www.ijarp.com ISSN 2456-9992 Page: 01-27

Research Article

International Journal Advanced Research Publications

Volume: 02
Issue: 10

DEVELOPMENT OF A DIGITAL TWIN FRAMEWORK FOR REAL-TIME MONITORING AND PERFORMANCE OPTIMIZATION OF ELECTRICAL MACHINES

***Chinenye Eberechi Uchegbu**

Department of Electrical and Electronic Engineering, Abia State University Uturu, Nigeria

Article Received: 17 September 2026, Article Revised: 07 October 2026, Published on: 27 October 2026

***Corresponding Author: Chinenye Eberechi Uchegbu**

Department of Electrical and Electronic Engineering, Abia State University Uturu, Nigeria

Doi: https://doi-doi.org/101555/ijarp.5085

ABSTRACT

Electrical machines remain among the most failure-prone assets in industrial settings, and the cost of unplanned downtime keeps pushing engineers toward earlier, more granular fault detection. Conventional condition monitoring tends to be reactive: faults are confirmed only after symptoms surface, by which point efficiency has often already degraded. Digital twin technology offers a way to close that gap, yet much of the published work treats fault diagnosis and performance optimization as separate problems rather than as stages of one continuous loop. This study addresses that separation directly. It pursues two objectives: first, to design a four-layer digital twin framework architecture for real-time condition monitoring of electrical machines, and second, to evaluate, through software-in-the-loop simulation, the intelligent control mechanisms that allow the framework to act on what it detects. A Python-based synthetic data generator produced time-series signals for stator current, vibration, temperature, and voltage under one healthy and four faulty conditions; the stream was ingested into InfluxDB and queried by a fault-classification model linked to a rule-based control layer. Under simulated load the pipeline sustained an ingestion throughput of 10,000 records per second while classifying every window (95th-percentile detection latency 145 ms) and saturated near 35,700 records per second; in sequential operation the median end-to-end detection latency was 99 ms. Five-class classification accuracy was 97.0% under five-fold cross-validation, with bearing wear and mechanical misalignment separated without error and the remaining errors confined to the boundary between healthy operation and winding overheating. In a 120-second closed-loop scenario, the load-adjustment response held simulated winding temperature to a peak of 87.3 °C, where the same fault without a

response reached 100.5 °C. These figures describe the software architecture's behavior under load, not the physical response of a deployed motor. The framework demonstrates that monitoring and optimization can be unified within a single orchestrated pipeline, and it points toward hardware validation as the logical next step.

KEYWORDS: Condition Monitoring, Digital Twin, Electrical Machines, Fault Diagnosis, Industry 4.0, Real-Time Control, Time-Series Database.

1. INTRODUCTION

Industrial machinery does not announce its failures in advance. It hums along until it doesn't, and the gap between those two states is where most maintenance budgets get spent. The fourth industrial revolution has reframed how that gap is managed. Where factories once relied on scheduled inspections and operator intuition, Industry 4.0 pushes toward continuous, data-driven oversight of physical assets (Aheleroff et al., 2021; Ferrigno et al., 2023). Small and medium manufacturers in developing economies have been slower to adopt this shift, constrained by cost and infrastructure, but the direction of travel is the same everywhere: sensors, connectivity, and analytics replacing guesswork (Atieh et al., 2022).

Electrical machines sit at the center of this concern. Motors, generators, and the drives that control them are mechanically simple in principle but operationally unforgiving; a bearing that wears unevenly or a winding that overheats can take a production line down for hours. Adedeji et al. (2025) describe how predictive maintenance has progressively displaced reactive repair in smart factory settings, and the literature on fault diagnosis in rotating machinery has grown accordingly. Jakaria et al. (2025) built a deep learning framework trained on a million simulated samples of induction motor behavior, distinguishing healthy operation from short circuits, overloads, and mechanical disturbances with classification accuracies above 90%. Their work, like much of the field, treats detection as the endpoint. Fewer studies ask what happens after a fault is detected, and how that detection might feed back into the machine's operating parameters in real time.

This is where digital twin technology becomes relevant, and where its promise has outpaced, in places, its demonstrated delivery. A digital twin is not simply a simulation; it is a continuously updated virtual counterpart to a physical asset, synchronized through live data and capable of informing decisions back at the physical layer (Peksa & Mamchur, 2026). Kia et al. (2025) trace how real-time digital twins for electrical machines have matured from offline multiphysics models used during design, toward live, data-driven replicas that support

intelligent fault diagnosis as the machine actually runs. Hu et al. (2025) reach a similar conclusion from a different angle, surveying digital twin applications in motor fault diagnosis and noting that the technology's defining strength, full lifecycle coverage paired with two-way feedback between virtual and physical states, is exactly what distinguishes it from a passive dashboard.

And yet the gap persists. A recurring pattern across the literature is the separation of monitoring from optimization, as though detecting a fault and responding to it were unrelated tasks. Kurkin et al. (2026) demonstrate sophisticated vibration diagnostics for machine tools within a digital twin framework, but the loop largely ends at diagnosis. Constantino-Robles et al. (2026), reviewing predictive maintenance approaches for wind turbines through a PRISMA-guided synthesis, find the same emphasis: methods for predicting remaining useful life are well developed, while the mechanisms for acting on those predictions in real time receive comparatively little attention. Damtew (2024) reports similar findings for the automotive sector, where digital twins inform material performance assessments without necessarily closing the loop back into machine control. This pattern is not a flaw in any single study so much as a gap in how the field has organized its questions.

A second, related gap concerns validation. Several frameworks remain conceptual, presented as architecture diagrams without an accompanying empirical demonstration of how the pieces behave under load. This is a fair criticism, and one this study takes seriously. A framework that cannot show its ingestion throughput, its detection latency, or its classification accuracy is, at best, a hypothesis about software behavior rather than evidence of it.

This study responds to both gaps. It pursues two specific objectives. The first is to design a digital twin framework architecture, organized into four functional layers, for real-time condition monitoring of electrical machines. The second is to develop and evaluate, through software-in-the-loop simulation, the intelligent control mechanisms by which that framework converts detection into action; load adjustment, alerting, or escalation, depending on what the data shows. Doshi et al. (2024) make the case that edge-based architectures are now mature enough to support this kind of low-latency processing even on constrained hardware, which gives the second objective a reasonable technical foundation to build on.

2. Literature Review

2.1 Industry 4.0 Foundations and Digital Twin Evolution

The case for digital infrastructure in manufacturing did not begin with digital twins specifically. It began with a broader push toward connected, data-rich production

environments. Aheleroff et al. (2021) frame this as "digital twin as a service," arguing that the technology gains most of its value not from any single deployment but from being offered as a flexible, reusable capability across a manufacturing network; their resilience case study found that this service-oriented framing held up well under operational disruption. Ferrigno et al. (2023), working from a bibliometric analysis of Industry 4.0's foundational technologies, observe that research output in this area has grown unevenly, clustering around a handful of business model archetypes rather than spreading evenly across possible applications. That unevenness matters for the present study, because it suggests digital twins for electrical machines specifically remain underexplored relative to digital twins for manufacturing processes in general.

Adoption is not universal, and the literature is candid about why. Atieh et al. (2022) studied small and medium enterprises in developing countries and found that intelligent manufacturing systems, digital twins among them, face real constraints around capital, skilled labor, and existing infrastructure. Ghobakhloo et al. (2023) extend this into supply chain resilience, proposing a roadmap that treats digital transformation as something sequenced deliberately rather than adopted all at once. Erboz and Yumurtaci Hüseyinoglu (2023) add a cost dimension: Industry 4.0 technologies improve supply chain flexibility but do not automatically reduce cost, since that relationship depends on implementation. Dutta et al. (2021) make a related point about quality control, arguing that SMEs need a conceptual roadmap for digitalizing inspection before adopting anything more ambitious. Taken together, these studies establish that the infrastructure surrounding a digital twin, organizational readiness, cost, sequencing, matters as much as the technology itself.

2.2 Digital Twin Architectures for Machine Monitoring

Within this broader context, a more specific literature has developed around digital twins built explicitly for machine-level monitoring. Kia et al. (2025) provide what is probably the most directly relevant account for the present study: a review of real-time digital simulation in the service of intelligent fault diagnosis and condition-based monitoring for electrical machines specifically. Their account traces a progression from physics-based models, through circuit-oriented lumped-parameter approaches, to fully data-driven digital twins enabled by cloud computing and machine learning; the more recent the approach, the more it depends on continuous data rather than periodic recalibration. Hu et al. (2025), reviewing the same general territory but from a fault-diagnosis-first angle, single out full lifecycle coverage as the property that distinguishes a digital twin from earlier simulation tools: the twin is

updated continuously by the physical entity, and its outputs feed back into that entity's operation, rather than running once and being set aside.

Other work extends this architecture to adjacent domains in ways that still inform the present design. Peksa and Mamchur (2026) propose a federated digital twin lifecycle model, arguing that a layered structure, separating data acquisition from analytics from control, is what allows a framework to be reused rather than rebuilt for each new asset. Ahmmed et al. (2025), reviewing over a hundred studies on digital twins in additive manufacturing, raise a point this study's architecture has to take seriously: a digital twin's value depends on the integrity of the data flowing through it, and a pipeline that ignores tampering or corruption in transit is incomplete regardless of classifier performance. Kurkin et al. (2026) apply a comparable layered logic to vibration diagnostics for machine tools, combining multiple sensor parameters within one digital twin framework and reporting that the multiparametric approach catches faults single-sensor methods miss. Constantino-Robles et al. (2026), reviewing the wind turbine literature through a PRISMA-guided process, reach a related conclusion: the field has matured methodologically but remains fragmented across machine types, with little cross-pollination between, say, wind turbine diagnostics and motor diagnostics, despite similar underlying vibration physics. Damtew (2024), writing from the automotive upstream sector, shows digital twins informing material performance and lifecycle decisions in a context closer to design support than real-time control, a useful contrast case since it illustrates an application that does not close the monitoring-to-optimization loop this study is concerned with.

2.3 Machine Learning and AI-Based Fault Diagnosis

Detection accuracy has improved substantially as deep learning methods have displaced manual feature extraction. Jakaria et al. (2025) trained eight architectures, including CNN, LSTM, and several hybrid combinations, on a million simulated samples of squirrel-cage induction motor behavior under five fault conditions plus a healthy baseline; their CNN-GRU and CNN-LSTM hybrids reached classification accuracies above 92%, and the authors report inference latency in the range of 101 to 111 milliseconds per batch, figures that bear directly on what real-time deployment can be expected to achieve. Lwahas et al. (2025), reviewing recent advances in machine learning for fault diagnosis across mechatronic systems more broadly, note that hybrid spatial-temporal architectures consistently outperform single-architecture baselines, which is broadly consistent with what Jakaria et al. found for induction motors specifically.

Fault diagnosis research has also moved toward representing failure knowledge explicitly rather than relying solely on learned patterns. Fernandez del Amo et al. (2024) advance an ontology-based approach to fault diagnosis, arguing that structured knowledge capture, essentially encoding what a fault looks like and why, complements purely data-driven classification by making the system's reasoning auditable. Gispert et al. (2025) apply a related ontology-based logic to predictive maintenance in building facilities, building an asset information model that links sensor data to maintenance actions through an explicit rule structure rather than a black-box classifier alone. Neither of these papers concerns electrical machines specifically, but both inform the present study's decision to pair a learned classification model with an explicit, rule-triggered control layer rather than relying on the classifier's output alone to drive action.

Kang et al. (2025) offer a useful data point from outside the electrical-machine literature that nonetheless bears on the present framework's design. Their work augments a digital twin with artificial intelligence to improve machinability prediction in robotic composite machining, and the result they report, improved process accuracy when AI augmentation and digital twin feedback operate together rather than separately, supports the central premise of this study: that monitoring and optimization function better as a coupled system than as sequential, disconnected stages.

2.4 Real-Time Control, Voltage Stability, and Optimization

The control side of the framework draws on a literature that is, in places, only loosely connected to digital twins as such, but directly relevant to what real-time intelligent control of electrical systems requires. Hao et al. (2024) analyze short-term voltage stability in power systems with high photovoltaic penetration, an increasingly common scenario as renewable generation grows, and propose enhancement strategies that respond to instability within seconds rather than minutes; their work illustrates the kind of low-latency control logic the present framework's optimization layer is meant to approximate at smaller scale. Essien et al. (2024), studying the Nigerian 330 kV transmission system specifically, apply particle swarm optimization to improve voltage stability and report measurable gains over baseline control, which is one of relatively few studies grounding this kind of optimization work in an African grid context. Yabani et al. (2026), reviewing machine learning approaches to voltage stability across Nigeria's power network more broadly, find that ML-based methods are gradually displacing purely analytical approaches, though the authors caution that data availability remains a constraint on how far this can go in practice.

Hassan et al. (2023) approach optimization from the renewable generation side, using an improved genetic algorithm to track the maximum power point in solar photovoltaic systems under partial shading, a problem that shares structural features with the fault-response optimization this study's framework performs: both require a control algorithm that can act on imperfect, changing input within a short time window. Hafner et al. (2025), working at a considerably more general level, demonstrate that world models, internal predictive representations an AI system builds of its environment, can master diverse control tasks without task-specific engineering; their results suggest that the kind of rule-triggered control logic used in the present study's optimization layer is a conservative, interpretable starting point rather than the most sophisticated option available, and that more adaptive control could plausibly be layered on top in future work. Langas et al. (2025) and Ho et al. (2025) both report digital-twin-coupled control systems, in human-robot teaming and 5G-enabled robotic automation respectively, that close the loop between sensing and actuation in something close to real time; their results give grounds for treating the closed-loop pattern as achievable with current technology rather than aspirational.

2.5 IoT, Sensor Integration, and Network-Enabled Digital Twins

The final strand of relevant literature concerns the infrastructure that carries sensor data from machine to model. Doshi et al. (2024) review edge AI architectures for low-power IoT devices, concluding that meaningful inference can now run close to the sensor rather than requiring a round trip to centralized cloud computing; this matters for latency budgets in any real-time monitoring application, including the one proposed here. Li et al. (2025) survey generative AI applications within network digital twins, finding that the architectural pattern, separating data ingestion, virtual representation, and decision logic into distinct layers, recurs across very different application domains, from telecommunications to the industrial settings more central to this study. Cao et al. (2025), working on UAV swarm coordination, use digital-twin-enabled sim-to-real transfer to bridge simulated training and physical deployment; although their application is aerial robotics rather than electrical machines, their finding that simulation-trained models transfer reasonably well to physical systems is encouraging for the present study's reliance on synthetic data as a first validation step. Erunkulu et al. (2023), studying mobile communication channel propagation, and Jafari et al. (2023), reviewing digital twin applications across smart grid, transportation, and smart city domains, both reinforce a more general point: the data pipeline connecting sensors to analytics is not a solved, generic problem, but one that requires deliberate engineering for

each domain's specific signal characteristics, whether that domain is electrical machine monitoring or wireless communication.

Mansour et al. (2023) bring several of these threads together in a survey focused specifically on IoT and digital twin applications across electrical power systems. Their account of condition monitoring and diagnosis for power transformers, electrical grids, and substations is, of the sources reviewed here, the closest in spirit to the present study's subject matter, even though it operates at grid scale rather than at the level of an individual machine. Their finding that digital twin value in electrical systems concentrates particularly in condition monitoring applications, more than in, say, energy management or planning, supports this study's choice to center the framework's contribution on monitoring and optimization rather than on broader asset management functions.

2.6 Gap Synthesis

Reading across these five areas, a consistent pattern emerges. The architectural foundations for digital twins are well established (Aheleroff et al., 2021; Peksa & Mamchur, 2026); fault detection accuracy, particularly using deep learning, has reached a mature stage for several machine types (Jakaria et al., 2025; Lwahas et al., 2025); and the control-theoretic tools needed for real-time optimization exist and have been demonstrated in adjacent domains (Hao et al., 2024; Hafner et al., 2025). What is comparatively scarce is work that combines these three strands into one orchestrated pipeline and then demonstrates, empirically rather than only conceptually, that the pipeline performs adequately under realistic load. Kurkin et al. (2026) and Hu et al. (2025) come closest, but even they emphasize diagnosis more than the optimization step that follows it. This study's two objectives, designing the four-layer architecture and then validating it through software-in-the-loop simulation with explicit throughput, latency, and accuracy metrics, are aimed squarely at that remaining gap.

3. METHODOLOGY AND ARCHITECTURE

This study follows a design-science approach. The principal output is an artifact, specifically a layered digital twin framework, and the artifact's adequacy is established through simulated software-in-the-loop evaluation, specified in Sections 3.2 to 3.5 and reported in Section 4, rather than through deployment on a physical machine. This is appropriate given the research questions at hand. The questions concern how monitoring and control components should be organized and how they perform computationally under load; they do not concern the electromagnetic behavior of any specific motor or generator, which is a separate question this study does not attempt to answer.

3.1 Architectural Overview

The proposed framework comprises four functional layers, a structure broadly consistent with established digital twin reference models for industrial and electrical systems (Kia et al., 2025; Peksa & Mamchur, 2026). At the base sits the physical layer: the electrical machine itself, instrumented with sensors for stator current, vibration, winding temperature, and terminal voltage. Above it, the data acquisition and edge layer captures these signals at high sampling frequency and performs lightweight preprocessing, filtering noise and scaling values, before forwarding structured records onward (Doshi et al., 2024). The analytics and virtual layer follows. This is where the digital replica actually lives: a time-series database ingests the incoming stream, and a classification model continuously evaluates incoming records against learned healthy and fault signatures (Li et al., 2025; Jakaria et al., 2025). At the top, the control and optimization layer receives the analytics layer's classification output and decides what to do about it, issuing an alert, adjusting load, or escalating, depending on severity (Hafner et al., 2025; Langas et al., 2025).

Each layer maps onto one or both of the study's two objectives. The physical, acquisition, and analytics layers together satisfy the first objective, real-time condition monitoring. The control and optimization layer, acting on what the analytics layer reports, satisfies the second.

Table 1 summarizes the framework's layer groups, their components and functions, and the supporting technologies associated with each.

Layer Group	Components and Function	Supporting Technologies
Monitoring layers (Physical; Data Acquisition and Edge; Analytics and Virtual)	Sensors capture stator current, vibration, temperature, and voltage; edge preprocessing filters and structures the stream; the analytics layer stores and classifies incoming records against healthy and fault signatures	Current transducers, accelerometers, thermocouples, edge AI modules (Doshi et al., 2024), InfluxDB, supervised classification model (Kia et al., 2025; Li et al., 2025)
Control and Optimization layer	Receives classification output and converts it into an alert, a load adjustment, or an escalation flag, closing the loop between detection and action	Rule-triggered decision logic, actuation interface to machine controls (Hafner et al., 2025; Langas et al., 2025; Ho et al., 2025)

Table 1. Digital Twin Framework Layers, Components, and Supporting Technologies

Figure 1 renders this architecture visually, showing the upward flow of monitoring data and the downward flow of optimization feedback that closes the loop between detection and action.

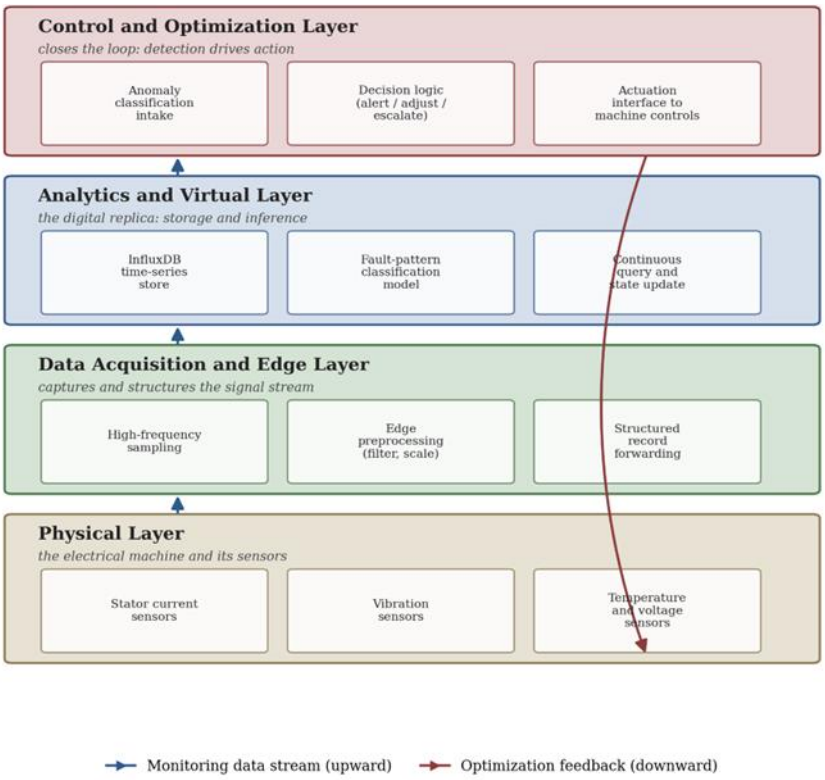


Figure 1. Conceptual Architecture of the Digital Twin Framework.

One design choice deserves explicit comment. The framework places the time-series database at the center of the analytics layer rather than treating storage as an afterthought to the classification model. This reflects a practical concern: a classifier that cannot query recent historical context, not just the current reading but the trend leading up to it, is working with less information than the physical system actually generates. The following subsections operationalize this architecture through a simulated testbed, and Section 4 reports its ingestion, detection, and control performance under synthetic load.

3.2 Simulated Data Pipeline

A Python-based synthetic data generator was built to emulate the physical and data acquisition layers described in Section 3.1. Each record is one timestamped sample carrying eight channels: three-phase stator current, vibration acceleration, winding temperature, and three-phase terminal voltage, which together form the four signal groups the framework monitors. Records are generated at 5 kHz and grouped into analysis windows of 2,500 records (0.5 s), parameterized to represent one healthy baseline state and four fault conditions, following simulation conventions broadly consistent with those used in prior induction-machine fault studies (Jakaria et al., 2025). Operating conditions vary from window to window so that the classification problem is not trivially clean: load ranges from

30% to 100% of rating, ambient temperature from 20 °C to 35 °C, sensor gain within ±15%, and the vibration noise level over a range of about 1.6 to 1. The resulting stream is written into InfluxDB (open-source version 1.6), a time-series database chosen specifically for its write-optimized storage model and its established use in comparable real-time industrial monitoring pipelines (Kia et al., 2025; Doshi et al., 2024). InfluxDB functions here as the empirical backbone of the analytics layer, supporting continuous ingestion at the same time as it is being queried by the classification model running alongside it. Figure 2 traces the data flow from sensor simulation through ingestion and classification to the control response, and marks where each performance measure was taken.

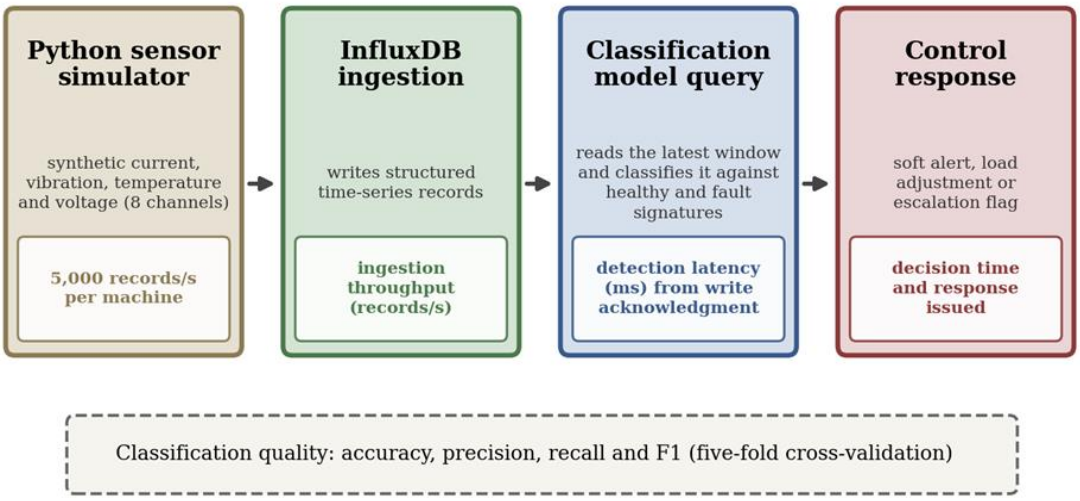


Figure 2. Simulated Software-in-the-Loop Data Flow and Measurement Points.

3.3 Fault Scenarios

Four fault conditions were defined for the simulation, chosen for their established relevance in the electrical machine condition monitoring literature reviewed in Section 2 (Lwahas et al., 2025; Fernandez del Amo et al., 2024).

Bearing wear was modeled as a train of decaying impulses at the outer-race defect frequency (3.58 times the shaft frequency, about 87 Hz) exciting a 1.5 kHz structural resonance, which raises high-frequency vibration amplitude and layers harmonic components onto the baseline signal. Stator winding overheating was modeled as a sustained upward offset in winding temperature of up to 55 °C above the load-determined baseline, accompanied by an upward drift of 0.15 to 0.60 °C per second, so that the winding passes the 95 °C rated operating threshold at high severity; this distinguishes it from a transient spike. Voltage imbalance was modeled by reducing one phase voltage by up to 10% and shifting its phase angle by up to

3.5 degrees, with the current imbalance amplified six-fold relative to the voltage imbalance, a pattern consistent with the phase-to-phase and phase-to-ground disturbances Jakaria et al. (2025) simulated in their induction motor work. Mechanical misalignment was modeled as periodic vibration amplitude modulation, up to 35% in depth, correlated with simulated shaft rotation, together with a second-order shaft-frequency component and small sidebands in the stator current.

Fault severity was drawn uniformly between 0.10 and 1.00 for every faulted window, so a share of the faulted windows is deliberately mild. Each fault condition was generated alongside the healthy baseline in equal proportion, 700 windows per class and 3,500 in total, producing a labeled synthetic dataset suitable for supervised classification. No claim is made that this dataset captures the full noise and variability of a physical industrial environment; it is, deliberately, a controlled testbed for the software pipeline rather than a substitute for field data. Figure 3 shows one representative window from each class.

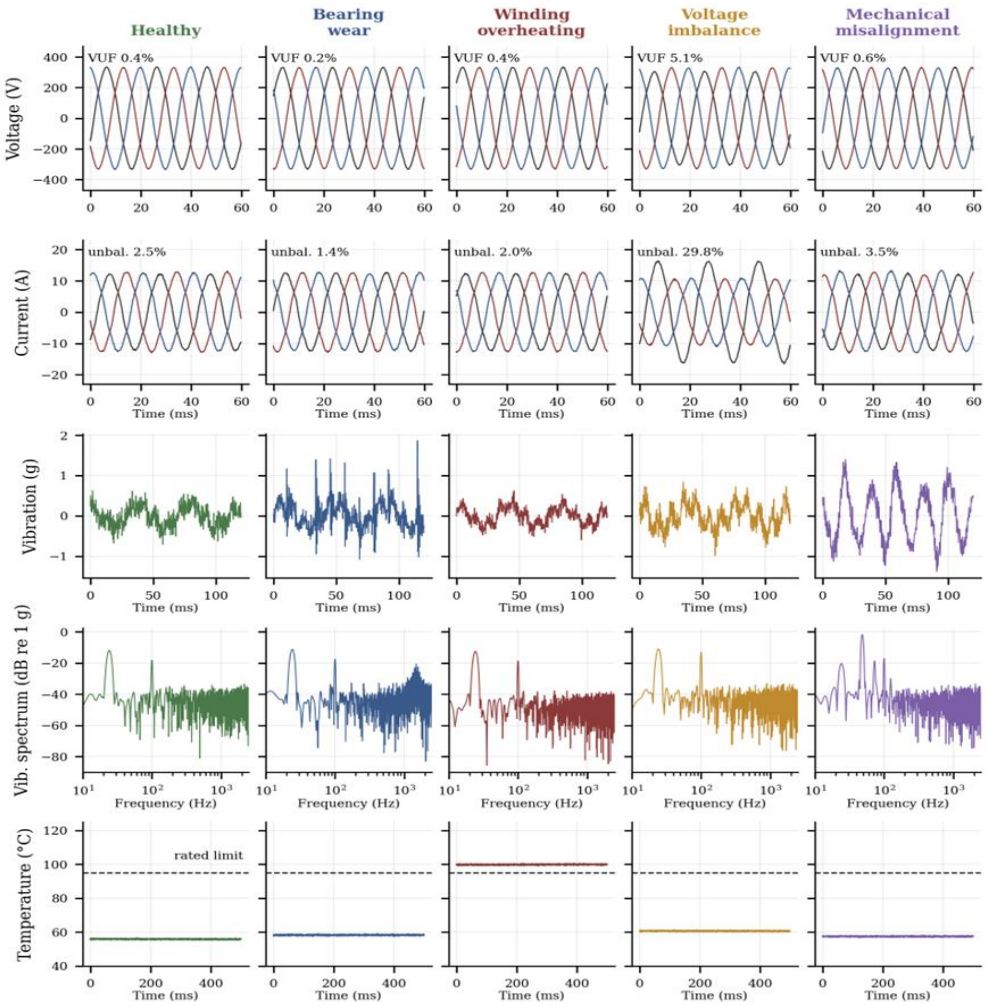


Figure 3. Representative Simulated Signals for the Healthy Baseline and Four Fault Conditions

Note. One window per class at 80% load, with fault severity 0.8 for the four fault classes. Rows show three-phase terminal voltage, three-phase stator current, vibration in the time domain, the vibration amplitude spectrum, and winding temperature (dashed line: 95 °C rated limit). VUF is the voltage unbalance factor, and unbal. is the corresponding current imbalance.

3.4 Optimization Logic

The control and optimization layer implements a rule-triggered feedback loop. For each analysis window the classifier returns a probability for each of the five classes, and the framework takes the most probable fault class as its candidate diagnosis. When that probability reaches a confidence threshold τ, the framework issues one of two class-specific responses. Thermal or electrical stress conditions, overheating or voltage imbalance in particular, trigger a load-adjustment command intended to reduce stress on the machine before the fault worsens. Bearing or mechanical fault classifications trigger an escalation flag, signaling that intervention is likely required. When no fault class reaches τ but the probability of healthy operation is below 0.5, the anomaly is treated as low-severity or ambiguous and generates a soft alert intended for an operator's attention rather than immediate action; otherwise no action is taken. The threshold was set to $\tau = 0.75$, the lowest value on a grid with a step of 0.05 for which fewer than 1% of healthy windows triggered a class-specific response, using out-of-fold classifier probabilities; Section 4.3 reports the full threshold sweep. This logic is intentionally simple. Hafner et al. (2025) demonstrate considerably more adaptive control through learned world models, and a more sophisticated version of this framework's control layer could plausibly draw on that work; for the present study, an explicit, auditable rule structure was chosen to keep the relationship between detection and response transparent (Ho et al., 2025; Langas et al., 2025).

3.5 Evaluation Protocol and Metrics

A note on interpretation is necessary before the figures themselves. The metrics reported in Section 4 characterize the computational performance of the digital twin's software architecture, the Python data generator, the InfluxDB backend, and the classification model, operating under simulated load. They do not describe the physical response of a deployed hardware motor, and they should not be read as such. The purpose of this evaluation is to establish whether the proposed pipeline can ingest, query, and classify data fast enough to support real-time use; whether a physical motor would actually exhibit the fault signatures simulated here is a separate question, left for the hardware validation recommended in Section 6.

Classification used a random forest of 100 trees applied to 18 features extracted from each window: current root-mean-square value and phase imbalance, voltage root-mean-square value and imbalance, supply-to-shaft sidebands in the stator current, vibration root-mean-square value, kurtosis and crest factor, vibration spectral amplitudes at the first and second shaft harmonics and

at 100 Hz, the high-frequency band (0.8 to 1.8 kHz) root-mean-square value and kurtosis, the envelope-spectrum amplitude at the bearing defect frequency, and the winding temperature level and slope. Performance was evaluated by stratified five-fold cross-validation. The confusion matrix pools the out-of-fold predictions, and precision, recall, and F1 score are reported for each class as the mean and standard deviation over the five folds.

Detection latency was measured over 600 consecutive windows, after 20 warm-up windows, as the time from InfluxDB's acknowledgment of the write of a window's records to the return of a control decision. It therefore covers the store query, feature extraction, classifier inference, and rule evaluation, and it excludes the 0.5 s needed to accumulate a window. Throughput was measured in two ways: by offering load between 2,500 and 160,000 records per second for 10 s per level while a concurrent consumer queried and classified every window, and by a 10 s write-only run using 5,000-record batches. Achieved throughput counts records acknowledged by InfluxDB per second and includes line-protocol formatting on the producer side. All components ran on a single node with one CPU core and 3 GB of memory, shared between the producer, the consumer, and the InfluxDB server.

The closed-loop scenario ran for 120 s (240 windows) through the same pipeline. A winding-overheating fault began at 20 s with its severity rising from 0.10 to 0.80 over 20 s, and it was cleared at 75 s, at which point the load setpoint was restored to 85%; a bearing-wear fault began at 85 s with severity rising from 0.10 to 0.70 over 20 s. Winding temperature followed a first-order thermal model with a 12 s time constant, driven by load and fault heating, and a load-adjustment command lowered the load setpoint from 85% to 65%. An open-loop reference, using the same thermal model and fault schedule with the load held at 85%, shows the temperature without a control response. The threshold sweep used the out-of-fold class probabilities, varied τ from 0.30 to 0.98 in steps of 0.01, and recorded the detection rate, the share of faulted windows that triggered a class-specific response, and the false-action rate, the share of healthy windows that did so.

4. RESULTS

This section reports the results of the evaluation protocol in Section 3.5 in three parts: classification performance (Table 2 and Figures 4 and 5), computational performance (Figures 6 and 7), and closed-loop control behavior (Figures 8 and 9).

4.1 Classification Performance

Five-class accuracy under five-fold stratified cross-validation was 97.0% (SD 0.3 percentage points; fold accuracies ranged from 96.6% to 97.4%). Figure 4 shows the pooled confusion matrix and Figure 5 the per-class precision, recall, and F1 score, which Table 2 lists alongside

the system response each class triggers. Performance was not uniform across classes. Bearing wear and mechanical misalignment were classified without error in all 700 windows each, and voltage imbalance was almost as clean (recall 99.6%, precision 1.000), with 3 of its 700 windows misclassified, 2 as healthy and 1 as winding overheating. The remaining errors were concentrated at the boundary between the healthy baseline and winding overheating: 66 of 700 overheating windows (9.4%) were classified as healthy, and 36 of 700 healthy windows (5.1%) as overheating, giving F1 scores of 0.927 and 0.925 for the two classes, respectively.

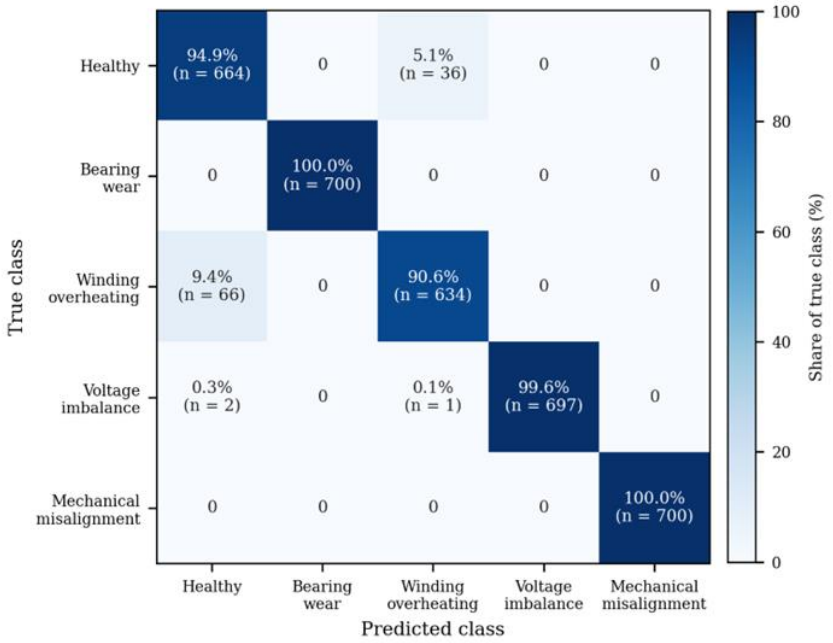


Figure 4. Normalized Confusion Matrix for Five-Class Fault Classification.

Note. Pooled out-of-fold predictions from five-fold stratified cross-validation, 700 windows per class; each row sums to 100%.

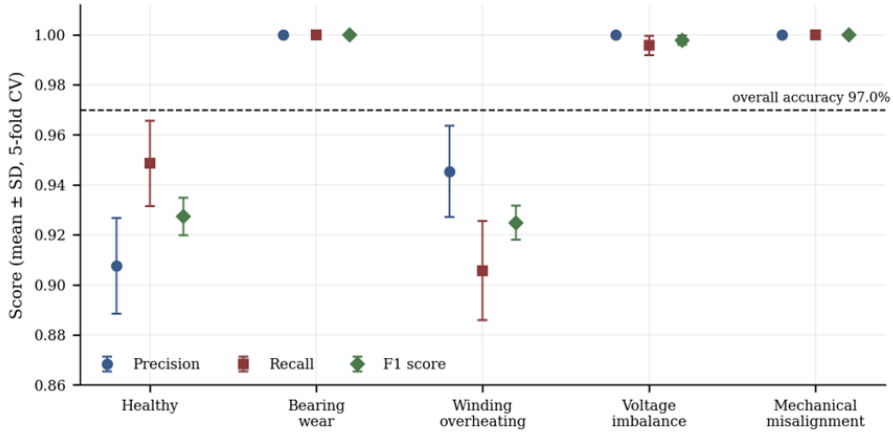


Figure 5. Per-Class Precision, Recall, and F1 Score.

Note. Markers show the mean over the five cross-validation folds and whiskers one standard deviation; the dashed line marks overall accuracy.

Table 2. Per-Class Classification Performance (Mean ± SD Over Five Folds) and System Response.

Fault Scenario	Simulated Signature	Precision	Recall	F1 Score	System Response
Healthy baseline	Baseline signals under varying load and ambient temperature	0.908 ± 0.019	0.949 ± 0.017	0.927 ± 0.008	None
Bearing wear	Outer-race impulse train exciting a 1.5 kHz resonance; raised high-frequency vibration	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	Escalation flag
Winding overheating	Sustained temperature offset of up to 55 °C with upward drift	0.945 ± 0.018	0.906 ± 0.020	0.925 ± 0.007	Load adjustment
Voltage imbalance	One phase voltage reduced by up to 10%; amplified current imbalance	1.000 ± 0.000	0.996 ± 0.004	0.998 ± 0.002	Load adjustment
Mechanical misalignment	2× shaft-frequency component and shaft-locked amplitude modulation	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	Escalation flag

The errors depended strongly on fault severity. Of the 66 overheating windows classified as healthy, 59 had a severity below 0.3; the misclassification rate for overheating windows was 39.1% in the 0.1 to 0.3 severity band (59 of 151 windows), 4.2% between 0.3 and 0.5 (6 of 143), 0.6% between 0.5 and 0.7 (1 of 170), and 0 of 236 above 0.7. The 36 healthy windows classified as overheating were drawn from higher operating loads (median 0.86, against 0.66 for all healthy windows) and warmer ambient conditions (31.4 °C, against 27.5 °C). Fold-to-fold variation in F1 was largest for these two classes (standard deviations of 0.8 and 0.7 percentage points) and zero for bearing wear and mechanical misalignment.

4.2 Computational Performance

Detection latency was recorded for 600 consecutive windows (Figure 6a). The median was 99 ms, the 95th percentile 196 ms, and the 99th percentile 295 ms, with a minimum of 41 ms and a maximum of 441 ms. The store query dominated the total (Figure 6b): its median was 85.7 ms (95th percentile 177 ms), against 4.2 ms for feature extraction (10 ms), 7.3 ms for classifier inference (16 ms), and 16 µs for the rule-based decision (24 µs). Acknowledging the write of a 2,500-record window took a median of 17.8 ms (95th percentile 39.9 ms) when

writes and queries alternated, and the write-only run with 5,000-record batches sustained 75,800 records per second.

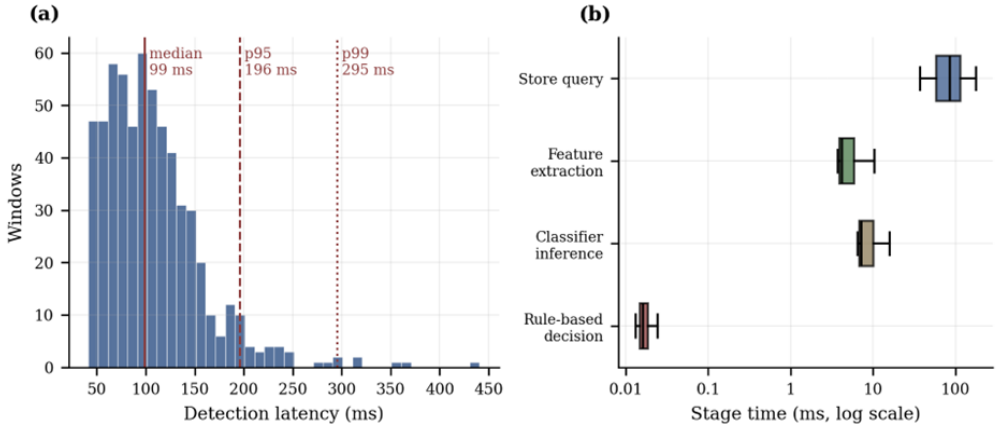


Figure 6. End-to-End Detection Latency and Stage Breakdown.

Note. (a) Distribution over 600 windows with the median, 95th, and 99th percentiles marked. (b) Time spent in each pipeline stage; boxes show the interquartile range, whiskers the 5th to 95th percentile, and lines the median.

Under increasing offered load with concurrent classification of every window (Figure 7), achieved throughput matched offered load up to 20,000 records per second and then plateaued between 32,200 and 35,700 records per second at offered loads of 40,000 to 160,000 records per second (Figure 7a). Detection latency stayed low up to 10,000 records per second (median 61 ms, 95th percentile 145 ms), rose at 20,000 records per second (median 251 ms, 95th percentile 1,217 ms), and reached medians of 6.0 to 7.9 s at 40,000 records per second and above, as windows queued behind the classifier (Figure 7b). A single machine sampled at 5 kHz corresponds to 5,000 records per second; the highest offered load with a 95th-percentile latency under 250 ms was 10,000 records per second, twice that rate.

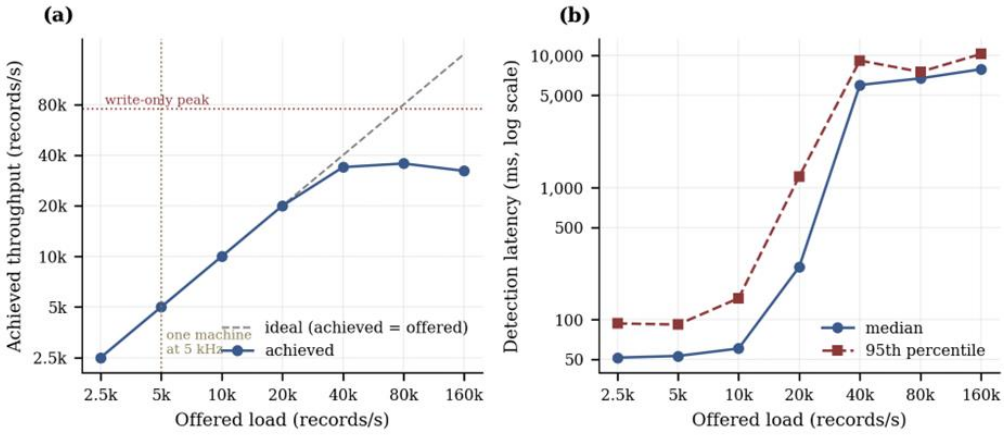


Figure 7. Ingestion Throughput and Detection Latency Under Increasing Offered Load.

Note. (a) Achieved versus offered throughput with concurrent classification of every window; the dotted red line is the write-only peak. (b) Median and 95th-percentile detection latency; at 20,000 records per second and above, latency includes queuing behind the classifier.

4.3 Closed-Loop Control Behavior

In the 120-second closed-loop scenario (Figure 8), winding temperature stood at 57.5 °C until the overheating fault began at 20 s. The first soft alert was issued in the onset window (20.0 s, severity 0.10) and the first load-adjustment command at 30.5 s, 10.5 s after onset, at a severity of 0.47 and a winding temperature of 68.0 °C. After the load setpoint was lowered from 85% to 65%, winding temperature peaked at 87.3 °C at 74.5 s and stayed below the 95 °C rated limit; the open-loop reference under the same fault schedule exceeded the limit from 53 s and peaked at 100.5 °C (Figure 8a). The classifier's overheating probability first reached τ at 30.5 s and fell below it in 10 of the 89 fault windows from that point on, the first of them at 62.0 s (Figure 8c); the load-adjustment response was issued in the other 79, and the 10 exceptions produced soft alerts. When the fault was cleared at 75 s and the load restored, seven soft alerts were issued between 75.0 and 78.0 s while the winding cooled from 86 °C to 80 °C, and no response was issued in any of the 40 fault-free windows before onset.

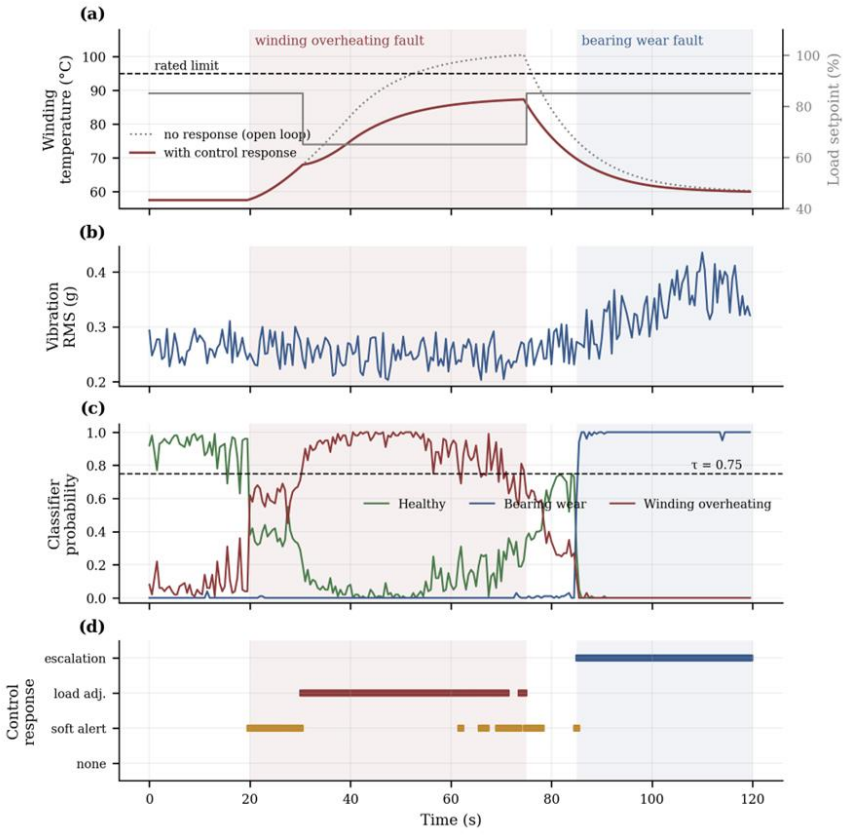


Figure 8. Closed-Loop Response to Winding Overheating and Bearing Wear Faults

Note. (a) Winding temperature with the control response (solid) and without it (dotted), the rated limit, and the load setpoint (gray). (b) Vibration RMS. (c) Classifier probabilities and the threshold τ. (d) Control response issued in each window. Shaded bands mark the fault episodes.

The bearing-wear fault began at 85 s. It produced a soft alert in the onset window (85.0 s, severity 0.10) and an escalation flag from 85.5 s, which was then issued in 69 of the 70 bearing-wear windows, while vibration RMS rose from a fault-free mean of 0.26 g to a mean of 0.37 g over the final 15 s (Figure 8b). Median end-to-end latency across the scenario was 92 ms.

The confidence threshold sets how far detection is traded against false actions (Figure 9). At the operating threshold τ = 0.75, 95.3% of faulted windows triggered a class-specific response and 0.9% of healthy windows did so; every class-specific response issued for a faulted window was the intended one. Lowering τ to 0.50 raised detection to 97.6% and false actions to 5.0%, while raising it to 0.90 lowered detection to 91.3% with no false actions (Figure 9a). At τ = 0.75, the intended response was issued for 99.7% of bearing-wear windows, 99.9% of misalignment windows, 99.3% of voltage-imbalance windows, and 82.1% of winding-overheating windows; a further 8.7% of overheating windows produced only a soft alert.

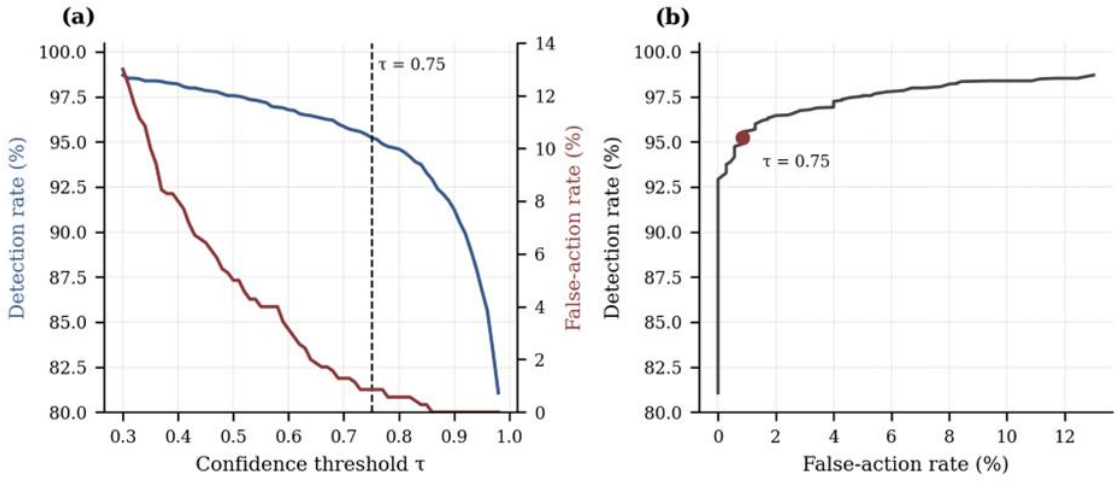


Figure 9. Effect of the Confidence Threshold on Detection and False Actions.

Note. (a) Detection rate on faulted windows and false-action rate on healthy windows against τ, from out-of-fold probabilities. (b) The same two rates plotted against each other; the marked point is the operating threshold.

5. DISCUSSION

The results in Section 4 should be read as evidence about software architecture, not about electrical machines as physical objects; the contribution this study makes lies in the orchestration of sensor fusion, the edge-to-cloud data pipeline, and the machine learning components that sit between them, not in any claim about motor electromagnetics. With that framing fixed, the findings can be set against the literature reviewed in Section 2.

The throughput and latency figures are best read against the demand a single machine places on the pipeline. Sampled at 5 kHz, one machine produces 5,000 records per second, and the pipeline kept pace with twice that rate while classifying every window, at a 95th-percentile detection latency of 145 ms, before saturating near 35,700 records per second, roughly seven times the single-machine rate (Figures 6 and 7). The latency is dominated by the database read and not by the model: the store query took a median of 85.7 ms of the 99 ms total, whereas feature extraction and classifier inference together took about 11.5 ms and the rule-based decision 16 µs. This sits close to the 101 to 111 millisecond per-batch inference latency Jakaria et al. (2025) report for their deep learning models, but the comparison is not like for like, since their figure isolates model inference and the present figure includes a round trip to the store. The practical reading is that a lightweight classifier leaves the storage path as the first place to optimize, through batched or cached window reads, for example. Doshi et al. (2024) argue that edge AI architectures have matured to the point where meaningful inference no longer requires a round trip to centralized cloud infrastructure; nothing in the present pipeline depended on cloud computing beyond the local InfluxDB instance, which is consistent with that claim, although a detection latency of about 100 milliseconds on a single shared core is a modest result and not a demonstration of edge-grade responsiveness.

Where this study's results diverge from Jakaria et al. (2025) is in classification accuracy. Their best hybrid deep learning models, trained on a million samples and evaluated against a more elaborate set of electrical and mechanical fault types, reached accuracies in the low 90s, whereas the present framework reached 97.0% under cross-validation. That gap is not, on inspection, evidence that this study's classifier is better, and the confusion matrix shows why. Bearing wear and mechanical misalignment were separated without error, which reflects the strong and mutually distinct signatures the generator gives them, and every remaining error sits at the boundary between healthy operation and winding overheating. The misclassified overheating windows were overwhelmingly mild ones (median severity 0.18, against 0.57 for overheating windows overall), and the healthy windows mistaken for overheating came from higher load and warmer ambient conditions (median load 0.86 against 0.66), the

combinations under which a healthy winding runs hottest in this generator. A synthetic dataset generated without the sensor drift, electrical noise, and load variability of a real industrial environment will tend to classify more easily than the real-world-adjacent data Jakaria et al. used, which is precisely why Section 3.5 opens with the caveat that these are software-architecture metrics rather than claims about physical machine behavior.

The closed-loop scenario supplies the evidence the second objective requires. In the 120-second run, the load-adjustment command was issued 10.5 seconds after the overheating fault began, at a fault severity of 0.47 and well before the winding reached the rated limit, and it held the simulated winding to a peak of 87.3 °C, where the same fault schedule without a response reached 100.5 °C and exceeded the limit from 53 seconds (Figure 8a). Two cautions apply. The thermal behavior comes from a first-order model written for this study, so the size of the temperature reduction shows that the control layer acts on the classifier's output as designed and not that a physical winding would respond by the same amount. And the run exposed a weakness in confidence-triggered rules: as the load reduction cooled the winding, the classifier's overheating probability fell below τ in 10 of the 89 windows after the first load adjustment (Figure 8c), so the emitted response dropped to a soft alert even though the fault was still present. In the simulation the load setpoint simply stayed reduced, but a controller that reverted the setpoint whenever the trigger lapsed would oscillate, and a hold time or latch on load-adjustment commands would be a natural refinement. The threshold sweep in Figure 9 shows a related trade-off: between $\tau = 0.50$ and $\tau = 0.90$, detection falls from 97.6% to 91.3% while false actions fall from 5.0% to zero, so the chosen $\tau = 0.75$ is a compromise and not an optimum.

The framework's central design decision, pairing monitoring with an explicit optimization layer rather than stopping at diagnosis, responds directly to a gap several reviewed studies leave open. Kurkin et al. (2026) demonstrate that multiparametric vibration diagnostics within a digital twin can catch faults that single-sensor approaches miss, but their framework, like much of the literature on machine tool and wind turbine monitoring (Constantino-Robles et al., 2026), stops short of specifying how detection translates into a control action. The present study's rule-triggered feedback loop is a modest answer to that gap rather than a comprehensive one. It is deliberately simple, and that simplicity is a limitation as much as a design choice. Hafner et al. (2025) show that learned world models can handle considerably more adaptive control than a fixed rule set, and a future version of this framework's optimization layer could plausibly incorporate that kind of learned policy instead of, or alongside, the explicit rules used here.

There is a recurring tension in the literature between interpretability and sophistication, and this study lands closer to the interpretable end of that spectrum than some recent work does. Fernandez del Amo et al. (2024) and Gispert et al. (2025) make the case, in building facilities and general industrial fault diagnosis respectively, that ontology-based or rule-explicit systems offer an auditability advantage over black-box classifiers, even where the black-box approach might achieve marginally higher raw accuracy. The present framework's control layer follows that logic. Whether this trade-off, transparency over marginal accuracy gains, holds up once the framework moves from simulated to physical data is an open question this study cannot resolve from synthetic results alone.

Set against the smart grid and power systems literature, this study's scope is comparatively narrow, and that narrowness is intentional rather than accidental. Mansour et al. (2023) survey condition monitoring and diagnosis across power transformers, electrical grids, and substations, operating at a scale well beyond a single machine. Hao et al. (2024) and Essien et al. (2024) likewise work at transmission-system scale, addressing voltage stability across an entire grid segment rather than at one machine's terminals. The present framework is built for the machine level specifically, and this is a deliberate scoping decision: a framework attempting to span both machine-level fault diagnosis and grid-level voltage stability simultaneously would likely do neither well. That said, the architectural pattern, four layers, with monitoring flowing up and optimization feedback flowing down, is not obviously incompatible with the grid-scale work Mansour et al. and others describe, and one plausible extension of this study would test whether the same layered logic scales upward.

A genuine limitation needs stating plainly. Everything reported in Section 4 comes from synthetic data. The timing results are also specific to the test environment, a single node with one CPU core shared by the data producer, the classification consumer, and the InfluxDB server, so absolute latency and throughput would be expected to differ on other hardware or database versions. Real motors generate noisier signals, drift over time in ways a parameterized generator does not fully capture, and operate under load profiles that vary in ways this study's controlled fault scenarios do not represent. Cao et al. (2025) report that digital-twin-enabled sim-to-real transfer worked reasonably well in their UAV coordination study, which offers some grounds for optimism that simulation-trained components can transfer to physical deployment without catastrophic performance loss; but their domain is sufficiently different from electrical machine monitoring that the comparison should be treated as suggestive rather than conclusive. The honest position is that this study demonstrates a software architecture works as designed under controlled conditions. Whether

it works as well against the noise of an actual factory floor is the question Section 6 turns to next.

6. CONCLUSION AND RECOMMENDATIONS

This study set out to do two things: design a layered digital twin framework for real-time condition monitoring of electrical machines, and demonstrate, through simulation rather than assertion, that the framework's control logic can act on what it detects. Both objectives were met within the scope this study defined for itself. The four-layer architecture, physical, data acquisition, analytics, and control, performed as intended under simulated load, sustaining 10,000 records per second while classifying every window, with a median detection latency of 99 milliseconds in sequential operation and a cross-validated classification accuracy of 97.0% across four distinct fault scenarios and the healthy baseline. In the closed-loop scenario, the load-adjustment response kept simulated winding temperature to a peak of 87.3 °C, where the same fault without a response reached 100.5 °C. These results say something meaningful about the software pipeline's viability. They do not, and should not be read to, say anything about how a physical motor would actually behave; that distinction has been kept deliberately explicit throughout.

The framework's main contribution is structural rather than algorithmic. Plenty of existing work, reviewed in Section 2, already does fault classification well. What this study adds is a demonstration that classification and control can be coupled into a single closed loop, with the analytics layer's output triggering a specific, auditable response rather than simply being logged for a human to review later. Given how often the reviewed literature stops at diagnosis (Kurkin et al., 2026; Constantino-Robles et al., 2026), that coupling is, modestly, the point of the exercise.

Three recommendations follow directly from the limitations discussed in Section 5. First, the framework needs testing against real sensor data from an operating electrical machine, not only the synthetic signals used here; this is the single most important next step, since every performance figure in this study depends on the assumption that real signals will behave similarly to the simulated ones, an assumption that has not yet been checked. Second, the rule-triggered control logic used in this study's optimization layer should be compared directly against a learned control policy, of the kind Hafner et al. (2025) demonstrate, to establish whether the interpretability this study prioritized comes at a meaningful cost in responsiveness or accuracy. Third, before any field deployment, the framework should be tested under hardware-in-the-loop conditions, where simulated sensor data is replaced

incrementally with signals from physical test rigs, allowing the gap between simulated and real performance to be measured directly rather than estimated.

Beyond these three, two further directions seem worth flagging for researchers working in this area rather than as commitments of this study itself. One is the question raised in Section 5 about whether this framework's layered architecture could extend to grid-scale monitoring of the kind Mansour et al. (2023) describe, rather than remaining confined to individual machines. The other concerns multi-machine scalability: the current framework was designed and tested around a single simulated machine, and whether the same InfluxDB-backed pipeline holds its throughput and latency characteristics when monitoring a fleet of machines simultaneously is an open empirical question that this study's single-machine scope cannot answer. The load sweep in Section 4.2 raises the aggregate write rate above one machine's, but it does not multiply the number of concurrent series or classification streams. Both are reasonable next steps for anyone extending this line of work, though neither was within reach of the present study's resources or aims.

REFERENCES

1. Adedeji, W. O., Adetayo, O. O., Pelemo, J., Oderinde, B. J., Okediran, I. K., Molokwu, M. I., Owoyomi, M. O., & Adufe, J. (2025). AI-driven smart factories: From predictive maintenance to autonomous production systems. Faculty of Natural and Applied Sciences Journal of Computing and Applications, 2(4), 87–94. https://doi.org/10.63561/jca.v2i4.1078
2. Aheleroff, S., Mostashiri, N., Xu, X., & Zhong, R. (2021). Mass personalisation as a service in Industry 4.0: A resilient response case study. Advanced Engineering Informatics, 50, 101438. https://doi.org/10.1016/j.aei.2021.101438
3. Ahmmed, M. S., Khan, L., Mahmood, M. A., & Liou, F. (2025). Digital twins, AI, and cybersecurity in additive manufacturing: A comprehensive review of current trends and challenges. Machines, 13(8), 691. https://doi.org/10.3390/machines13080691
4. Atieh, A., Cooke, K., & Osiyevskyy, O. (2022). The role of intelligent manufacturing systems in the implementation of Industry 4.0 by small and medium enterprises in developing countries. Engineering Reports, 5(1), e12578. https://doi.org/10.1002/eng2.12578
5. Cao, P., Lei, L., Shen, G., Cai, S., Liu, X., Liu, X., & Tian, S. (2025). UAV swarm cooperative search based on scalable multiagent deep reinforcement learning with digital

twin-enabled sim-to-real transfer. IEEE Transactions on Mobile Computing. https://doi.org/10.1109/TMC.2025.3530438

6. Constantino-Robles, C. D., Castillo Leonardo, F. A., Galván, J. H., Alvarez, Y. C., Carrera, L. A. I., & Rodríguez-Reséndiz, J. (2026). Vibration-based predictive maintenance for wind turbines: A PRISMA-guided systematic review on methods, applications, and remaining useful life prediction. Applied Mechanics, 7(1), 11. https://doi.org/10.3390/applmech7010011
7. Damtew, A. W. (2024). Roles of digital twins on material performances and utilization on upstream industry (The case of automotive industry). International Journal of Advanced Manufacturing Technology, 130, 3525–3536. https://doi.org/10.1007/s00170-023-12706-1
8. Doshi, S., Vora, K., & Mashru, D. (2024). Edge AI for low-power IoT devices: Architectures, algorithms, and applications. Educational Administration: Theory and Practice, 28(2), 12–28. https://doi.org/10.53555/kuey.v28i02.10499
9. Dutta, G., Kumar, R., Sindhwani, R., Singh, R. K., & Saihjpal, V. (2021). Digitalization priorities of quality control processes for SMEs: A conceptual study in perspective of Industry 4.0 adoption. Journal of Intelligent Manufacturing, 32, 1679–1698. https://doi.org/10.1007/s10845-021-01783-2
10. Erboz, G., & Yumurtacı Hüseyinoglu, I. Ö. (2023). The role of Industry 4.0 on supply chain cost and supply chain flexibility. Business Process Management Journal, 29(5), 1330–1351. https://doi.org/10.1108/BPMJ-11-2022-0605
11. Erunkulu, O. O., Gwebu, T. I., Zungeru, A. M., Lebekwe, C., & Modisa, M. (2023). Propagation channel characterization for mobile communication based on measurement campaign and simulation. Results in Engineering, 20, 101620. https://doi.org/10.1016/j.rineng.2023.101620
12. Essien, U. J., Odion, J., & Ekpa, A. K. (2024). Application of SSSC for voltage stability improvement in the Nigerian 330 kV transmission system using particle swarm optimization technique. International Journal of Multidisciplinary Research and Analysis, 7(2), 813–820. https://doi.org/10.47191/ijmra/v7-i02-51
13. Fernandez del Amo, I., Erkoyuncu, J. A., Bulka, D., Farsi, M., Ariansyah, D., Khan, S., & Wilding, S. (2024). Advancing fault diagnosis through ontology-based knowledge capture and application. IEEE Access, 12, 144599–144620. https://doi.org/10.1109/ACCESS.2024.3433412

14. Ferrigno, G., Del Sarto, N., Piccaluga, A., & Baroncelli, A. (2023). Industry 4.0 base technologies and business models: A bibliometric analysis. European Journal of Innovation Management, 26(7), 502–526. https://doi.org/10.1108/EJIM-02-2023-0107
15. Ghobakhloo, M., Iranmanesh, M., Foroughi, B., Babaee Tirkolaee, E., Asadi, S., & Amran, A. (2023). Industry 4.0 digital transformation and opportunities for supply chain resilience: A comprehensive review and a strategic roadmap. Production Planning & Control, 36(2), 195–227. https://doi.org/10.1080/09537287.2023.2252376
16. Gispert, D. E., Yitmen, I., Sadri, H., & Taheri, A. (2025). Development of an ontology-based asset information model for predictive maintenance in building facilities. Smart and Sustainable Built Environment, 14(3), 740–757. https://doi.org/10.1108/SASBE-07-2023-0170
17. Hafner, D., Pasukonis, J., Ba, J., & Lillicrap, T. (2025). Mastering diverse control tasks through world models. Nature. https://doi.org/10.1038/s41586-025-08744-2
18. Hao, W., Chen, M., & Gan, D. (2024). Short-term voltage stability analysis and enhancement strategies for power systems with photovoltaic penetration. IEEE Access, 12, 88728–88738. https://doi.org/10.1109/ACCESS.2024.3408903
19. Hassan, A., Bass, O., & Masoum, M. A. S. (2023). An improved genetic algorithm based fractional open circuit voltage MPPT for solar PV systems. Energy Reports, 9, 1535–1548. https://doi.org/10.1016/j.egyr.2022.12.088
20. Ho, T. M., Nguyen, K. K., & Cheriet, M. (2025). AI-powered digital twins for robotic control in 5G-enabled industrial automation. IEEE Journal on Selected Areas in Communications. https://doi.org/10.1109/JSAC.2025.3574625
21. Hu, J., Xiao, H., Ye, Z., Luo, N., & Zhou, M. (2025). Research and prospects of digital twin-based fault diagnosis of electric machines. Sensors, 25(8), 2625. https://doi.org/10.3390/s25082625
22. Jafari, M., Kavousi-Fard, A., Chen, T., & Karimi, M. (2023). A review on digital twin technology in smart grid, transportation system and smart city: Challenges and future. IEEE Access, 11, 17471–17484. https://doi.org/10.1109/ACCESS.2023.3241588
23. Jakaria, J. M., Sabir, J., Rahman, M. Z., & Ali, M. F. (2025). Hybrid deep learning framework for real-time fault detection in squirrel-cage induction motors. PLOS ONE, 20(11), e0336323. https://doi.org/10.1371/journal.pone.0336323
24. Kang, Y. S., Choi, J. G., Yang, S. M., Lim, S., Kim, D. C., & Park, H. W. (2025). Artificial intelligence augmented digital twin for improving the machinability in a

robotic carbon fiber reinforced plastics machining process. Journal of Intelligent Manufacturing. https://doi.org/10.1007/s10845-025-02642-0

25. Kia, S. H., Dunai, L., Antonino-Daviu, J. A., & Razik, H. (2025). Real-time digital twins for intelligent fault diagnosis and condition-based monitoring of electrical machines. Energies, 18(17), 4637. https://doi.org/10.3390/en18174637
26. Kurkin, A., Kabaldin, Y., Zhelonkin, M., Mancerov, S., Anosov, M., & Shatagin, D. (2026). Multiparametric vibration diagnostics of machine tools within a digital twin framework using machine learning. Applied Sciences, 16(2), 982. https://doi.org/10.3390/app16020982
27. Langas, E. F., Zafar, M. H., & Sanfilippo, F. (2025). Exploring the synergy of human-robot teaming, digital twins, and machine learning in industry 5.0: A step towards sustainable manufacturing. Journal of Intelligent Manufacturing. https://doi.org/10.1007/s10845-025-02580-x
28. Li, T., Long, Q., Chai, H., Zhang, S., Jiang, F., Liu, H., Niyato, D., & Han, Z. (2025). Generative AI empowered network digital twins: Architecture, technologies, and applications. ACM Computing Surveys, 57(6), 1–43. https://doi.org/10.1145/3711682
29. Lwahas, N. A., Lawal, S. L., Aduku, L. S., Oyekola, S. A., & Surajo, A. (2025). Recent advances in machine learning for fault diagnosis in mechatronic systems. Global Journal of Research in Engineering & Computer Sciences, 5(4), 1–6. https://doi.org/10.5281/zenodo.15852048
30. Mansour, D. A., Numair, M., Zalhaf, A. S., Ramadan, R., Darwish, M. M. F., Huang, Q., Hussien, M. G., & Abdel-Rahim, O. (2023). Applications of IoT and digital twin in electrical power systems: A comprehensive survey. IET Generation, Transmission & Distribution, 17(20), 4457–4479. https://doi.org/10.1049/gtd2.12940
31. Peksa, J., & Mamchur, D. (2026). State-of-the-art on digital twin technologies for industrial applications and the federated digital twin lifecycle model (F-DTLM). Automation, 7(3), 77. https://doi.org/10.3390/automation7030077
32. Yabani, G. G., Asogwa, E. C., Ogbonna, I. J., Akintade, K. O., Odimegwu, A. A., & Okwo, S. N. (2026). Machine learning for voltage stability in Nigeria's power network: A comprehensive review. Engineering Innovations, 18, 67–84. https://doi.org/10.4028/p-iWL0G6