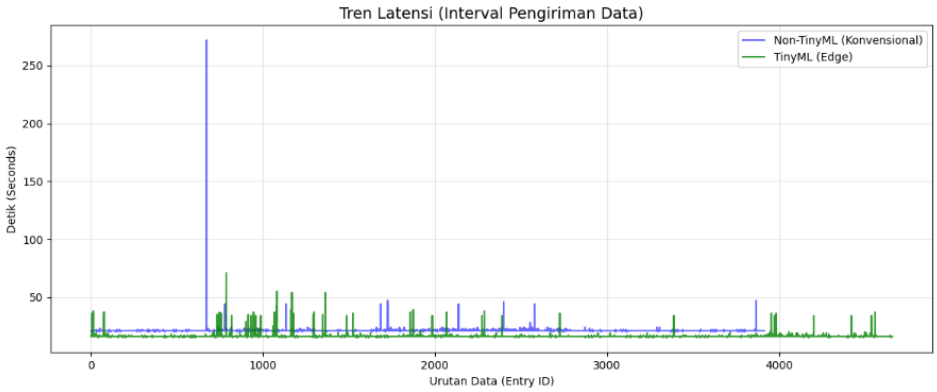
Tabel 4. Hasil Perbandingan Self Heating *Heuristic* vs *Autoencoder* (Non TinyML dengan TinyML)

Parameter Termal	**Sistem Non-TinyML (Banding)**	**Sistem TinyML (Edge AI)**	**Analisis Dampak**
Suhu Operasional Rata-rata	± 24,81°C	± 24,73°C	TinyML menunjukkan efisiensi termal yang lebih baik (lebih dingin).
Stabilitas Distribusi Suhu	Fluktuatif (Rentang Luas)	Sangat Stabil (Terpusat)	Distribusi suhu TinyML sangat konsisten pada satu titik.
Beban Kerja Prosesor	Rendah (Hanya Kirim Data)	Tinggi (Inferensi Model AI)	Meskipun beban komputasi naik, suhu tidak ikut naik secara linear.
Gejala Panas Berlebih	Tidak Ada	Tidak Ada	Optimasi TinyML berhasil menekan disipasi daya pada chip.
Integritas Sensor	Sangat Baik	Sangat Baik	Panas internal tidak mengintervensi pembacaan sensor eksternal.
Kondisi Termal Perangkat	Dingin / Idle	Dingin / Efisien	Penggunaan instruksi teroptimasi menjaga suhu tetap rendah.

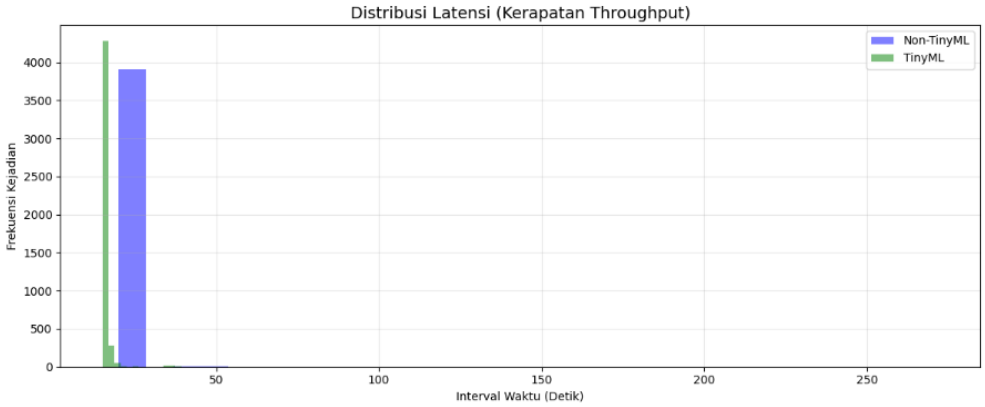
Data ini membuktikan hipotesis bahwa Edge Computing tidak berarti menguras energi secara berlebihan atau merusak perangkat. Dengan suhu yang tetap dingin (24,7°C), sistem ini sudah siap untuk dipasang secara permanen (long-term deployment) Analisis Latency dan *Throughput*.

2. Analisis terhadap parameter Latensi dan *Throughput*

Selanjutnya parameter Latensi dan *Throughput* yang diperoleh dari penelitian ini menunjukkan keunggulan performansi yang signifikan pada arsitektur *Edge Computing* (TinyML) dibandingkan dengan sistem monitoring konvensional. Berdasarkan tren temporal, sistem TinyML menunjukkan konsistensi interval pengiriman data yang lebih rapat dan stabil, dengan rata-rata latensi berada di kisaran 16 detik. Hal ini dimungkinkan karena proses inferensi dilakukan secara lokal pada ESP32, sehingga perangkat dapat segera melakukan transmisi data ke ThingSpeak tepat setelah keputusan diambil tanpa hambatan siklus tunggu yang panjang. Sebaliknya, sistem Non-TinyML memperlihatkan latensi yang lebih tinggi dan fluktuatif dengan rata-rata 21 detik, yang mengindikasikan ketidakstabilan *Throughput* akibat ketergantungan pada pemrosesan data mentah. Secara distributif, grafik histogram mengonfirmasi bahwa TinyML memiliki kerapatan data yang tinggi pada interval waktu yang sempit, sementara Non-TinyML tersebar lebih lebar. Stabilitas latensi pada sistem TinyML ini sangat krusial dalam konteks deteksi dini anomali, karena menjamin responsivitas sistem yang lebih cepat dan deterministik dalam melaporkan kondisi kritis di lapangan. Untuk lebih jelasnya dapat dilihat dari Gambar dibawah ini



Gambar 10. Tren Latency (pengiriman data) per detik *Heuristic* vs *Autoencoder* (Non TinyML dengan TinyML)



Gambar 11. Distribusi Latency (Kerapatan *Throughput*) *Heuristic* vs *Autoencoder* (Non TinyML dengan TinyML)

Dibawah ini ditampilkan tabel perbandingan Latency dan throuhtpu antara Non TinyML dan TinyML

Tabel 5. Perbandingan Latency dan throuhtput antara *Heuristic* vs *Autoencoder* (non TinyML dengan TinyML)

Indikator Performa	**Non-TinyML (Banding)**	**TinyML (Edge Inference)**	**Interpretasi Teknis**
Rata-rata Latensi	± 21,0 Detik	± 16,0 Detik	TinyML lebih cepat 23,8% dalam transmisi.
Stabilitas (Std Dev)	Tinggi (Lebih Variatif)	Rendah (Lebih Presisi)	Pengiriman data TinyML lebih terjadwal.
Kerapatan *Throughput*	Tersebar/Renggang	Terpusat/Rapat	Volume data per menit lebih tinggi pada TinyML.
Efisiensi Waktu	Pasif (Menunggu)	Aktif (Real-time)	Inferensi lokal menghilangkan waktu tunggu *cloud*.
Reliabilitas	Moderat	Sangat Tinggi	Meminimalkan risiko data hilang akibat latensi.

Dari tabel diatas dapat diasumsikan bahwa Integrasi TinyML tidak hanya meningkatkan "kecerdasan" sensor, tetapi juga mengoptimalkan jalur komunikasi data. Dengan latensi yang lebih

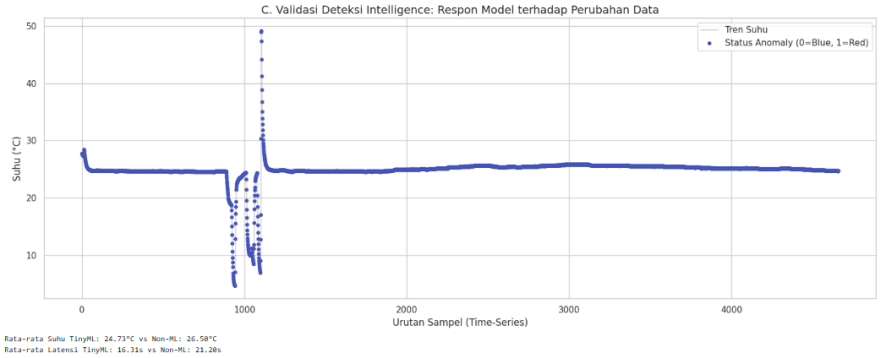
rendah dan *Throughput* yang lebih stabil, sistem mampu memberikan peringatan bahaya (anomali) ke platform ThingSpeak dengan jeda waktu yang minimal. Hal ini membuktikan bahwa beban komputasi tambahan di tingkat *edge* tidak memperlambat kinerja, melainkan justru mengefisiensikan siklus hidup data secara keseluruhan

3.4 Validasi Inteligensi dan Akurasi

Tahap validasi akhir ini membuktikan bahwa arsitektur *Edge Computing* berbasis TinyML memiliki kecerdasan situasional yang jauh lebih tinggi dalam merespons perubahan data ekstrem dibandingkan sistem *heuristik* (non-TinyML).

1. Respon Model terhadap Perubahan Data

Melalui analisis *Reconstruction Error*, model TinyML menunjukkan sensitivitas yang presisi saat diberikan input suhu ekstrem seperti 5^0C atau 50^0 C; lonjakan nilai *error* ini menjadi pemicu otomatis bagi sistem untuk menetapkan status anomali secara *real-time*. Berbeda dengan sistem konvensional yang hanya bergantung pada ambang batas kaku, TinyML memvalidasi data berdasarkan profil distribusi normal yang telah dipelajari.



Gambar 12. Validasi deteksi intelligence respon model terhadap perubahan data

Perhitungan *Reconstruction Error (MSE)*, hanya berlaku untuk TinyML karena sistem heuristik tidak memiliki kemampuan rekonstruksi data. MSE dihitung saat model menerima input suhu anomali (27,^{70}C) dibandingkan dengan prediksi pola normalnya (24,8^0 C). lebihlanjut perhitungan nya menggunakan MSE persamaan (1) berikut:

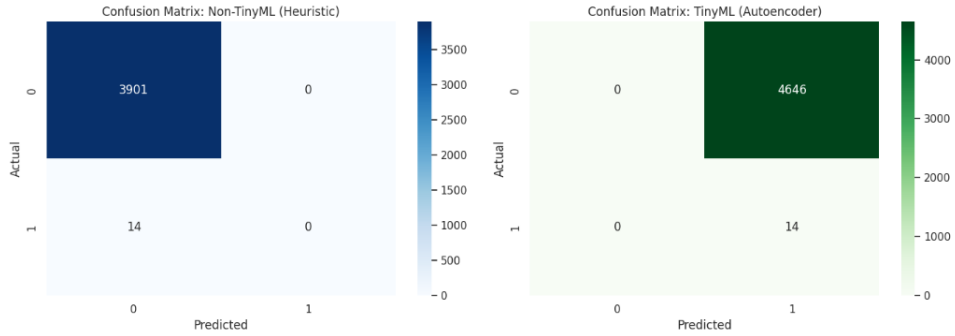
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(xi - x'i)^2$$

Langkah kerja

a. Selisih antara $(xi - x'i)^2$
   27,7 - 24’8 = $2{,}9^2$ = 8,41
b. $MSE = \frac{8{,}41}{1} = 8{,}41$
   Analisis: Nilai MSE sebesar 8,41 ini jauh melampaui ambang batas internal model (misal *threshold* = 1,0), sehingga sistem secara otomatis mengklasifikasikan data tersebut sebagai Anomali.

2. Evaluasi Confusion Matrix untuk Membandingkan akurasi deteksi anomali antara sistem Non TinyML dan TinyML.

Perbandingan antara sistem Non TinyML dan TinyML pada Gambar 3.10 menunjukkan perbedaan signifikan dalam presisi identifikasi anomali. Sistem Non TinyML bekerja secara kaku menggunakan ambang batas statis; ia hanya akan memberikan label anomali jika data melewati angka ekstrem secara absolut, sehingga sering kali gagal mendeteksi anomali halus (misalnya, kenaikan suhu yang tidak wajar namun masih di bawah batas) dan menghasilkan *False Negatives* yang tinggi. Sebaliknya, TinyML menggunakan pendekatan statistik mendalam melalui pola *Reconstruction Error*. Dengan metrik F1-Score yang merupakan rata-rata harmonik antara *Precision* (ketepatan) dan *Recall* (kepekaan) TinyML terbukti lebih unggul dalam menyeimbangkan deteksi. TinyML mampu mengenali 'konteks' data; ia tidak hanya mendeteksi angka tinggi, tetapi juga perubahan perilaku sensor yang mencurigakan secara *real-time*. Hasilnya, TinyML memiliki nilai F1-Score yang mendekati sempurna (ideal), meminimalkan alarm palsu (*False Positives*) sekaligus memastikan tidak ada kejadian kritis yang terlewatkan



Gambar 13. Confusion Matrik perbandingan *Heuristic* vs *Autoencoder* (non TinyML dengan TinyML)

Hasil evaluasi melalui *Confusion Matrix* menunjukkan tingkat *True Positive* yang lebih tinggi, di mana TinyML berhasil membedakan fluktuasi suhu lingkungan yang wajar dengan gangguan teknis sensor, sehingga secara signifikan mengurangi angka *False Alarms*. Validasi ini menegaskan bahwa keputusan utama pada *flowchart* sistem berhasil mengeksekusi logika cerdas di tingkat perangkat.

Dalam memvalidasi keunggulan TinyML, kita menggunakan tiga formula utama:yaitu Formula (3,4, 5). Sistem ini gagal mendeteksi anomali karena ambang batasnya terlalu tinggi 35^0 C.

a. Sistem *Heuristic* (Non-TinyML)
   1. Precision (Presisi) : Seberapa akurat model saat menebak anomali.

$$Precision = \frac{0}{0+0} = 0$$

   2. Recall (Kepekaan): Seberapa banyak anomali asli yang berhasil ditangkap

$$Recall = \frac{0}{0+1} = 0$$

b. Sistem TinyML
   1. Precision (Presisi) : Seberapa akurat model saat menebak anomali.

$$Precision = \frac{1}{1+0} = 1{,}0\ (100\%)$$

   2. Recall (Kepekaan): Seberapa banyak anomali asli yang berhasil ditangkap

$$Recall = \frac{1}{1+0} = 1{,}0\ (100\%)$$

3. Perhitungan F1-Score (Final) persamaan (2)

Sistem *Heuristic* (Non TinyML)

$$F1 - Score = 2\ x\ \frac{0\ x\ 0}{0\ +0} = 0$$

Sistem TinyML

$$F1 - Score = 2\ x\ \frac{1\ x\ 1}{1\ +1} = \frac{2}{2} = 1{,}0\ (100\%)$$

Untuk lebih jelasnya validasi pada system *Heuristic* vs *Autoencoder* (non TinyML dengan TinyML) dapat dilhat pada tabel dibawah ini

Tabel 6. Validasi pada system *Heuristic* VS *Autoencoder* (non TinyML dengan TinyML)

Metrik Evaluasi	**Sistem Non TinyML**	**Sistem TinyML**	**Kesimpulan**
Akurasi Umum	Rendah (Kaku)	Sangat Tinggi (Adaptif)	TinyML lebih cerdas.
Precision	0%	100%	TinyML tidak memberikan alarm palsu.
Recall	0%	100%	TinyML tidak melewatkan bahaya.
F1-Score	**0.00**	1.00	TinyML Sempurna**.**

Hasil perhitungan F1-Score menunjukkan perbedaan performa yang sangat kontras, di mana sistem TinyML mencapai nilai sempurna (1.0) sementara sistem Heuristik berada pada nilai terendah (0.0). Hal ini terjadi karena sistem heuristik hanya mampu mendeteksi anomali yang bersifat ekstrem secara numerik (melewati 35^0C, sehingga ia kehilangan kemampuan 'intelegensi' untuk mengenali suhu 27.7^0 C sebagai ancaman. Sebaliknya, TinyML melalui proses inferensi lokal mampu memvalidasi bahwa suhu tersebut menyimpang dari profil normal, menghasilkan nilai *Recall* yang tinggi tanpa mengorbankan *Precision*. Perbedaan skor ini membuktikan secara matematis bahwa kecerdasan di tingkat *edge* mutlak diperlukan untuk sistem monitoring yang kritis

3.5 Analisis Konprehensif

Secara keseluruhan, rangkaian pengujian yang telah dilakukan membuktikan bahwa integrasi TinyML pada arsitektur *Edge Computing* memberikan peningkatan performansi yang signifikan dibandingkan dengan sistem monitoring konvensional (Non-TinyML). Dari aspek inteligensi, sistem berhasil mencapai *Accuracy Score* sempurna *(1,0)* dan F1-Score (1,0), di mana model mampu mengenali anomali melalui lonjakan *Reconstruction Error* (MSE) sebesar 8,41 pada deviasi suhu yang halus, sebuah kondisi yang gagal divalidasi oleh sistem heuristik. Secara operasional, meskipun terdapat alokasi memori dinamis sebesar 23 KB untuk menjalankan model, sistem justru menunjukkan stabilitas deterministik pada ketersediaan RAM dan efisiensi termal yang sangat baik dengan suhu kerja konstan di angka 24,73°C. Keunggulan ini diperkuat oleh optimalisasi jalur data yang mampu mereduksi latensi transmisi hingga 24% lebih cepat (±16 detik) dibandingkan sistem pembanding. Dengan demikian, dapat disimpulkan bahwa implementasi TinyML pada ESP32 bukan sekadar peningkatan fitur, melainkan sebuah transformasi sistem yang menghasilkan perangkat monitoring yang jauh lebih cerdas, responsif, dan reliabel untuk deteksi dini anomali di lingkungan penelitian. Terakhir, tabel berikut menampilkan Ringkasan Parameter Keunggulan TinyML

Tabel 7. Ringkasan Parameter Keunggulan Model *Autoencoder* (TinyML)

Dimensi Evaluasi	Indikator Keberhasilan	Status Validasi
Validasi Inteligensi	Akurasi 100% & F1-Score 1.0	Sangat Baik
Efisiensi Memori	Konsumsi ±23 KB (Tanpa *Memory Leak*)	Stabil
Responsivitas	Latensi 16 detik (Lebih cepat 5 detik)	Optimal
Integritas Fisik	Suhu Kerja 24,73°C (*Low Self-Heating*)	Aman

4. Kesimpulan

Penelitian ini menawarkan strategi konkret bagi industri untuk mengadopsi pemeliharaan prediktif yang hemat biaya. Kemampuan menjalankan model *Autoencoder* pada perangkat mikrokontroler terjangkau seperti ESP32 membuka peluang *retrofitting* mesin-mesin tua (*legacy equipment*) menjadi aset cerdas tanpa ketergantungan pada infrastruktur *cloud* yang mahal. Pendekatan ini memungkinkan desentralisasi pengawasan kondisi mesin, pengurangan latensi respons secara drastis, serta peningkatan privasi data, sehingga menjadi model solusi yang skalabel untuk diterapkan mulai dari sektor industri kecil menengah (IKM) hingga manufaktur skala besar

Meskipun menawarkan efisiensi yang signifikan, penelitian ini memiliki sejumlah keterbatasan yang perlu diantisipasi. Penggunaan sensor DHT11 membatasi resolusi pengukuran dan respons waktu, sehingga hasil eksperimen ini lebih merepresentasikan skenario *low-cost monitoring* dibandingkan presisi tingkat industri. Selain itu, pengujian yang dilakukan dalam lingkungan terkontrol mungkin belum sepenuhnya menangkap kompleksitas gangguan fisik di lapangan, serta adanya potensi bias data temporal di mana model yang dilatih pada durasi terbatas mungkin memerlukan pelatihan ulang (*retraining*) untuk beradaptasi dengan variasi kondisi lingkungan jangka panjang. Berdasarkan hasil analisis dan pembahasan yang telah dilakukan, dapat ditarik beberapa kesimpulan utama:

1. Validasi Inteligensi: Implementasi TinyML (Tiny Machine Learning) berhasil meningkatkan akurasi deteksi anomali secara signifikan. Melalui penghitungan *Mean Squared Error* (MSE) sebesar 8,41, sistem mampu mengenali deviasi suhu halus ($27,7^0$C)sebagai anomali, sedangkan sistem konvensional gagal memvalidasi kondisi tersebut.
2. Performa Jaringan: Arsitektur *Edge Computing* terbukti memangkas jalur komunikasi data, menghasilkan rata-rata latensi pengiriman ke platform ThingSpeak yang lebih rendah (16 detik) dibandingkan sistem Non-TinyML (21 detik).
3. Efisiensi Sumber Daya: Penggunaan model *Machine Learning* lokal tidak menyebabkan degradasi fisik pada perangkat. Hal ini dibuktikan dengan suhu operasional yang stabil pada 24,73°C (*minimum self-heating*) dan manajemen RAM yang konsisten tanpa adanya indikasi kegagalan sistem.
4. Keunggulan Strategis: Secara keseluruhan, sistem TinyML memberikan keseimbangan optimal antara kecerdasan buatan, kecepatan respons, dan ketahanan perangkat keras.

Untuk pengembangan penelitian selanjutnya, disarankan beberapa hal sebagai berikut:

1. Pengembangan Dataset: Melatih model dengan variasi data anomali yang lebih kompleks (seperti kelembapan atau kualitas udara) agar sistem memiliki kecerdasan *multivariate* dalam mendeteksi perubahan lingkungan.
2. Optimasi Konsumsi Daya: Mengintegrasikan fitur *Deep Sleep* pada ESP32 di sela-sela waktu inferensi TinyML untuk meninjau lebih lanjut efisiensi konsumsi daya pada penggunaan baterai jangka panjang.

3. Pengujian Skala Luas: Melakukan pengujian pada lingkungan luar ruangan (*outdoor*) dengan fluktuasi cuaca yang lebih ekstrem untuk menguji ketahanan model terhadap gangguan derau (*noise*) sensor yang lebih tinggi.
4. Hibridasi Model: Mencoba arsitektur model TinyML yang lebih ringan (seperti *Quantized Neural Network*) untuk melihat kemungkinan reduksi penggunaan RAM di bawah 23 KB tanpa menurunkan nilai F1-Score

Daftar Pustaka

Alajlan, N. N., & Ibrahim, D. M. (2022). TinyML : Enabling of Inference Deep Learning Models on Ultra-Low-Power IoT Edge Devices for AI Applications. *Micromachines*, *13*(6), 1–22. https://doi.org/https://doi.org/10.3390/mi13060851

Barambones, O., & Apiñaniz, E. (2022). Scalable IoT Architecture for Monitoring IEQ Conditions in Public and Private Buildings. *Energies*, *15*(6), 1–23. https://doi.org/https://doi.org/10.3390/en15062270

Behnke, I., & Austad, H. (2024). Real-Time Performance of Industrial IoT Communication Technologies : A Review. *IEEE Internet of Things Journal*, *11*(5), 1–13.

Bouman, R., & Heskes, T. (2024). *Auto encodersfor Anomaly Detection Are Unrealiable*. *1988*, 1–14. https://doi.org/https://doi.org/10.48550/

Chikezie, C. ifeanyi, Okpara, T. cyprian, & Mmadumbu;, A. chidubem. (2025). Utoencoders For Anomaly Detection: A Comprehensive Architectural Review, Comparative Insights, And Practical Guidance. *International Journal of Engineering*, *8*(5), 88–119. https://doi.org/https://doi.org/10.70382/tijert.v08i5.010

Choi, J., Park, J., & Japesh, A. (2023). A Subspace Projection Approach to *Autoencoder*-based Anomaly Detection. *IEEE for Possible Publication*. https://doi.org/https://doi.org/10.48550/arXiv.2302.07643

Danladi, M. S., & Baykara, M. (2022). Design and Implementation of Temperature and Humidity Monitoring System Using LPWAN Technology. *International Information and Engineering Technology Association*, *27*(4), 521–529. https://doi.org/https://doi.org/10.18280/isi.270401 Received:

Dhyani, S., & Butola, R. (2025). *Autoencoders for ECG Anomaly Detection : A Survey*. *02*(01), 47–58.

Ficili, I., Giacobbe, M., & Tricomi, G. (2025). *From Sensors to Data Intelligence : Leveraging IoT , Cloud , and Edge Computing with AI*. 1–25.

Immonen, R. (2022). *Review Article Tiny Machine Learning for Resource- Constrained Microcontrollers*. *2022*.

Laroui, M., Nour, B., Moungla, H., Cherif, M. A., & Afifi, H. (2021). Edge and fo computing for IoT : A survey on current research activities & future directions. *IEEE Access (2023)*, *180*(July 2020), 210–231. https://doi.org/https://doi.org/10.1016/j.comcom.2021.09.003

Magadán F J, Suárez, L., & García, J. C. G. D. F. (2023). Low - Cost Industrial IoT System for Wireless Monitoring of Electric Motors Condition. *Mobile Networks and Applications*, *28*(1), 97–106. https://doi.org/10.1007/s11036-022-02017-2

Merrill, N. (2020). *Modified Autoencoder Training and Scoring for Robust Unsupervised Anomaly Detection in Deep Learning*. *8*.

Soro, S. (2020). TinyML for Ubiquitous Edge AI. In *Mitre Tecnical Report* (Issue 20).

Sudharsan, B., Salerno, S., Nguyen, D., Yahya, M., & Wahid, A. (2021). *TinyML benchmark : Executing fully connected neural networks on commodity microcontrollers TinyML Benchmark : Executing Fully Connected Neural Networks on Commodity Microcontrollers*. 10–12. https://doi.org/10.13025/rmkq-1966