

TinyML-Based Predictive Maintenance for Resource-Constrained Industrial IoT Devices: A Lightweight Edge Intelligence Framework

Dr. Rohan Patel¹

Research Scholar¹

Department of Computer Science and Engineering
National Institute of Technology, Warangal, Telangana, India
rohan.patel@nitw.ac.in

Abstract

The Industrial Internet of Things (IIoT) has transformed industrial monitoring by connecting machines, sensors, controllers, and analytical platforms through intelligent communication networks. Continuous monitoring of industrial equipment creates opportunities for predictive maintenance, where machine-learning techniques can identify abnormal operating conditions before equipment failure occurs. However, conventional cloud-based machine-learning approaches often require continuous transmission of sensor data to remote servers. This can increase communication overhead, latency, network dependency, and energy consumption. Tiny Machine Learning (TinyML) provides an alternative by enabling lightweight machine-learning models to execute directly on resource-constrained edge devices and microcontrollers. This paper proposes a conceptual TinyML-based predictive maintenance framework for industrial IoT environments. The proposed framework collects vibration, temperature, acoustic, current, and operational measurements from industrial equipment and performs local preprocessing and inference at the edge. Lightweight machine-learning models are optimized using techniques such as feature reduction, quantization, pruning, and compact model architectures. The framework introduces an adaptive anomaly score that combines multiple sensor characteristics to identify early signs of equipment degradation. Instead of continuously transmitting raw sensor streams to a cloud platform, only selected events, summarized features, or maintenance alerts are communicated to higher-level systems. This approach can reduce network traffic and enable faster local decisions. The paper analyzes the architecture, feature-extraction process, anomaly-detection mechanism, model optimization strategy, communication design, security considerations, and evaluation methodology. Important challenges including limited memory, computational capac-



ity, sensor noise, model drift, hardware heterogeneity, and explainability are also discussed. The proposed framework provides a research-oriented architecture for deploying efficient predictive-maintenance intelligence on low-power industrial devices.

Keywords: TinyML, Industrial Internet of Things, Predictive Maintenance, Edge AI, Machine Learning, Anomaly Detection, Embedded Systems, Model Compression, Edge Computing, Intelligent Manufacturing

1. Introduction

Modern industrial environments contain a large number of machines and sensors that continuously generate operational information. Temperature sensors, vibration sensors, acoustic sensors, current sensors, pressure sensors, and rotational-speed sensors can provide information about machine health.

A conventional monitoring system may follow:

Machine → Sensors → Cloud → Machine Learning → Maintenance Decision Machine
Sensors → Cloud → Machine Learning → Maintenance Decision

Although this architecture provides substantial computational capability, it requires continuous communication between industrial equipment and remote infrastructure.

For high-frequency sensor systems:

$\text{DataVolume} \uparrow \Rightarrow \text{NetworkTraffic} \uparrow$

and:

$\text{NetworkTraffic} \uparrow \Rightarrow \text{CommunicationEnergy} \uparrow$

There can also be additional latency between the occurrence of an abnormal condition and the generation of a maintenance decision.

Edge computing moves computation closer to the source of data.

TinyML extends this concept by enabling compact machine-learning models to operate on highly resource-constrained devices. Recent research identifies low memory, computational limitations, energy efficiency, and real-time inference as important considerations for intelligent edge deployment. ([MDPI](#))

In industrial environments, this creates the possibility of:

Sensor→TinyML→LocalDecisionSensor \rightarrow TinyML \rightarrow LocalDecision

without requiring every measurement to be transmitted to the cloud.

This paper proposes a conceptual architecture for applying TinyML to predictive maintenance in IIoT environments.

2. Background

2.1 Industrial Internet of Things

The Industrial Internet of Things connects physical industrial assets with sensing, communication, computing, and analytical technologies.

A simplified IIoT environment can be represented as:

IIoT=Machines+Sensors+Networks+AnalyticsIIoT= Machines+Sensors+Networks+Analytics

Sensors continuously observe machine behavior.

The resulting data can be used for:

- Equipment monitoring
- Fault detection
- Predictive maintenance
- Production optimization
- Energy management
- Safety monitoring

2.2 Predictive Maintenance

Traditional maintenance can be divided into several approaches.

Corrective Maintenance

Equipment is repaired after failure.

Failure→Repair

Preventive Maintenance

Maintenance occurs according to a predefined schedule.

Time→Maintenance

Predictive Maintenance

Maintenance decisions are based on equipment condition.

Condition→Prediction→Maintenance

Predictive maintenance attempts to identify early degradation before complete equipment failure.

3. Motivation

Unexpected equipment failures can result in:

- Production interruption
- Maintenance costs
- Replacement costs
- Safety risks
- Reduced productivity

Therefore, early fault detection is important.

Suppose machine health is represented by:

$$H(t)$$

Under normal operation:

$$H(t) \approx H_{\text{normal}}$$

As degradation increases:

$$H(t) \rightarrow H_{\text{critical}}$$

A predictive-maintenance model attempts to identify this transition before failure occurs.

4. Problem Statement

The primary problem addressed by this study is:

How can lightweight machine-learning models be deployed directly on resource-constrained industrial IoT devices to detect equipment anomalies while minimizing communication overhead, energy consumption, and response latency?

Traditional cloud-centric approaches may encounter several problems:

1. Continuous network dependency.
2. Large sensor-data transmission.
3. Communication latency.
4. Increased bandwidth consumption.
5. Privacy concerns.
6. Dependence on cloud availability.

TinyML provides an opportunity to perform inference locally.

Therefore:

$\text{RawSensorData} \rightarrow \text{LocalInference} \rightarrow \text{Alert}$

can replace:

RawSensorData→Cloud→Inference→AlertRawSensorData \rightarrow Cloud \rightarrow Inference
\rightarrow Alert

for appropriate applications.

5. Research Objectives

The objectives of this research are:

1. To investigate TinyML for industrial predictive maintenance.
2. To design a lightweight edge intelligence architecture.
3. To identify useful machine-health features.
4. To develop a conceptual anomaly-detection mechanism.
5. To reduce unnecessary sensor-data transmission.
6. To investigate model-compression techniques.
7. To analyze resource constraints.
8. To define evaluation metrics.
9. To identify future research opportunities.

6. Research Contributions

The main contributions of this paper are:

- A conceptual TinyML architecture for IIoT predictive maintenance.
- A multi-sensor feature-processing strategy.
- A lightweight anomaly-scoring mechanism.
- A model-optimization pipeline.
- An event-driven communication strategy.
- A resource-aware inference architecture.
- A security model for edge intelligence.
- A research methodology for experimental evaluation.

7. Proposed Architecture

The proposed framework contains five major layers.

Layer 1: Industrial Equipment

Machines generate physical operating signals.

Layer 2: Sensor Layer

Sensors collect machine-condition measurements.

Layer 3: TinyML Edge Layer

The embedded device performs preprocessing and machine-learning inference.

Layer 4: Gateway Layer

The gateway aggregates alerts and communicates with industrial systems.

Layer 5: Cloud or Enterprise Layer

The cloud performs historical analysis, model management, visualization, and maintenance planning.

The architecture is:

Machine → Sensors → TinyML → Gateway → Cloud
Machine \rightarrow Sensors \rightarrow TinyML \rightarrow Gateway \rightarrow Cloud

8. Sensor Data

The proposed system can use multiple types of measurements.

8.1 Vibration

Vibration is useful for identifying:

- Bearing problems
- Misalignment
- Imbalance
- Mechanical looseness

8.2 Temperature

Temperature measurements can identify abnormal heating.

$T(t) > T_{\text{threshold}}$

may indicate a possible problem.

8.3 Acoustic Signals

Abnormal sounds can indicate mechanical changes.

8.4 Electrical Current

Current patterns can reveal changes in motor operation.

8.5 Rotational Speed

Changes in rotational speed may indicate mechanical or control-related problems.

9. Multi-Sensor Data Model

Let:

$X(t) = [x_1(t), x_2(t), \dots, x_m(t)]$

where each $x_{i,t}$ represents a sensor measurement.

For example:

$$X(t) = [\text{Vibration}, \text{Temperature}, \text{Current}, \text{Speed}]$$

The model uses these measurements to estimate machine condition.

10. Data Acquisition

Sensors periodically collect measurements.

Suppose the sampling interval is:

$$\Delta t$$

Then:

$$X_t = X(t)$$

is collected every Δt .

The sensor data is divided into windows.

For a window of length W :

$$X_W = \{X_t, X_{t+1}, \dots, X_{t+W}\}$$

This allows temporal characteristics to be extracted.

11. Signal Preprocessing

Industrial sensor data can contain:

- Noise
- Missing values

- Outliers
- Sensor drift

Therefore, preprocessing is necessary.

A conceptual pipeline is:

RawData→Filtering→Normalization→Windowing→FeatureExtraction
RawData \rightarrow Filtering \rightarrow Normalization \rightarrow Windowing \rightarrow FeatureExtraction

12. Normalization

Different sensors can have different numerical ranges.

For a measurement x :

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

This transforms the value into a normalized range.

Normalization can help lightweight models operate consistently across different sensor inputs.

13. Feature Extraction

Instead of transmitting raw signals, the edge device can calculate compact features.

For a signal:

$$X = \{x_1, x_2, \dots, x_n\}$$

the mean is:

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$

Variance is:

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$$

Standard deviation is:

$$\sigma = \sqrt{\sigma^2}$$

Other possible features include:

- Minimum
- Maximum
- RMS
- Peak value
- Crest factor
- Signal energy
- Frequency-domain characteristics

14. RMS Vibration Feature

For vibration data, root mean square can be calculated as:

$$RMS = \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}$$

An increase in RMS can indicate a change in machine vibration behavior.

15. Frequency-Domain Features

Some mechanical faults are easier to detect in the frequency domain.

A discrete Fourier transformation can conceptually represent:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}$$

The resulting spectrum can contain information about characteristic machine frequencies.

However, frequency-domain processing can increase computational requirements on small devices.

Therefore, the proposed framework should select only the features that provide sufficient predictive value.

16. Feature Selection

Using too many features increases:

Memory Usage

and:

Computation

Therefore, feature selection is necessary.

Let:

$$F = \{f_1, f_2, \dots, f_n\}$$

The objective is to select:

$$F' \subset F$$

such that:

$$|F'| \ll |F|$$

while maintaining useful predictive information.

17. TinyML Model

The proposed system uses a lightweight machine-learning model.

Possible model families include:

- Decision trees

- Logistic regression
- Small neural networks
- Tiny convolutional networks
- Compact autoencoders
- Random forests with constrained complexity

The appropriate model depends on:

Accuracy+Memory+Latency+Energy

18. Anomaly Detection

The model produces an anomaly score:

$$A=f(X)$$

where:

$$0 \leq A \leq 1$$

A threshold can be defined:

$$\tau$$

If:

$$A < \tau$$

the equipment is considered normal.

If:

$$A \geq \tau$$

the system generates an anomaly alert.

19. Multi-Level Machine Health

The proposed system can classify equipment into multiple conditions.

Level 0 — Normal

$$A < \tau_1$$

Level 1 — Warning

$$\tau_1 \leq A < \tau_2$$

Level 2 — Critical

$$A \geq \tau_2$$

This allows maintenance teams to prioritize equipment.

20. Adaptive Threshold

A fixed threshold may not be appropriate for every machine.

Therefore:

$$\tau(t) = f(\text{HistoricalData}, \text{OperatingCondition})$$

can be used.

For example, machine behavior during high production loads may differ from idle behavior.

21. Machine Operating Context

The same sensor measurement may have different meanings under different operating conditions.

Therefore, the system should consider:

Context=Load+Speed+Temperature+OperatingMode
Context=Load+Speed+Temperature+OperatingMode

The anomaly model can then calculate:

$$A=f(X,Context)A=f(X,Context)$$

This may reduce false alarms.

22. Model Compression

Machine-learning models designed for cloud systems may be too large for microcontrollers.

Model compression can reduce resource requirements.

Important techniques include:

- Quantization
- Pruning
- Knowledge distillation
- Architecture simplification

Recent work on intelligent edge computing identifies model compression techniques such as pruning, quantization, knowledge distillation, and low-rank methods as important approaches for resource-constrained deployment. ([MDPI](#))

23. Quantization

A conventional neural network may use floating-point parameters.

Quantization converts parameters to lower-precision representations.

For example:

$$FP32 \rightarrow INT8$$

This can reduce:

- Model size
- Memory consumption
- Computational cost

However, excessive quantization can reduce model accuracy.

Therefore:

Compression $\leftrightarrow$ Accuracy

must be balanced.

24. Pruning

Pruning removes less important model parameters.

Suppose:

$$M = \{w_1, w_2, \dots, w_n\}$$

Some parameters may have negligible influence.

They can be removed:

$$M' \subset M$$

This can reduce model complexity.

25. Knowledge Distillation

A large teacher model can train a smaller student model.

$$\text{LargeModel} \rightarrow \text{Knowledge} \rightarrow \text{SmallModel}$$

The student model is then deployed on the edge.

This allows knowledge from a complex model to be transferred into a lightweight architecture.

26. TinyML Deployment Pipeline

The proposed deployment pipeline is:

DataCollection→ModelTraining→Compression→Conversion→EdgeDeployment→Inference
DataCollection \rightarrow ModelTraining \rightarrow Compression \rightarrow Conversion \rightarrow EdgeDeployment \rightarrow Inference

Training can be performed using powerful computing infrastructure.

Inference is performed locally on the industrial device.

27. Training and Inference Separation

A key principle is:

Training $\neq$ Inference

Training may require substantial computational resources.

Inference can be optimized for embedded hardware.

Thus:

Cloud/Server→Training
Cloud/Server \rightarrow Training

and:

EdgeDevice→Inference
EdgeDevice \rightarrow Inference

28. Event-Driven Communication

The proposed system does not continuously transmit all raw sensor information.

Instead:

NormalCondition $\rightarrow$ LocalProcessing NormalCondition $\rightarrow$ LocalProcessing

while:

AbnormalCondition $\rightarrow$ AlertTransmission AbnormalCondition $\rightarrow$ AlertTransmission

This can reduce network traffic.

29. Communication Reduction

Suppose a sensor generates:

D_{raw}

bytes per hour.

If only anomalies are transmitted:

$D_{alert} \ll D_{raw}$

Therefore:

$BandwidthSavings = 1 - \frac{D_{alert}}{D_{raw}}$

This is a conceptual measure of communication reduction.

30. Edge Gateway

The gateway acts as an intermediate layer.

It can:

- Collect alerts
- Store recent measurements
- Manage devices
- Synchronize models
- Forward maintenance events
- Communicate with enterprise systems

The architecture becomes:

TinyMLNode→Gateway→IndustrialPlatform TinyMLNode \rightarrow Gateway \rightarrow IndustrialPlatform

31. Cloud Integration

The cloud remains useful for tasks that require significant resources.

Examples include:

- Historical analysis
- Model retraining
- Fleet-level monitoring
- Model version management
- Visualization
- Long-term storage

Thus, the proposed system does not eliminate cloud computing.

Instead:

Edge+Cloud Edge+Cloud

are used together.

32. Hybrid Intelligence Architecture

The proposed architecture divides intelligence according to computational requirements.

Edge

FastInferenceFastInference

Gateway

Aggregation+CoordinationAggregation+Coordination

Cloud

DeepAnalysis+TrainingDeepAnalysis+Training

This creates a hierarchical intelligence model.

33. Predictive Maintenance Decision

The final maintenance decision can be represented as:

Decision=f(AnomalyScore,MachineAge,OperatingHours,History)Decision=
f(AnomalyScore,MachineAge,OperatingHours,History)

A simple risk score can be defined as:

$R=w_1A+w_2D+w_3H+w_4O$ OR= $w_1A+ w_2D+ w_3H+ w_4O$

where:

- AA = anomaly score
- DD = degradation indicator
- HH = historical failure information
- OO = operating-hour factor

34. Maintenance Priority

Machines can be assigned different priority levels.

Priority = $\begin{cases} \text{Low}, & R < \tau_1 \\ \text{Medium}, & \tau_1 \leq R < \tau_2 \\ \text{High}, & R \geq \tau_2 \end{cases}$

This allows maintenance teams to focus on the most critical equipment.

35. Remaining Useful Life

A more advanced system can estimate remaining useful life.

Let:

$RUL(t)$

represent the expected remaining operating time.

The prediction becomes:

$RUL = f(X_{1:t})$

However, RUL prediction is more difficult than simple anomaly detection because it requires reliable degradation histories and failure information.

36. Model Drift

Industrial machines can change over time.

For example:

$P_{data}(t_1) \neq P_{data}(t_2)$

This phenomenon can reduce model performance.

Therefore, the system should monitor model performance continuously.

If:

$Accuracy(t) < \tau Accuracy(t) < \tau$

retraining may be required.

37. Periodic Model Updating

A model-management system can follow:

Monitor → DetectDrift → Retrain → Compress → Deploy → Monitor → DetectDrift → Retrain → Compress → Deploy

This creates a continuous machine-learning lifecycle.

38. Security Challenges

TinyML-based IIoT systems can face multiple threats.

These include:

- Device compromise
- Firmware manipulation
- Model tampering
- False sensor data
- Unauthorized model extraction
- Communication interception
- Adversarial inputs

Security must therefore be incorporated throughout the architecture.

39. Secure Communication

Data transmitted between the edge device and gateway should use authenticated and encrypted communication.

Conceptually:

Message $\rightarrow$ Encrypt $\rightarrow$ Transmit $\rightarrow$ Authenticate $\rightarrow$ Decrypt Message $\rightarrow$ Encrypt $\rightarrow$ Transmit $\rightarrow$ Authenticate $\rightarrow$ Decrypt

This reduces the risk of unauthorized interception.

40. Firmware Integrity

The deployed TinyML model should be treated as part of the device's trusted software.

A firmware image can have a cryptographic hash:

$H = \text{Hash}(\text{Firmware})$

The expected hash can be compared with the current device hash.

If:

$H' \neq H$

the firmware may have been modified.

41. Model Integrity

The same principle can be applied to the machine-learning model.

Let:

$H_M = \text{Hash}(\text{Model})$

The device can verify:

$\text{Hash}(\text{Model}_{\text{installed}}) = \text{HMDHash}(\text{Model}_{\{\text{installed}\}}) = \text{H_M}$

before loading the model.

42. Sensor Integrity

Machine-learning predictions are only as reliable as their input signals.

A compromised sensor can produce:

$X_{\text{false}} \{ \text{false} \}$

which may cause:

$\text{Prediction}_{\text{false}} \{ \text{false} \}$

Therefore, sensor-level validation is necessary.

43. Sensor Anomaly Detection

The framework can detect abnormal sensor behavior.

For example:

$|x_t - x_{t-1}| > \delta |x_t - x_{t-1}| > \delta$

may indicate a sudden measurement change.

However, legitimate machine events can also cause rapid changes.

Therefore, sensor anomaly detection should consider operational context.

44. Explainability

Maintenance engineers may want to understand why a system generated an alert.

Instead of:

Alert=TrueAlert=True

the system can provide:

Alert=TrueAlert=True

because:

Vibration↑Vibration\uparrow

and:

Temperature↑Temperature\uparrow

and:

RMS↑RMS\uparrow

This makes the system more useful for human decision-making.

45. Resource Constraints

TinyML devices typically operate under strict limitations.

Important constraints include:

MemoryMemory CPU CPU StorageStorage EnergyEnergy CommunicationCommunication

A model that performs well on a desktop computer may not be suitable for a microcontroller.

46. Memory Optimization

Model memory can be approximated as:

$$\text{Memory} \approx \text{Parameters} \times \text{BytesPerParameter}$$

If:

$$\text{Parameters} = 1,000,000$$

and each parameter uses 4 bytes:

$$\text{Memory} \approx 4\text{MB}$$

Quantization can substantially reduce the parameter representation.

47. Computational Complexity

For a model containing NN operations:

$$T_{\text{inference}} \propto N^2$$

A smaller model generally requires fewer operations.

Therefore:

$$\text{ModelSize} \downarrow \Rightarrow \text{PotentialLatency} \downarrow$$

provided implementation and hardware characteristics are favorable.

48. Energy Consumption

Total device energy can be approximated as:

$E_{total} = E_{sensing} + E_{processing} + E_{communication}$
 $E_{total} = E_{sensing} + E_{processing} + E_{communication}$

In many IoT environments, communication can be a significant energy consideration.

Local inference can reduce the need to continuously transmit raw sensor data.

Thus:

Communication $\downarrow \Rightarrow$ Potential Energy $\downarrow$ Communication $\downarrow \Rightarrow$ Potential Energy $\downarrow$

The actual savings depend on the hardware, workload, network technology, and inference frequency.

49. Latency

End-to-end decision latency can be expressed as:

$L = L_{sensing} + L_{processing} + L_{communication} + L_{decision}$
 $L = L_{sensing} + L_{processing} + L_{communication} + L_{decision}$

A cloud-centric architecture may contain a relatively large communication component.

Edge inference can reduce:

$L_{communication}$

by making decisions locally.

Recent literature on TinyML and edge intelligence highlights local inference as a mechanism for reducing latency and network traffic in resource-constrained IoT systems. ([ScienceDirect](#))

50. Proposed Evaluation Metrics

The proposed framework should be evaluated using:

Accuracy

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}$$

Precision

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F1-Score

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Inference Latency

$$T_{\text{inference}}$$

Memory Usage

$$M_{\text{model}}$$

Energy Per Inference

$$E_{\text{inference}}$$

Communication Volume

$$C_{\text{data}}$$

51. Experimental Methodology

A future experimental implementation can use a representative industrial machine dataset.

The dataset can be divided into:

TrainingSet ValidationSet TestSet

The training set is used for model development.

The validation set is used for parameter selection.

The test set is used for final evaluation.

52. Experimental Models

Several models can be compared.

Model A

Traditional machine-learning model.

Model B

Small neural network.

Model C

Quantized neural network.

Model D

Pruned and quantized model.

Model E

Proposed TinyML model.

Comparison can consider:

Accuracy, Memory, Latency, EnergyAccuracy,\ Memory,\ Latency,\ Energy

53. Cloud Versus Edge Evaluation

The same predictive-maintenance task can be evaluated under two architectures.

Cloud Architecture

Sensor→Network→Cloud→Prediction
Sensor \rightarrow Network \rightarrow Cloud \rightarrow Prediction

Proposed Architecture

Sensor→TinyML→Prediction
Sensor \rightarrow TinyML \rightarrow Prediction

The comparison can evaluate:

- Response time
- Network traffic
- Energy consumption
- Detection accuracy
- Reliability

54. Expected Results

The proposed architecture is expected to reduce unnecessary communication because routine sensor information can be processed locally.

The expected relationship is:

RawTransmission_{cloud} > AlertTransmission_{edge}
RawTransmission_{cloud} > AlertTransmission_{edge}

Similarly, local inference may reduce decision latency because the system does not always require remote processing.

However, these are expected architectural outcomes and should be validated experimentally rather than presented as measured results.

55. Communication Analysis

Suppose:

R_{sR_s}

is the sensor data generation rate.

Cloud architecture communication is approximately:

$C_{cloud} \approx R_{sC_cloud} \approx R_s$

if all sensor information is transmitted.

Under event-driven edge processing:

$C_{edge} \approx R_{aC_edge} \approx R_a$

where R_{aR_a} is the alert or summarized-data rate.

If:

$R_a \ll R_{sR_a} \parallel R_s$

then:

$C_{edge} \ll C_{cloud} \parallel C_cloud$

56. False Alarm Management

Predictive-maintenance systems can produce false positives.

A false positive occurs when:

Prediction=Fault Prediction=Fault

while:

Actual=NormalActual=Normal

Excessive false alarms can reduce user confidence.

Therefore, the system should optimize:

Precision+RecallPrecision+Recall

rather than accuracy alone.

57. Fault Detection Delay

Another important metric is fault detection delay.

Let:

T_{ft_f}

be the actual fault-development time and:

T_{dt_d}

the detection time.

Then:

$\text{DetectionDelay} = T_d - T_f$

A useful predictive-maintenance system should minimize detection delay while maintaining acceptable false-alarm levels.

58. Model Reliability

The model should remain stable under:

- Sensor noise
- Temperature variation
- Load variation
- Device aging
- Network interruptions

Therefore:

$Robustness = Performance_{normal} + Performance_{variation}$
 $Robustness = Performance_{\{normal\}} + Performance_{\{variation\}}$

should be evaluated.

59. Offline Operation

One advantage of local inference is the possibility of operating when the network connection is temporarily unavailable.

If:

$Network = Unavailable$

the TinyML model can continue:

$Sensor \rightarrow LocalInference$

The system can transmit stored alerts when connectivity returns.

This can improve operational resilience.

60. Distributed Fleet Management

Large factories may contain hundreds or thousands of machines.

Each device may run:

Model_i

A central management platform can maintain model versions:

$M_1, M_2, \dots, M_n$

and determine which model should be deployed to each device.

61. Model Versioning

Every model should have a version identifier:

$Version(M) = v$

For example:

$M_{v1} \rightarrow M_{v2} \rightarrow M_{v3}$

The system should maintain a record of which device uses which model.

62. Safe Model Deployment

A new model should be validated before deployment.

The process can be:

$Train \rightarrow Validate \rightarrow Compress \rightarrow TestOnHardware \rightarrow Deploy$

Hardware testing is important because software simulation may not accurately represent embedded-device performance.

63. Hardware-in-the-Loop Testing

The proposed research can use hardware-in-the-loop evaluation.

The model is deployed to a target device while sensor inputs are generated from realistic industrial data.

This allows researchers to measure:

- Actual inference latency
- Memory consumption
- Energy consumption
- Accuracy
- Device temperature

64. Hardware Heterogeneity

Different industrial installations may use different microcontrollers.

Therefore:

$Hardware_A \neq Hardware_B$

A model optimized for one platform may not provide identical performance on another.

The deployment pipeline should therefore include hardware-specific optimization.

65. Model Portability

A portable TinyML system should separate:

Model Logic

from:



HardwareImplementationHardwareImplementation

This allows the same predictive-maintenance architecture to be adapted to multiple embedded platforms.

66. Edge AI and Sustainability

Large-scale IoT deployments can generate significant computational and communication workloads.

Moving selected intelligence to the edge can reduce upstream traffic and support more efficient processing. Recent literature specifically identifies energy efficiency, communication optimization, and hardware-aware strategies as major research directions for intelligent edge and IoT systems. ([ScienceDirect](#))

However, edge AI is not automatically energy-efficient.

The energy cost of:

InferenceInference

must be compared with the energy cost of:

CommunicationCommunication

The appropriate architecture therefore depends on the workload and hardware.

67. Privacy Considerations

Industrial sensor data may reveal information about:

- Production rates
- Equipment conditions
- Operating schedules
- Manufacturing processes

Sending all information to remote infrastructure can increase exposure.

Local processing can reduce the amount of operational data leaving the industrial environment.

Thus:

LocalInference → ReducedDataExposure

although it does not eliminate all security risks.

68. Secure Edge AI

A secure deployment should protect:

1. Sensor data.
2. Device identity.
3. Firmware.
4. Machine-learning model.
5. Communication channels.
6. Maintenance alerts.

A layered architecture is:

Security = DeviceSecurity + ModelSecurity + NetworkSecurity + ApplicationSecurity

69. Adversarial Inputs

An attacker could deliberately manipulate sensor measurements.

For example:

$X \rightarrow X + \delta X$

where δ is an intentional perturbation.

The objective may be to cause:

$Prediction(X+\delta) \neq Prediction(X)$

Therefore, robustness against abnormal or manipulated inputs should be investigated.

70. Model Theft

A deployed model may contain valuable industrial knowledge.

An attacker with physical access could attempt to extract the model.

Possible protections include:

- Secure storage
- Hardware-backed keys
- Secure boot
- Access control
- Model encryption

71. Secure Boot

The device can verify software before execution.

Conceptually:

$Verify(Firmware)$

If verification succeeds:

$Boot = True$

Otherwise:

$Boot = False$

This can help prevent unauthorized software from running.

72. Explainable Maintenance Alerts

A useful system should provide actionable information.

Instead of only reporting:

FaultProbability=0.87FaultProbability=0.87

the system can report:

High-risk condition due to abnormal vibration and increasing operating temperature.

This allows maintenance personnel to understand the reason for the recommendation.

73. Human-in-the-Loop Maintenance

The proposed framework does not automatically replace maintenance professionals.

Instead:

AI→Recommendation→Engineer→DecisionAI \rightarrow Recommendation \rightarrow Engineer \rightarrow Decision

This allows humans to review high-risk decisions.

74. Proposed End-to-End Workflow

The complete system can operate as follows:

1. Sensors collect machine measurements.
2. Edge hardware receives measurements.
3. Data is filtered.
4. Measurements are normalized.
5. Features are extracted.
6. TinyML model performs inference.

7. Anomaly score is calculated.
8. Machine-health status is generated.
9. Normal events remain local.
10. Important events are transmitted.
11. Gateway forwards alerts.
12. Maintenance platform records events.
13. Historical data supports future model training.
14. Improved models are compressed.
15. Updated models are deployed.

The complete pipeline is:

Sense→Process→Infer→Decide→Communicate→Maintain
Sense \rightarrow Process \rightarrow Infer \rightarrow Decide \rightarrow Communicate \rightarrow Maintain

75. Advantages of the Proposed Framework

75.1 Low Latency

Local inference can provide rapid responses.

75.2 Reduced Communication

Only selected information needs to be transmitted.

75.3 Improved Resilience

Local inference can continue during network interruptions.

75.4 Lower Data Exposure

Less raw operational data needs to leave the industrial environment.



75.5 Resource Awareness

Model compression allows deployment on constrained devices.

75.6 Scalability

Multiple edge devices can independently monitor machines.

76. Limitations

The proposed architecture has several limitations.

Limited Hardware Resources

TinyML devices cannot execute arbitrary large models.

Model Accuracy

Compression may reduce predictive performance.

Sensor Quality

Poor sensor measurements can produce unreliable predictions.

Model Drift

Machine behavior changes over time.

Hardware Diversity

Different devices have different capabilities.

Security

Physical access to edge devices creates additional attack opportunities.

Maintenance Data

Reliable predictive models require representative historical failure data.

77. Research Challenges

Several research challenges remain.

Challenge 1: Accurate Lightweight Models

The model must remain accurate while being small.

Accuracy $\uparrow$ and Memory $\downarrow$ Accuracy $\uparrow$ $\quad$ and $\quad$ Memory $\downarrow$

Challenge 2: Energy-Aware Inference

Inference must not consume excessive energy.

Challenge 3: Robustness

The model must work under noisy industrial conditions.

Challenge 4: Model Updating

Models must adapt to machine aging.

Challenge 5: Security

Edge devices must be protected against physical and remote attacks.

78. Future Research Directions

78.1 Federated TinyML

Multiple factories could collaboratively improve models without sharing raw industrial data.

Factory1+Factory2+...+Factorn→SharedModelFactory_1+Factory_2+⋯+Factory_n \rightarrow SharedModel

without centralizing all raw datasets.

Energy-efficient federated learning is an active research direction for edge and IoT environments, particularly around model compression, client selection, communication optimization, and hardware-aware techniques. ([ScienceDirect](#))

78.2 Self-Adaptive Models

Future systems could automatically adjust thresholds according to machine behavior.

78.3 TinyML Digital Twins

Digital twins could simulate machine degradation before deploying models to physical equipment.

78.4 Explainable TinyML

Lightweight explanation techniques could improve maintenance-engineer trust.

78.5 Energy-Aware Scheduling

The system could schedule inference according to battery and workload conditions.

78.6 Continual Learning

Future systems could learn from new operating conditions without requiring complete retraining.

79. Discussion

TinyML provides an important opportunity to move predictive intelligence closer to industrial equipment.

The traditional approach:

Machine→Cloud→DecisionMachine \rightarrow Cloud \rightarrow Decision

can introduce communication and latency dependencies.

The proposed approach:

Machine→TinyML→DecisionMachine \rightarrow TinyML \rightarrow Decision

places initial intelligence at the edge.

This does not mean that cloud infrastructure becomes unnecessary.

Instead, different computational responsibilities can be distributed:

Edge=InferenceEdge=Inference Gateway=CoordinationGateway=Coordination
Cloud=TrainingCloud=Training

This division provides a practical architecture for industrial environments.

A major advantage is event-driven communication.

If machines operate normally for long periods, continuously transmitting raw measurements may be inefficient.

The proposed system instead performs local analysis and communicates when significant events occur.

However, this approach introduces a new challenge: **the edge model must be reliable enough to decide which information is important.**

If the model misses an important anomaly:

FalseNegativeFalseNegative

could delay maintenance.

Therefore, communication reduction must never come at the expense of unacceptable fault-detection performance.

Another challenge is model drift.

Industrial equipment changes with:

- Age
- Maintenance
- Production load
- Environmental conditions
- Component replacement

Consequently:

$$P(X,Y,t_1) \neq P(X,Y,t_2) \Rightarrow P(X,Y,t_1) \neq P(X,Y,t_2)$$

may occur.

Continuous monitoring and periodic retraining are therefore necessary.

Security is equally important.

A TinyML device is physically located near industrial machinery and may be exposed to unauthorized physical access. Protecting the model, firmware, credentials, and sensor inputs is therefore essential.

Finally, TinyML should be evaluated using more than accuracy.

A useful industrial model must balance:

Accuracy+Latency+Memory+Energy+Reliability+Security
Accuracy+ Latency+ Memory+ Energy+ Reliability+ Security

The optimal model is not necessarily the model with the highest accuracy. It is the model that provides an appropriate balance between predictive quality and deployment constraints.

80. Conclusion

The rapid development of Industrial Internet of Things technologies has created new opportunities for continuous equipment monitoring and predictive maintenance. However, cloud-centric machine-learning architectures can introduce communication overhead, network dependency, latency, and increased data exposure.

This paper proposed a conceptual TinyML-based predictive-maintenance framework that places lightweight machine-learning inference directly on resource-constrained industrial edge devices.

The proposed architecture follows:

Sensors→Preprocessing→FeatureExtraction→TinyML→AnomalyDetection→MaintenanceAlert
Sensors \rightarrow Preprocessing \rightarrow FeatureExtraction \rightarrow TinyML \rightarrow AnomalyDetection \rightarrow MaintenanceAlert

Routine sensor information can be processed locally, while important events can be transmitted to gateways and cloud platforms.

Model optimization techniques such as quantization, pruning, feature selection, and knowledge distillation can help reduce the resource requirements of machine-learning models. Current research on TinyML and intelligent edge computing supports the importance of these techniques for deploying machine learning under memory, computation, latency, and energy constraints. ([MDPI](#))

The proposed framework also emphasizes security, model lifecycle management, explainability, and human involvement. A predictive-maintenance system should assist engineers rather than operate as an isolated decision-making mechanism.

The central principle of the framework can be summarized as:

EfficientIndustrialAI=LocalIntelligence+LightweightModels+EventDrivenCommunication+SecureDeployment
EfficientIndustrialAI= LocalIntelligence+ LightweightModels+ EventDrivenCommunication+ SecureDeployment

Future research should experimentally validate the proposed architecture on real industrial hardware and representative machine-condition datasets. Further investigation into federated TinyML, continual learning, explainable edge AI, hardware-aware optimization, energy-aware inference, and secure model updates can strengthen the practical applicability of the approach.

TinyML therefore represents a promising direction for transforming conventional sensor-based industrial monitoring into intelligent, responsive, and resource-efficient predictive-maintenance systems.

References

1. Ray, P. P. (2022). A review on TinyML: State-of-the-art and prospects. *Journal of King Saud University—Computer and Information Sciences*, 34(4), 1595–1623. ([DOI](#))
2. Wang, X., Tang, Z., Guo, J., Meng, T., Wang, C., Wang, T., & Jia, W. (2025). Empowering edge intelligence: A comprehensive survey on on-device AI models. *ACM Computing Surveys*, 57(9), 228. ([DOI](#))
3. Cajas Ordóñez, S. A., Samanta, J., Suárez-Cetrulo, A. L., & Carbajo, R. S. (2025). Intelligent edge computing and machine learning: A survey of optimization and applications. *Future Internet*, 17(9), 417. ([MDPI](#))
4. Alahmari, S., & Alghamdi, I. (2025). A comprehensive survey on energy-efficient and privacy-preserving federated learning for edge intelligence and IoT. *Results in Engineering*, 28, 107849. ([ScienceDirect](#))
5. Moin, A., Challenger, M., Badii, A., & Günemann, S. (2021). Supporting AI engineering on the IoT edge through model-driven TinyML. *arXiv preprint arXiv:2107.02690*. ([arXiv](#))
6. Zila, A. (2026). Edge AI and TinyML for enhancing MAC protocols: A new paradigm for wireless sensor networks in IIoT. *International Journal of Communication Systems*, 39(3), e70403. ([Wiley Online Library](#))

7. Rawat, S. K., & Tuli, N. (2026). Lightweight intelligence—TinyML based energy usage prediction for resource-constrained IoT edge nodes. *International Journal of Innovations in Science, Engineering and Management*, 5(3), 247–253. (ijisem.com)
8. Baliyar Singh, R. K., & Reddy, K. H. K. (2025). Edge-AI empowered cyber-physical systems: A comprehensive review on performance analysis. *Computer Science Review*, 58, 100769. ([ScienceDirect](https://www.sciencedirect.com))
9. Makhdoom, I., Abolhasan, M., Lipman, J., Liu, R. P., & Ni, W. (2019). Anatomy of threats to the Internet of Things. *IEEE Communications Surveys & Tutorials*, 21(2), 1636–1675.
10. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4), 2347–2376.
11. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things: A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7), 1645–1660.
12. Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, 54(15), 2787–2805.
13. Sicari, S., Rizzardi, A., Grieco, L. A., & Coen-Porisini, A. (2015). Security, privacy and trust in Internet of Things: The road ahead. *Computer Networks*, 76, 146–164.
14. Ferrag, M. A., Shu, L., Yang, X., Derhab, A., & Maglaras, L. A. (2020). Security and privacy for the Internet of Things: A review of blockchain solutions. *IEEE Access*, 8, 129953–129975.
15. Priyadarshini, R., & Dash, S. (2023). Machine learning approaches for predictive maintenance in industrial Internet of Things environments. *Journal of Intelligent Manufacturing and Special Equipment*, 4(2), 101–118.