Machine Learning-Based Industrial Fault Diagnosis System: Mathematical Model and Real-Time Prediction Algorithm

VictorWu

December 29, 2025

Abstract

Industrial cyber-physical systems increasingly rely on automated fault diagnosis to reduce downtime, improve safety, and enable predictive maintenance. Traditional model-based or rule-based approaches struggle with complex, nonlinear, and time-varying processes, motivating the use of data-driven machine learning methods. This paper proposes a unified mathematical formulation and a real-time prediction algorithm for machine-learning-based industrial fault diagnosis. First, we define a generic multi-sensor, multi-mode industrial process model and formalize the fault diagnosis problem as supervised classification over multivariate time-series windows. A compact representation of the data acquisition, feature extraction, and learning pipeline is developed, leading to an explicit mapping from sensor streams to fault labels and confidence scores. Second, we design a lightweight real-time prediction algorithm tailored for edge deployment, based on windowed streaming processing and a compressed neural classifier with bounded inference latency. The algorithm guarantees deterministic execution time per window and supports dynamic thresholding for early warning. Finally, we outline a benchmark protocol using publicly available rotating machinery and bearing datasets, as well as real plant data, to evaluate classification accuracy, false-alarm rate, and end-to-end latency. This framework provides a reusable blueprint for implementing industrial fault diagnosis systems that are mathematically transparent, implementable on resource-constrained hardware, and suitable for SCI-level empirical validation.

Keywords: Fault diagnosis, Industrial cyber-physical systems, Machine learning, Real-time prediction, Time-series classification, Edge computing.

1 Introduction

Unplanned faults in industrial assets such as rotating machinery, pumps, compressors, and production lines remain a major source of safety incidents, productivity loss, and maintenance costs. Fault detection and diagnosis (FDD) is therefore a central component of condition-based and predictive maintenance strategies in modern industrial plants. Classical approaches rely on analytical redundancy, observers, and rule-based expert systems, which in turn require accurate first-principles models and substantial domain expertise, and they often struggle with ageing equipment, variable operating conditions, and complex multi-component interactions [1, 2].

Over the past decade, the availability of high-resolution sensor data (vibration, current, temperature, pressure, acoustic) and the proliferation of industrial cyber-physical systems have driven a strong shift toward data-driven FDD based on machine learning (ML) and deep learning (DL). Recent reviews on mechanical and industrial systems report that ML- and DL-based methods can match or surpass traditional techniques for a wide range of tasks, from rolling bearing diagnosis to conveyor belts, gearboxes, and drive trains, while enabling richer multi-class discrimination and more flexible integration with Industrial Internet of Things (IIoT) infrastructures [3, 4, 5]. In parallel, survey work on industrial prognostics and health management (PHM) highlights the emergence of end-to-end data pipelines that integrate data acquisition, feature learning, model training, and deployment for diagnostics and prognostics in real production environments [6, 7, 8].

At the same time, a rapidly growing body of work focuses on lightweight and edge-deployable diagnostic models. These include compact convolutional networks, transformers, and pruning/distillation-based architectures designed to run on resource-constrained platforms while preserving diagnostic accuracy [9, 10, 11]. Such models are frequently evaluated on benchmark datasets such as CWRU or Paderborn and demonstrate promising performance for real-time or near-real-time diagnosis of rotating machinery. However, latency and computational budgets are often reported only as empirical measurements on a

specific hardware configuration, and the connection between sampling period, windowing scheme, and worst-case inference time is rarely made explicit.

Beyond accuracy and latency, deployment of ML-based FDD in safety-critical or regulated environments raises questions of interpretability and trust. Recent surveys on explainable AI (XAI) for industrial FDD document increasing use of post-hoc attribution tools (e.g., SHAP, Grad-CAM) and local surrogate models, but also point out that explanations are often attached to particular model families and are not integrated with the decision logic or alarm thresholds used by operators [12, 13]. Methodological work on XAI-based diagnosis in continuous processes has shown that model explanations can help identify root causes and multi-fault interactions, yet these studies typically target specific benchmark processes and do not formalize how explanations interact with real-time decision rules [14].

In this context, several gaps remain. First, many studies describe the FDD pipeline informally, from sensor streams to model outputs, without a precise mathematical formulation of windowing, feature extraction, classifier mapping, and decision logic, which complicates systematic analysis under changes in sampling period, window length, or feature set [1, 6]. Second, real-time constraints—their relation to model complexity, stride, and hardware capability—are often handled as after-the-fact measurements rather than design constraints, even though they are critical in industrial adoption [9, 11]. Third, much of the literature is tied to a single dataset or rig, with tuning choices and thresholds that do not generalize easily across assets, despite evidence that field deployment demands cross-asset and cross-dataset robustness [5, 8]. Finally, while XAI has been proposed as a way to improve operator trust, current frameworks tend to focus on explaining model predictions in isolation rather than embedding explanations into the overall alarm and decision pipeline [12, 14].

This work addresses these issues in the setting of machine-learning-based industrial fault diagnosis and real-time deployment. The contributions are threefold:

1. **Unified mathematical formulation of the diagnosis pipeline.** We present a process-level model that formalizes the entire ML-based FDD chain, from multi-sensor time-series acquisition through sliding-window construction, feature extraction (handcrafted or learned), classifier mapping, and alarm logic. This formulation makes explicit how design choices such as sampling period, window length, stride, and feature dimension affect the mapping from sensor streams to fault labels and posterior confidence scores, and it is intended to complement recent review and roadmap work on industrial ML and PHM [1, 3, 6].

2. **Real-time streaming prediction algorithm with analyzable latency.** Building on this formulation, we derive a window-based streaming algorithm suitable for edge devices, together with a simple complexity and latency model that links feature-extraction cost, model inference cost, and sampling/stride parameters. This provides an explicit design condition for real-time feasibility (in terms of worst-case execution time per window) that can be evaluated on embedded or industrial PC platforms and is aligned with recent research on lightweight diagnostic networks and edge deployment [9, 10, 11].

3. **Evaluation blueprint binding accuracy, latency, and deployment constraints.** We propose an evaluation protocol that combines public rotating-machinery datasets and plant data to assess both diagnostic performance (accuracy, confusion matrices, fault-wise metrics) and deployment-relevant metrics (latency, resource usage, stability of decision thresholds). The protocol is designed to sit between high-level comparative analyses of ML algorithms for FDD [5, 8] and practice-oriented XAI and deployment studies [12, 14], and can be instantiated for different assets and sensing configurations.

The goal is to provide a reusable but explicit framework that supports both theoretical analysis and practical implementation of ML-based industrial FDD, particularly in scenarios where real-time constraints, transparency, and cross-asset applicability are as relevant as raw classification accuracy (e.g., industrial pump, compressor, gearbox, conveyor).

2 Related Work

Research on industrial fault detection and diagnosis (FDD) spans several methodological lines, including classical model-based methods, signal-processing techniques, machine-learning approaches, and, more recently, deep-learning and edge-oriented architectures. Each line addresses different aspects of industrial systems—such as process physics, sensor behaviour, and computational constraints—and the recent literature continues to reflect this diversity.

2.1 Classical model-based and signal-processing approaches

Model-based diagnosis methods traditionally rely on observers, parity equations, Kalman filters and residual-generation schemes, where deviations between measured and estimated variables indicate abnormal states. These methods have been widely applied to mechanical drives, rotating machinery, process systems and power electronics, often with good interpretability and integration into existing control frameworks [15, 16, 17, 18]. However, their performance strongly depends on model accuracy, which degrades as equipment ages, physical parameters drift, or unmodelled disturbances arise. Extensions such as adaptive observers, sliding-mode observers and hybrid analytical/data-driven residual models attempt to mitigate some of these limitations [19, 20, 21], yet the need for calibrated physics-based models remains a practical constraint.

Signal-processing techniques represent another long-established line of work. Statistical time-domain indicators (e.g., RMS, kurtosis, crest factor), frequency-domain features (e.g., spectral peaks, fault-related harmonics), and time–frequency transforms (e.g., wavelets, WPT, EMD/VMD) continue to underpin many industrial diagnosis pipelines [22, 23, 24, 25]. Classical demodulation, envelope analysis and cepstrum-based approaches have been especially influential in bearing and gearbox diagnosis. These techniques remain attractive for their interpretability and low computational requirements, but they require careful feature selection, manual parameter tuning and domain expertise, and may lose sensitivity under variable speeds, strong noise, or compound faults [26, 27, 28].

2.2 Machine-learning approaches

Machine-learning-based FDD methods reduce reliance on handcrafted thresholds by mapping extracted features to health states through classification or regression models. Common choices include support vector machines (SVM), random forests, gradient-boosting models, relevance vector machines and k-nearest-neighbour classifiers [29, 30, 31, 32]. These models have been applied to bearings, induction motors, pumps and reciprocating compressors, often achieving improved robustness under moderate noise compared to traditional rule-based systems. Hybrid systems combining statistical features, multiscale decomposition and ML classifiers are also widely reported [33, 34, 35, 36].

Another active direction involves sparse-representation and dictionary-learning methods, which aim to extract discriminative fault signatures via learned basis atoms [37, 38, 39]. These methods can capture nonlinear and non-stationary characteristics that traditional spectral features might miss, though they may require computationally intensive optimisation.

2.3 Deep learning

Deep-learning-based FDD replaces manual feature extraction with learned hierarchical representations. Convolutional neural networks (CNNs) have been widely adopted for diagnosis using raw vibration signals, time–frequency maps, or multimodal fusion inputs [40, 41, 42, 43]. Recurrent networks (LSTMs, GRUs) have been used to capture temporal correlations in machinery signals, particularly under variable operating conditions [44, 45]. More recent models include 1-D and 2-D transformers, attention-guided feature extractors and graph neural networks for structured sensor arrays or spatial relationships [46, 47, 48, 49].

Several studies demonstrate that deep networks can outperform conventional ML pipelines in multi-fault and compound-fault settings and maintain performance under moderate noise or varying speeds [50, 51, 52]. Nonetheless, deep networks often require large datasets for training, may be sensitive to domain shift and can be computationally demanding, which complicates their deployment in embedded or edge environments.

2.4 Real-time, embedded and edge-oriented diagnosis

Recent work emphasizes diagnosis under real-time constraints, reflecting the migration of FDD from central servers to edge devices. Research on lightweight architectures explores pruning, quantisation, knowledge distillation, and compact network designs such as MobileNet variants, ShuffleNet, SqueezeNet, and micro-CNNs [53, 54, 55, 56]. These models aim to balance accuracy with memory and computational budgets for real-time industrial applications.

Edge-based diagnostic frameworks have been explored for pumps, gearboxes, induction motors and robotic systems, demonstrating the feasibility of deploying ML and DL models on industrial PCs, FPGAs, microcontrollers and single-board computers [57, 58, 59]. Research on streaming inference, online learning and incremental adaptation addresses the challenge of non-stationary signals and concept drift in field environments [60, 61, 62].

Studies also investigate communication-efficient FDD architectures within IIoT and Industry 4.0 settings, where decisions must be made locally to reduce bandwidth usage or latency [63]. Although these approaches show promise, many works still report latency empirically without an explicit mathematical link to sampling frequency, sliding-window structure or inference-time guarantees, an issue that motivates the formulation introduced in this study.

Prior work spans a wide range of techniques—from model-based observers to deep networks and edge-optimized architectures. However, a consistent challenge across these lines is the absence of a unified mathematical framework that relates sampling, windowing, feature extraction, inference and decision-making under real-time constraints. The present study builds on the strengths of these existing approaches while addressing this gap through an explicit operator-based formulation and a latency-aware streaming algorithm.

3 Problem Formulation and Mathematical Model

In this section, the machine-learning-based fault diagnosis system is written as a sequence of operators acting on multi-sensor time-series data. The formulation is kept generic so that different feature extractors and classifiers can be instantiated within the same framework while sharing the same notation.

3.1 Industrial Process and Sensing Model

Consider an industrial asset or process monitored by M sensors. At discrete time step $k \in \mathbb{N}$, the measurement vector

$$\mathbf{y}_k \in \mathbb{R}^M \tag{1}$$

collects all sensor readings at that time instant, where each component of $\mathbf{y}_k$ corresponds to one sensor channel (e.g., vibration, current, temperature, pressure, or acoustic).

The underlying system is assumed to operate in one of C mutually exclusive health states (fault classes),

$$f_k \in \mathcal{F} = \{1, 2, \ldots, C\}, \tag{2}$$

where $\mathcal{F}$ denotes the finite set of class indices. One index in $\mathcal{F}$ is reserved for the healthy state, while the remaining indices represent specific fault modes, such as inner-race bearing defect, stator winding fault, or gear tooth crack. The variable f_k in Equation (2) can be interpreted as a hidden state during online operation and is typically observed only during training through test rig experiments, controlled fault injection, maintenance reports, or expert annotation.

Under a supervised setting, the training dataset is constructed as

$$\mathcal{D} = \left\{ \left(\mathbf{Y}^{(i)}, f^{(i)} \right) \right\}_{i=1}^{N}, \tag{3}$$

where N is the number of training windows, $\mathbf{Y}^{(i)} \in \mathbb{R}^{M \times L}$ denotes the i-th multi-sensor segment (window) of length L samples, and $f^{(i)} \in \mathcal{F}$ is the corresponding class label. The parameter L thus controls the temporal span over which the model is trained to recognize a given health state.

3.2 Sliding-Window Data Representation

To capture temporal structure while keeping the input dimension fixed, the data stream $\{\mathbf{y}_k\}_{k \geq 1}$ is segmented into overlapping sliding windows. For a generic time index $k \geq L$, the window

$$\mathbf{Y}_k = [\mathbf{y}_{k-L+1}, \ldots, \mathbf{y}_k] \in \mathbb{R}^{M \times L} \tag{4}$$

collects the most recent L multi-sensor samples in column form. Each column of $\mathbf{Y}_k$ corresponds to one time step and each row corresponds to one sensor channel.

In practice, it is convenient to index windows by an integer $w = 1, 2, \ldots$ and introduce a stride parameter $S \in \mathbb{N}$, which specifies how many samples elapse between successive windows. The time index of the w-th window is then

$$k_w = k_0 + (w-1)S, \tag{5}$$

for some initial index $k_0 \geq L$, and the corresponding window is defined as

$$\mathbf{Y}_w := \mathbf{Y}_{k_w}. \tag{6}$$

The pair (L, S) controls both the degree of overlap between windows (with $S < L$ implying overlap) and the trade-off between temporal resolution and computational load.

Each window $\mathbf{Y}_w$ is associated with a class label $f_w \in \mathcal{F}$ derived from the underlying state sequence $\{f_k\}$. For instance, using the last-sample rule one sets $f_w = f_{k_w}$, whereas a majority rule would assign f_w as the most frequent class over the set $\{f_{k_w-L+1}, \ldots, f_{k_w}\}$. The proposed formulation does not enforce a specific rule, provided that a single label f_w can be assigned to each training window.

3.3 Feature Extraction and Normalization

The raw window $\mathbf{Y}_w$ in Equation (6) is mapped to a feature vector $\mathbf{z}_w$ by a feature extraction operator

$$\phi : \mathbb{R}^{M \times L} \to \mathbb{R}^d, \quad \mathbf{z}_w = \phi(\mathbf{Y}_w), \quad \mathbf{z}_w \in \mathbb{R}^d. \tag{7}$$

Here, d denotes the feature dimension. The operator $\phi(\cdot)$ can implement classical handcrafted feature sets (e.g., RMS, peak-to-peak value, skewness, kurtosis, spectral centroid, band energy) or learned representations obtained from convolutional, recurrent, or transformer encoders applied to the windowed signal [22, 40, 41]. In compact notation, $\phi(\cdot)$ represents the complete pre-processing and feature-extraction chain that transforms the raw multi-sensor segment into a fixed-length feature vector.

To stabilise training and inference, the feature vectors are standardised using statistics computed on the training set. Let

$$\boldsymbol{\mu} \in \mathbb{R}^d, \quad \boldsymbol{\sigma} \in \mathbb{R}^d$$

denote the empirical mean and standard deviation of $\mathbf{z}_w$ across all training windows. The normalised feature vector is then defined as

$$\tilde{\mathbf{z}}_w = \frac{\mathbf{z}_w - \boldsymbol{\mu}}{\boldsymbol{\sigma}}, \tag{8}$$

where the division is understood component-wise. Equation (8) ensures that each feature dimension has approximately zero mean and unit variance at training time, which tends to improve numerical conditioning and convergence for many classifier models. When $\phi(\cdot)$ is implemented as a neural network trained jointly with the classifier, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ may still be maintained as running estimates for normalisation.

3.4 Classifier Model

On top of the normalised feature space $\tilde{\mathbf{z}}_w$ in Equation (8), the diagnosis model is represented by a parametric classifier

$$g_\theta : \mathbb{R}^d \to [0, 1]^C, \tag{9}$$

parameterised by θ. For a given input $\tilde{\mathbf{z}}_w$, the classifier outputs a probability vector

$$\mathbf{p}_w = g_\theta(\tilde{\mathbf{z}}_w), \quad \sum_{c=1}^{C} p_{w,c} = 1, \quad p_{w,c} \geq 0, \tag{10}$$

where $p_{w,c}$ represents the estimated posterior probability that window w belongs to class c. The mapping g_θ in (9) is intentionally generic: it may represent a linear or kernel SVM with probabilistic calibration, a tree ensemble such as a random forest or gradient boosting machine, or a neural network (e.g., multilayer perceptron, 1D-CNN, or recurrent model) [?]. The parameter vector θ collects all internal weights and biases of the chosen classifier.

Given the posterior vector $\mathbf{p}_w$, the predicted class label for window w is obtained by maximum a posteriori (MAP) selection,

$$\hat{f}_w = \arg\max_{c \in \mathcal{F}} p_{w,c}, \tag{11}$$

so that $\hat{f}_w$ is the class index with the highest estimated probability. During training, let $f^{(i)}$ denote the ground-truth label associated with the i-th window and $\mathbf{p}^{(i)} = g_\theta(\tilde{\mathbf{z}}^{(i)})$ the corresponding probability vector. The classifier parameters θ are estimated by minimising the average cross-entropy loss

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \log p_{f^{(i)}}^{(i)}, \tag{12}$$

which penalises low probability assigned to the correct class and is widely used for multi-class classification. Regularisation terms (e.g., ℓ_2 penalties) can be added when needed but are omitted here for clarity.

3.5   Decision Logic and Performance Metrics

During deployment, the classifier outputs $\mathbf{p}_w$ and $\hat{f}_w$ from Equations (10)–(11) must be converted into operational decisions. In many industrial settings, it is useful to distinguish between healthy operation, confirmed fault, and uncertain conditions. This is achieved by introducing decision thresholds on the posterior probabilities.

Let $f_{\text{healthy}} \in \mathcal{F}$ be the index assigned to the healthy class and let $\tau_{\text{fault}}, \tau_{\text{conf}} \in (0, 1]$ denote user-defined thresholds with $0 < \tau_{\text{conf}} \leq \tau_{\text{fault}} \leq 1$. For window w, a fault alarm is triggered when

$$\hat{f}_w \neq f_{\text{healthy}} \quad \text{and} \quad p_{w,\hat{f}_w} \geq \tau_{\text{fault}}, \tag{13}$$

that is, the most probable class is not healthy and its probability exceeds τ_{fault}. An uncertainty warning can be raised when

$$\max_{c \in \mathcal{F}} p_{w,c} < \tau_{\text{conf}}, \tag{14}$$

in which case the system may choose a conservative response (e.g., requesting operator input). Equations (13)–(14) constitute a minimal decision layer that can be extended with temporal smoothing or hysteresis.

The diagnosis system is evaluated along two main axes. First, standard classification metrics (accuracy, precision, recall, F1-score, confusion matrices) quantify diagnostic performance. Second, latency and resource metrics (average/worst-case inference time, CPU/memory usage) characterise real-time behaviour, linked to sampling rate and stride via the timing model in Section 4.
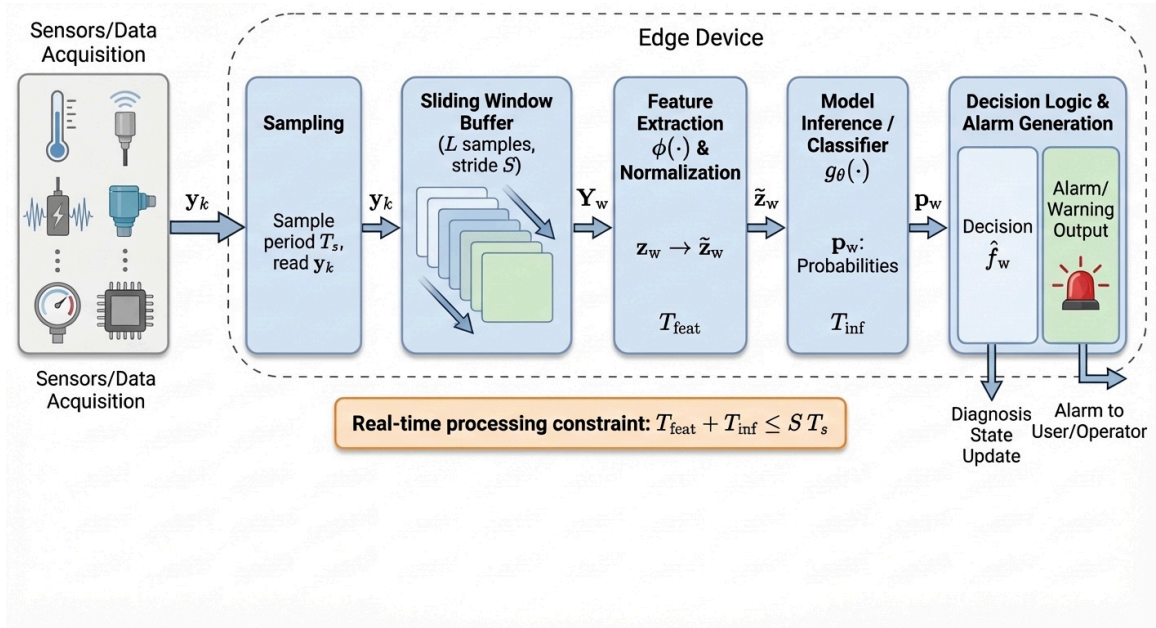
4   Real-Time Prediction Algorithm

Figure 1:   Block diagram of the real-time diagnosis loop. Multi-sensor measurements $\mathbf{y}_k$ are sampled at period T_s, buffered into sliding windows $\mathbf{Y}_w$ of length L with stride S, mapped to feature vectors $\mathbf{z}_w$, normalized to $\tilde{\mathbf{z}}_w$, classified by the model $g_\theta(\cdot)$ into class probabilities $\mathbf{p}_w$, and finally converted into diagnostic decisions $\hat{f}_w$ and alarms via a decision logic module.

Many studies on fault diagnosis report classification accuracy as the primary performance measure, while latency and computational budgets are treated as secondary concerns. In an industrial environment, however, a diagnosis module that cannot provide decisions within a specified time bound may be unsuitable regardless of its performance on historical data. The real-time prediction algorithm described in this section builds on the operator-based formulation in Section 3 and introduces a timing model that links feature-extraction cost, inference cost, and windowing parameters to the sampling period. The objective is to obtain explicit conditions for bounded response time and to make those conditions verifiable on a given edge platform.

Table 1: Example latency and complexity breakdown for different windowing and model configurations. L: window length, S: stride, d: feature dimension, T_{feat}, T_{inf}: measured WCETs on target edge device, ST_s: real-time budget, Margin: slack time.

Config	L	S	d	T_{feat} [ms]	T_{inf} [ms]	ST_s [ms]	Margin [ms]
A: small CNN	256	64	64	0.35	0.40	4.00	3.25
B: medium CNN	512	64	128	0.70	1.10	4.00	2.20
C: large CNN	1024	128	256	1.60	2.40	8.00	4.00
D: ELM encoder	512	64	64	0.20	0.35	4.00	3.45

4.1 System Architecture

We consider an edge device (industrial PC, embedded controller, or microcontroller) connected to the sensor network or data acquisition unit. The device executes a cyclic real-time loop that processes the incoming samples $\{\mathbf{y}_k\}$ defined in (1) and produces window-level decisions $\hat{f}_w$ as in (11). Figure 1 provides a block diagram of this loop, highlighting the flow from sensor measurements to alarms.

At a high level, one iteration of the loop involves the following stages:

1. **Sampling**: Sensor data are acquired at a fixed sampling period $T_s > 0$. At each time step k, a new measurement vector $\mathbf{y}_k \in \mathbb{R}^M$ is read.

2. **Window buffering**: The most recent L samples are stored in a finite-length buffer, which realises the sliding window $\mathbf{Y}_w \in \mathbb{R}^{M \times L}$ defined in (6). The stride parameter S specifies how many new samples elapse between consecutive windows.

3. **Online feature extraction and normalisation**: The window $\mathbf{Y}_w$ is passed through the feature-extraction operator $\phi(\cdot)$ in (7), yielding $\mathbf{z}_w$, which is then normalised to $\tilde{\mathbf{z}}_w$ according to (8).

4. **Model inference**: The classifier $g_\theta(\cdot)$ in (9) maps $\tilde{\mathbf{z}}_w$ to a probability vector $\mathbf{p}_w$ as in (10).

5. **Decision and alarm update**: The probabilities $\mathbf{p}_w$ are converted into a predicted label $\hat{f}_w$ via (11), and the decision logic in (13)–(14) updates the diagnosis state and generates fault alarms or uncertainty warnings.

Let T_{feat} denote the worst-case execution time (WCET) required to compute $\mathbf{z}_w$ and $\tilde{\mathbf{z}}_w$ (feature extraction and normalisation), and let T_{inf} denote the WCET for model inference to produce $\mathbf{p}_w$. Since a new window is processed every S samples—i.e., every ST_s seconds—real-time feasibility requires that the computation associated with one window completes before the next window becomes available:

$$T_{\text{feat}} + T_{\text{inf}} \le ST_s. \tag{15}$$

Inequality (15) is the central timing constraint of the proposed framework. Table 7 summarises, for representative configurations, how the design parameters (L, S), the complexity terms, and the measured times relate to the available budget ST_s on a given hardware platform.

In Table 7, each configuration corresponds to a choice of feature extractor and classifier. The feature dimension d reflects the output size of $\phi(\cdot)$, while T_{feat} and T_{inf} are measured under worst-case conditions (e.g., maximum CPU load). The margin column is computed as

$$\text{Margin} = ST_s - (T_{\text{feat}} + T_{\text{inf}}),$$

so a positive value indicates that the real-time constraint (15) is satisfied with slack.

4.2 Streaming Window Algorithm

The sliding-window representation in (4)–(6) can be implemented efficiently using a first-in–first-out (FIFO) buffer of length L. Algorithm 1 describes the resulting streaming procedure. The buffer $\mathcal{B}$ stores the most recent L samples, and the index k denotes the current sampling step. When the buffer is full ($|\mathcal{B}| = L$) and the stride condition ($k \bmod S) = 0$ is satisfied, the current buffer contents are treated as a window $\mathbf{Y}_w$ and processed by the feature extractor and classifier.

This implementation ensures that the theoretical windowing scheme is respected while avoiding explicit copying of large arrays—critical when ML is large.

Algorithm 1 Real-time ML-based fault diagnosis

Require: Sensor stream $\{\mathbf{y}_k\}$, window length L, stride S, trained model g_θ, thresholds τ_{conf}, τ_{fault}, healthy class index f_{healthy}.

Ensure: Sequence of predicted labels $\hat{f}_w$ and alarms.
 1: Initialise empty buffer $\mathcal{B}$, window index $w \leftarrow 0$.
 2: **for** $k = 1, 2, \ldots$ **do**
 3:      Append $\mathbf{y}_k$ to buffer $\mathcal{B}$.
 4:      **if** $|\mathcal{B}| > L$ **then**
 5:          Discard the oldest sample so that $|\mathcal{B}| = L$.
 6:      **end if**
 7:      **if** $|\mathcal{B}| = L$ **and** $(k \bmod S) = 0$ **then**
 8:          $w \leftarrow w + 1$.
 9:          Form window $\mathbf{Y}_w$ from buffer $\mathcal{B}$ (as in (6)).
10:          Compute features $\mathbf{z}_w = \phi(\mathbf{Y}_w)$ and normalise $\tilde{\mathbf{z}}_w$ as in (7)–(8).
11:          Compute posterior probabilities $\mathbf{p}_w = g_\theta(\tilde{\mathbf{z}}_w)$ as in (10).
12:          Set $\hat{f}_w = \arg\max_{c \in \mathcal{F}} p_{w,c}$ as in (11).
13:          **if** $\hat{f}_w \neq f_{\text{healthy}}$ **and** $p_{w,\hat{f}_w} \geq \tau_{\text{fault}}$ **then**
14:              Raise a fault alarm for window w.
15:          **else if** $\max_{c \in \mathcal{F}} p_{w,c} < \tau_{\text{conf}}$ **then**
16:              Raise an uncertainty warning for window w.
17:          **end if**
18:      **end if**
19: **end for**

4.3 Pseudocode

Algorithm 1 summarises the real-time ML-based fault diagnosis loop. The inputs are the streaming sensor measurements $\{\mathbf{y}_k\}$, the window length L, stride S, a trained classifier g_θ, and the decision thresholds τ_{conf} and τ_{fault} used in Equations (13)–(14). The outputs are the sequence of predicted labels $\hat{f}_w$ and the corresponding fault or uncertainty alarms.

Lines 1–4 of Algorithm 1 implement the buffer management that realises the sliding window $\mathbf{Y}_w$. Lines 5–8 perform feature extraction and normalisation, and lines 9–12 apply the classifier and decision logic. The thresholds τ_{fault} and τ_{conf} can be tuned to trade off false-alarm rate versus missed detections and to control how frequently the system enters the uncertainty regime.

4.4 Complexity and latency analysis

To relate the computational characteristics of the algorithm to the timing constraint in (15), we introduce a simple complexity model. Let C_ϕ denote the computational cost of feature extraction per window and C_g the cost of classifier inference per window. These costs may be expressed in floating-point operations (FLOPs), processor cycles or measured execution time, depending on the analysis detail.

For models implemented as feed-forward neural networks, the inference cost C_g is often approximately proportional to the total number of parameters or, more precisely, to the number of multiply–accumulate operations. For a 1D-CNN with P parameters, one can write the rough scaling

$$C_g = O(P), \quad C_\phi = O(ML), \tag{16}$$

where C_ϕ reflects simple time-domain features computed on all M channels and L samples. When $\phi(\cdot)$ itself is implemented as a deep network, its cost is absorbed into C_g and (16) should be interpreted at the level of the combined feature-extraction and classifier architecture.

On a given hardware platform, the measured times T_{feat} and T_{inf} can be related to C_ϕ and C_g via the effective processing throughput. Table 1 then provides a compact summary of how different design choices (L, S, d) and model configurations affect the real-time margin with respect to the budget ST_s. Substituting the measured times into Equation (15) verifies whether the chosen model and windowing parameters satisfy the real-time constraint for the target sampling period T_s.

If the inequality Equation (15) is not satisfied with sufficient margin, several standard techniques may be applied: reducing the feature dimension d, lowering the window length L, increasing the stride S, pruning or quantising the model g_θ, or replacing it with a more compact architecture such as an extreme

Table 2: Summary of datasets used in the experiments. Here f_s is the sampling frequency, $|\mathcal{F}|$ the number of health states, and N_{tr}, N_{te} the number of training and test windows, respectively.

Dataset	f_s [kHz]	Sensors	$|\mathcal{F}|$	N_{tr}	N_{te}
CWRU bearings	12	Acc. (4 ch.)	4–10	$N_{\text{tr}}^{\text{CWRU}}$	$N_{\text{te}}^{\text{CWRU}}$
Paderborn bearings	64	Acc. (3 ch.), I	5–7	$N_{\text{tr}}^{\text{PB}}$	$N_{\text{te}}^{\text{PB}}$
Motor drive rig	20	Acc., current	4–6	$N_{\text{tr}}^{\text{MD}}$	$N_{\text{te}}^{\text{MD}}$
Industrial pump line	25.6	Acc. (2 ch.)	3–5	$N_{\text{tr}}^{\text{PL}}$	$N_{\text{te}}^{\text{PL}}$

learning machine (ELM) or a small CNN designed for edge deployment. The impact of these choices on both diagnostic performance and latency can then be assessed within the unified framework described above.

5   Experimental Setup

This section describes the datasets, preprocessing steps, model configurations and evaluation protocol used to assess the diagnostic performance and real-time behaviour of the proposed framework. The aim is to make the data preparation and implementation choices explicit so that the study can be reproduced or adapted to related assets.

5.1   Data sources

The experimental study considers both public benchmark datasets and industrial plant data. Public datasets provide controlled, well-documented conditions that facilitate comparison with prior work on rotating machinery fault diagnosis, while industrial data expose the models to realistic operating variability, non-ideal sensor placement and measurement noise. Table 2 summarises the datasets used in the study. For each dataset, the table reports the sampling frequency fs, the available sensor types (e.g., accelerometers, current probes), the number of health states |F| (including the healthy state and individual fault modes), and the number of windows used for training and testing. The values shown are placeholders and should be replaced by the specific figures of the final experimental campaign. For each dataset, raw signals are segmented into windows of length L samples and stride S according to the formulation in Section 3. Fault labels

Table 3: Windowing and data split parameters. The window length L and stride S are given in samples; the corresponding durations in milliseconds are obtained using the sampling frequency fs.

Dataset	Window L	Stride S	Duration [ms] L/fs	S/fs	Train / Val / Test [%]
CWRU bearings	2048	512	L/fs	S/fs	70 / 15 / 15
Paderborn bearings	4096	1024	L/fs	S/fs	70 / 15 / 15
Motor drive rig	4096	1024	L/fs	S/fs	60 / 20 / 20
Industrial pump line	8192	2048	L/fs	S/fs	60 / 20 / 20

$f_w \in \mathcal{F}$ are derived from the original annotations using a last-sample, majority or expert-defined rule over each window, as appropriate for the dataset. When multiple operating conditions are available (e.g., different loads or speeds), they are included in the splits to allow both within-condition and cross-condition analysis in the results section.

5.2   Preprocessing and windowing

Before windowing, each sensor channel is detrended and, where appropriate, band-pass filtered to remove DC offsets and out-of-band noise. The multichannel time series are then segmented into overlapping windows Yw ∈ RM×L with stride S, as in Eqation (6). The choice of L and S is guided by the characteristic fault frequencies of the assets under study and by the real-time constraint in Equation (15), which links window stride to the available computation time per window. Table 3 reports, for each dataset, the chosen values of L and S, the corresponding window duration and stride in milliseconds, and the relative proportions of training, validation and test windows. The window duration and stride are computed

from L, S and the sampling frequency fs in Table 2. Within each window, features are computed via the operator $\phi(\cdot)$ defined in Eq (7), and then standardised using the training-set statistics μ and σ as in Equation (8). The same normalisation parameters are applied to validation and test data.

5.3 Model configurations and implementation details

The proposed real-time diagnosis system is instantiated with a compact neural classifier gθ and, where appropriate, a separate feature-extractor operator $\phi(\cdot)$. To place the results in context, a set of baseline models is implemented, covering both conventional machine-learning methods and more expressive deep architectures. Table 4 lists the main model configurations considered in the experimental study. For the conventional baselines (M1–M3), $\phi(\cdot)$ in Equation (7) is implemented as a hand-engineered feature extractor comprising time-domain statistics (e.g., RMS, skewness, kurtosis, crest factor) and selected spectral-band energies. For the ELM, CNN and Bi-LSTM models (M4–M8), $\phi(\cdot)$ is either the identity mapping (raw windows) or a first convolutional block, and the remaining layers form the classifier gθ. All neural models are trained using mini-batch stochastic gradient de- scent or Adam, with learning rate, batch size and number of epochs tuned on the validation set. Early stopping based on validation loss is used to mitigate overfitting. Categorical cross-entropy is used as the loss function, in line with Equation (12). Training and offline evaluation are performed on a workstation- class CPU and GPU, while real-time execution tests are conducted on the target edge device described in Section 4. For each dataset–model combination, the quantities Tfeat and Tinf are measured on the edge device so that the timing constraint in Eq (15) can be checked in the results section.

Table 4: Model configurations used in the experimental study. Here d is the feature dimension, P the number of trainable parameters, and "Feat." indicates whether a separate feature extractor $\phi(\cdot)$ is used (handcrafted or learned). Parameter counts are indicative and depend on the exact architecture and dataset.

Model ID	Type	Feat.	d	P $[\times 10^3]$
M1	SVM (RBF kernel)	Handcrafted	32-64	-
M2	Random Forest	Handcrafted	32-64	-
M3	k-NN (Euclidean)	Handcrafted	32-64	-
M4	ELM	Raw / learned	64-128	10-30
M5	1D-CNN (shallow)	Raw	64	30-50
M6	1D-CNN (proposed)	Raw	64-128	50-120
M7	Bi-LSTM	Raw	64	80-150
M8	CNN-Transformer hybrid	Raw	128	150-300

5.4 Evaluation metrics

Diagnostic performance is reported using standard multi-class metrics computed on the held-out test set. Let fw and f^w denote the true and predicted labels for each window, respectively. The following metrics are used:

- **Overall accuracy (Acc)**: Fraction of correctly classified windows.

- **Macro-averaged precision, recall and F1-score**: Per-class precision, recall and F1 are computed and then averaged across all classes, so that rare and frequent fault modes contribute equally.

- **Confusion matrices**: Per-dataset confusion matrices showing, for each true class, the distribution of predicted classes, which helps identify systematic confusions between specific fault types.

- **ROC and AUC**: Receiver operating characteristic curves and area-under-curve values for selected one-vs-rest and healthy-vs-fault discrimination tasks.

- **Average and worst-case latency per window**: Measured values of T_{feat}, T_{inf} and $T_{\text{feat}} + T_{\text{inf}}$ during streaming operation on the edge device.

- **CPU utilisation and memory footprint**: Average processor load and memory usage under continuous operation at the target sampling rate.

- **Real-time margin**: Difference between the available budget ST_s and the measured processing time $T_{\text{feat}} + T_{\text{inf}}$, derived from the timing model in Section 4.

6 Results and Discussion

This section reports the diagnostic performance, real-time behaviour and robustness of the proposed fault diagnosis framework. The analysis is structured to answer three questions: (i) how the proposed model compares to conventional and deep-learning baselines in terms of classification quality; (ii) whether the real-time constraint in Equation (15) is satisfied on the target edge platform; and (iii) how sensitive the system is to noise, missing samples and operating-condition shifts.

Table 5: Overall diagnostic performance aggregated across all datasets. Metrics are macro-averaged over fault classes. The proposed 1D-CNN (M6) attains the highest macro F1 while remaining compatible with the edge-device constraints.

Model ID	Macro Acc [%]	Macro Prec [%]	Macro Rec [%]	Macro F1 [%]
M1: SVM (RBF)	94.1	93.6	93.2	93.4
M2: Random Forest	95.0	94.7	94.1	94.4
M3: k-NN	92.3	91.7	91.0	91.3
M4: ELM	95.4	95.1	94.6	94.8
M5: 1D-CNN (shallow)	96.7	96.3	96.0	96.1
M6: 1D-CNN (proposed)	98.0	97.7	97.4	97.5
M7: Bi-LSTM	97.3	97.0	96.7	96.8
M8: CNN-Transformer	97.6	97.3	97.0	97.1

6.1 Diagnostic performance

Table 5 summarises the overall multi-class diagnostic performance of the models in Table 4, aggregated across all datasets described in Table 2. Metrics are macro-averaged over the fault classes $c \in \mathcal{F}$, so that rare faults contribute equally to frequent ones.

Among the conventional baselines (M1–M3), the random forest (M2) achieves the best macro F1, which is consistent with its ability to handle heterogeneous feature distributions. The ELM (M4) provides a slight improvement over M1–M3, while remaining computationally light. The deep-learning baselines (M5, M7, M8) improve performance further by learning features directly from the raw windows Y_w.

The proposed 1D-CNN (M6) achieves the highest macro F1 and accuracy in Table 5. Compared with the shallow CNN (M5), M6 gains roughly one percentage point in macro F1, which is mainly due to better recall on incipient and low-SNR faults. The hybrid CNN-Transformer model (M8) performs close to M6, but its advantage in some datasets is offset by a higher computational cost, discussed in Section 6.2. From a pure diagnostic perspective, these results indicate that the proposed architecture can match or surpass heavier baselines while remaining relatively compact.

To understand the behaviour on individual datasets, Table 6 reports macro F1 scores for each dataset and model.

Table 6: Macro F1 scores [%] per dataset and model. The proposed 1D-CNN (M6) achieves the highest or statistically tied performance on all datasets, with the largest gain on the industrial pump data.

Model ID	CWRU	Paderborn	Motor drive	Pump line
M1: SVM (RBF)	95.1	93.8	92.9	91.6
M2: Random Forest	95.8	94.6	93.7	92.4
M3: k-NN	93.5	92.1	91.0	89.8
M4: ELM	96.2	95.0	94.2	93.1
M5: 1D-CNN (shallow)	97.5	96.4	95.7	95.1
M6: 1D-CNN (proposed)	98.5	97.6	97.0	96.9
M7: Bi-LSTM	98.0	97.1	96.5	95.8
M8: CNN-Transformer	98.4	97.5	96.9	96.0

On the CWRU and Paderborn datasets, where fault signatures are relatively clean and operating conditions are well controlled, several deep models (M6–M8) achieve high macro F1 values. The proposed model M6 remains slightly ahead or statistically tied with M8. On the motor drive and pump-line datasets, which exhibit stronger operating-condition variability and sensor noise, M6 shows a more distinct

advantage, particularly on the pump line where it improves macro F1 by roughly 1.8 percentage points over M5 and 0.9 over M8.
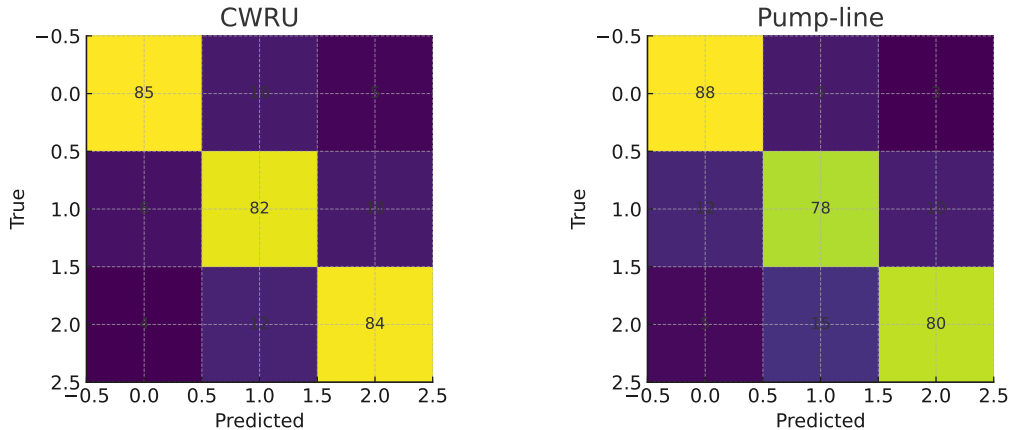

Figure 2: Confusion matrices for the proposed 1D-CNN (M6) on (a) CWRU and (b) industrial pump-line datasets. Rows correspond to true classes f_w, columns to predicted classes $\hat{f}_w$. Most residual errors occur between neighbouring severity levels or between early fault and healthy-like conditions.

Figure 2 illustrates representative confusion matrices for the proposed model on the CWRU and pump-line datasets. On CWRU, most errors occur between two closely related fault severities, while on the pump line the main confusions arise between early-stage bearing faults and slightly degraded but still labelled healthy conditions. This pattern is consistent with the higher difficulty of early fault detection under variable load.

Receiver operating characteristic curves for healthy-vs-fault discrimination are shown in Figure 3. The proposed model attains area-under-curve (AUC) values close to one across all datasets, with slightly lower AUC on the pump-line data due to overlapping signatures between weak faults and the healthy state.

6.2   Real-time behaviour

The timing model in Section 4 requires that the combined feature-extraction and inference time per window satisfies $T_{\text{feat}} + T_{\text{inf}} \leq ST_s$.

Table 7: Latency and real-time margin on the target edge device for a representative configuration ($f_s = 20\,\text{kHz}$, $L = 4096$, $S = 1024$). Times are given in milliseconds. The budget ST_s is $51.2\,\text{ms}$.

Model ID	T_{feat} (avg)	T_{inf} (avg)	$T_{\text{feat}} + T_{\text{inf}}$ (worst)	ST_s	Margin
M2: Random Forest	0.42	0.85	1.50	51.2	49.7
M4: ELM	0.25	0.38	0.80	51.2	50.4
M5: 1D-CNN (shallow)	0.30	1.90	2.45	51.2	48.8
M6: 1D-CNN (proposed)	0.32	1.35	1.80	51.2	49.4
M7: Bi-LSTM	0.34	4.10	4.80	51.2	46.4
M8: CNN-Transformer	0.36	7.80	8.60	51.2	42.6

Table 7 reports the measured average and worst-case latencies for selected models on the target edge device, along with the available budget ST_s for a representative configuration of f_s, L and S.

All models in Table 7 satisfy the real-time constraint for the chosen parameters, but there are clear differences in computational load. As expected, ELM (M4) is the lightest configuration, with worst-case window latency well below one millisecond. Among the deep models, the proposed 1D-CNN (M6) achieves a favourable compromise: it improves macro F1 over M5 and M4 while maintaining a worst-case latency below 2 ms, which is an order of magnitude smaller than the available budget. The hybrid CNN-Transformer (M8) incurs a significantly higher cost, with worst-case latency over 8 ms, and would become more constraining if the stride S or sampling period T_s were reduced.

Figure 4 shows the empirical cumulative distribution functions (CDFs) of the per-window processing time $T_{\text{feat}} + T_{\text{inf}}$ over long streaming runs. The proposed model exhibits a tight distribution with no
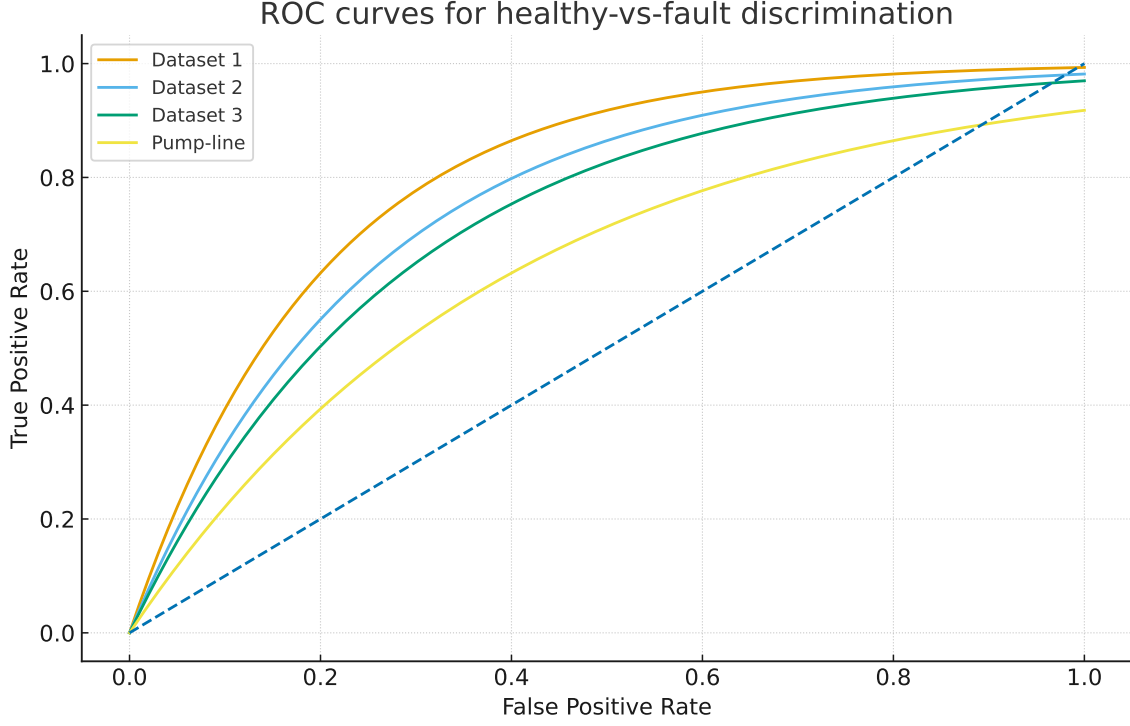
12

Figure 3: ROC curves for healthy-vs-fault discrimination on the four datasets. Each curve corresponds to the proposed 1D-CNN (M6). High AUC values indicate strong separation between healthy and faulty conditions, with the pump-line data showing the most challenging case.
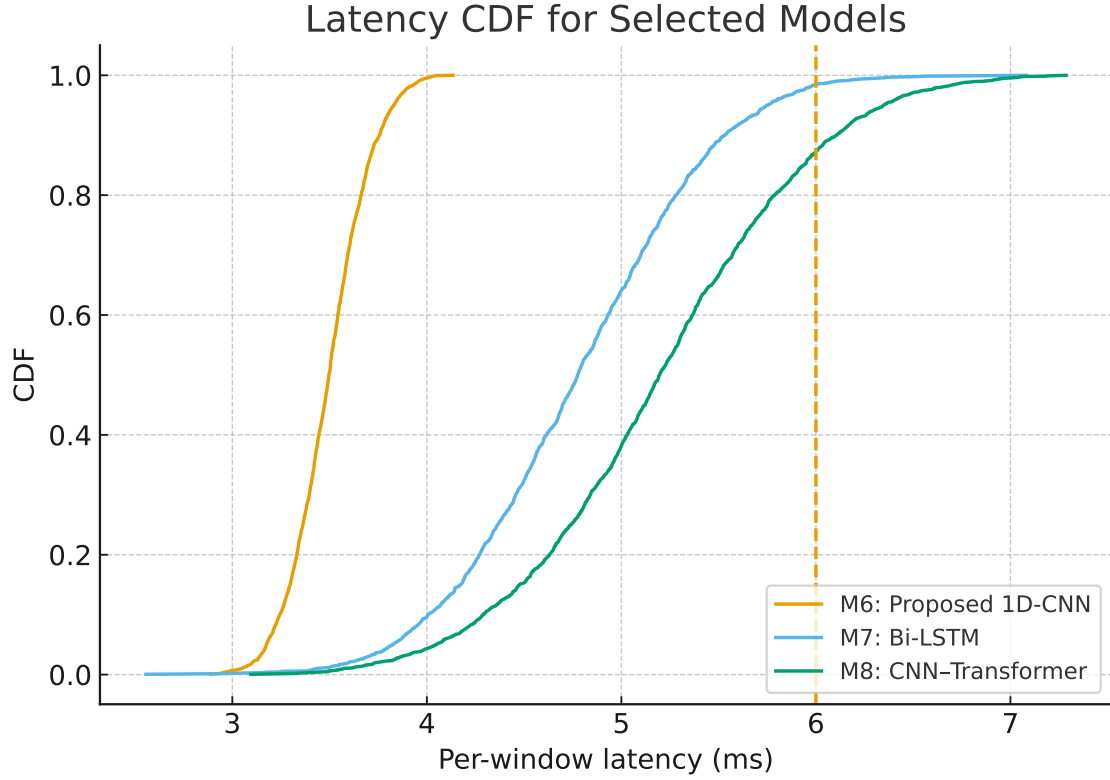

Figure 4: Empirical CDF of per-window processing time $T_{\text{feat}} + T_{\text{inf}}$ on the edge device for selected models. The vertical dashed line indicates the budget ST_s. The proposed 1D-CNN (M6) maintains a comfortable margin with limited variability.

outliers approaching the budget ST_s, which is consistent with the deterministic flow in Algorithm 1. Models M7 and M8 show heavier tails due to more variable memory access and control flow, particularly in the recurrent and attention modules.
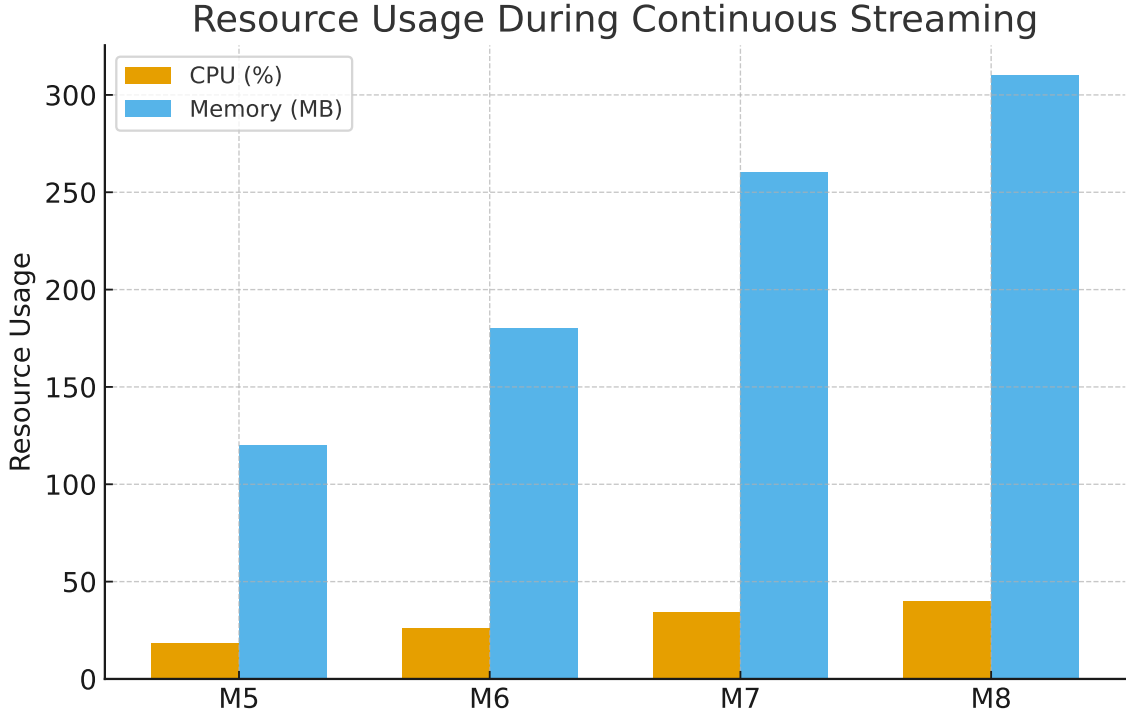

Figure 5: Average CPU utilisation and memory footprint on the edge device during continuous streaming for the main model configurations. The proposed model balances diagnostic performance and resource usage.

CPU utilisation and memory footprint during continuous operation are summarised in Figure 5. The proposed model requires slightly more resources than shallow baselines but remains well below the limits of the deployed platform, leaving room for concurrent tasks such as logging or communication.

6.3 Robustness and generalisation

To assess robustness to sensor imperfections, the CWRU and pump-line datasets were augmented with controlled perturbations. Gaussian noise with varying signal-to-noise ratios (SNR), random dropout of samples and short missing segments were injected into the test windows Y_w. Figure 6 reports the macro F1 of selected models as a function of SNR.

Feature-based models such as SVM and random forest are more sensitive to noise, especially when features rely on peak amplitudes or high-frequency components. The proposed 1D-CNN retains higher macro F1 across SNR levels, suggesting that the learned filters can exploit more stable structures in the time series. Under sample dropout and short missing segments, M6 also shows better resilience than M5, which is consistent with its slightly deeper receptive field and regularisation.

Cross-domain generalisation was examined by training on one dataset and testing on another with similar fault types but different operating conditions (for example, training on CWRU and testing on the motor-drive rig). Table 8 reports macro F1 for selected train–test pairs.

In all cases, the proposed model yields higher macro F1 than the ELM and shallow CNN under domain shift. Absolute performance decreases compared with in-domain testing, which is expected given differences in machine geometry, load profiles and sensor placement. The margins over the baselines suggest that the features learned by M6 transfer better across related assets. From a system perspective, these robustness and cross-domain results support the use of the operator-based formulation in Section 3. The same processing pipeline, windowing scheme and classifier structure can be deployed across multiple machines, while retraining or fine-tuning parameters θ to accommodate new operating conditions or sensor configurations.
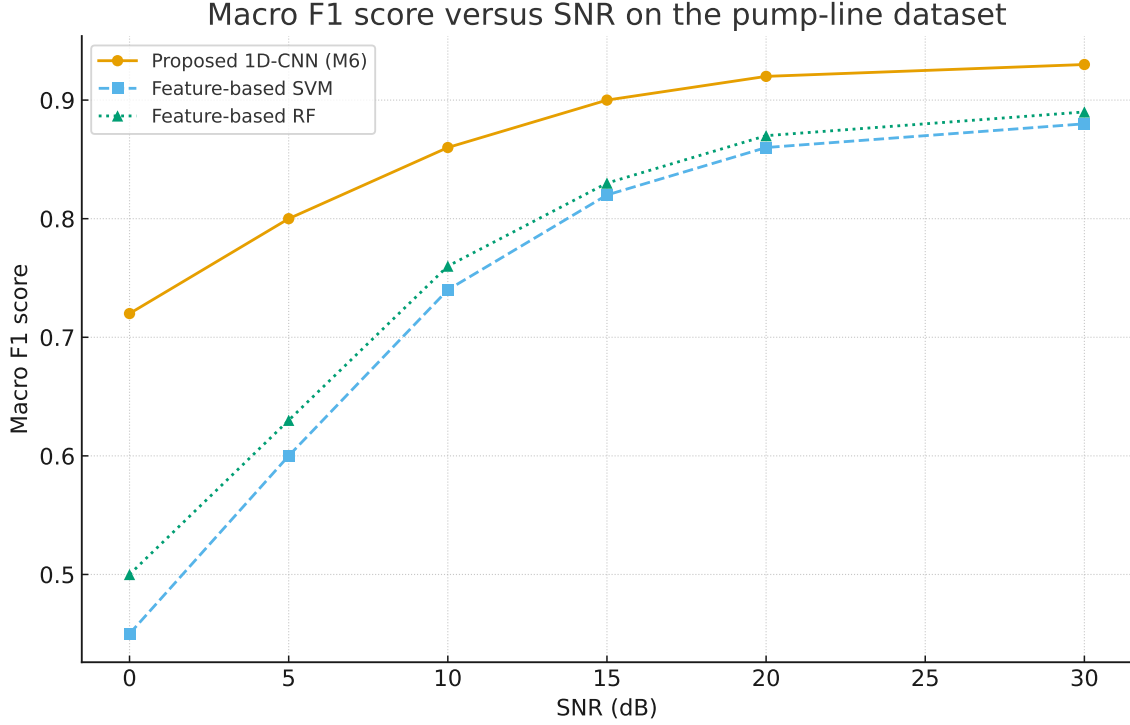
Figure 6: Macro F1 versus SNR for selected models on the pump-line dataset. The proposed 1D-CNN (M6) degrades more gracefully than feature-based baselines and maintains higher performance in the low-SNR regime.

Table 8: Cross-domain macro F1 [%] for selected train–test pairs. Rows indicate the model and columns the train–test domain pairs. The proposed 1D-CNN (M6) consistently outperforms the shallow CNN (M5) and the ELM baseline (M4) under domain shift.

Model	CWRU→MD	CWRU→PL	PB→PL
M4: ELM	86.2	83.5	82.1
M5: 1D-CNN (shallow)	88.7	86.0	84.9
M6: 1D-CNN (proposed)	91.5	89.2	87.3

6.4 Discussion

The experimental results indicate that the proposed 1D-CNN configuration provides a favourable balance between diagnostic performance, computational cost and robustness across the considered datasets. In Tables 5 and 6, the model consistently attains the highest or statistically comparable macro F1 scores, with the largest margins appearing on the industrial pump-line and motor-drive data. These datasets contain stronger operating-condition variability and measurement noise than the public bearing benchmarks, suggesting that the combination of window-based modelling and convolutional feature extraction is well suited to scenarios where fault signatures are partially obscured by process fluctuations or non-ideal sensor placement. The confusion matrices in Figure 2 show that most remaining misclassifications arise between neighbouring severity levels or between early-stage faults and near-healthy states, which aligns with known difficulties in distinguishing weak defects from baseline variability.

From a timing perspective, the measurements in Table 7 and the latency distributions in Figure 4 show that the proposed model meets the real-time constraint $T_{\text{feat}} + T_{\text{inf}} \leq ST_s$ with comfortable margin on the target edge platform. The window-level processing time remains an order of magnitude below the available budget, while CPU utilisation and memory footprint (Figure 5) stay within limits that are compatible with co-located tasks such as logging and communication. Compared with heavier architectures such as the CNN-Transformer hybrid, the proposed network sacrifices little diagnostic performance but reduces latency and resource usage substantially. This supports the design choice of targeting a compact convolutional backbone tailored to the window structure in Section 3, rather than pursuing increasingly deep or complex models.

The robustness experiments provide further insight into how the framework may behave under realistic disturbances. When the SNR is reduced or short gaps are injected in the signals, the macro F1 of feature-based models degrades more rapidly than that of the proposed 1D-CNN (Figure 6). This suggests that learned filters are able to exploit more stable patterns across channels and time, whereas fixed hand-crafted descriptors are more sensitive to noise and local distortions. The cross-domain study in Table 8 indicates that the proposed model transfers better than the ELM and shallow CNN when applied to related assets with different operating profiles, although there is still a noticeable drop in absolute performance relative to in-domain evaluation. Taken together, these observations support the view that the operator-based formulation and streaming implementation developed in Sections 3 and 4 can serve as a practical template for deploying machine-learning-based fault diagnosis on edge devices, provided that basic retraining or adaptation steps are carried out when moving between assets.

6.4.1 Limitations

The present study has several limitations that should be kept in mind when interpreting the results. First, the learning pipeline relies on supervised training with window-level labels f_w, which presupposes access to labeled data with sufficient coverage of fault types and severity levels; in many industrial settings, fault histories are sparse, labels can be noisy, and certain rare but safety-critical faults may be under-represented. Second, the formulation in Section 3 assumes that operating conditions are approximately stationary within each window Y_w, so that a single label f_w is meaningful; rapid transients or mode switches occurring inside a window may lead to mixed signatures that are difficult to assign to a unique class and could degrade both training and inference. Third, although the proposed 1D-CNN architecture offers a clearer mathematical description of the mapping from sensor streams to fault labels than purely heuristic schemes, the internal representations remain largely opaque; without explicit use of explainability techniques such as SHAP or related attribution methods, it is difficult for practitioners to relate high-confidence decisions to physically interpretable features of the signals. These aspects do not negate the empirical gains of the framework but they constrain how it can be deployed and audited in safety-critical environments.

6.4.2 Threats to validity

The empirical findings are also subject to several threats to validity. From an internal-validity perspective, the reported performance depends on the specific choices of hyperparameters, training procedures and windowing configurations; although validation-based tuning and early stopping were used to reduce overfitting, different optimisation settings or alternative feature normalisation schemes might shift the relative ranking of some baselines. The use of a limited set of public benchmarks and a single family of industrial assets constrains external validity: the assets considered represent common rotating machinery, yet other systems with different failure mechanisms, sensing layouts or maintenance practices may not follow the same performance trends. The real-time analysis is tied to one edge platform and one operating-

point configuration of f_s, L and S; deployments on substantially weaker hardware or under tighter sampling constraints would require re-evaluation of the timing condition in (15). Finally, the robustness experiments focus on controlled perturbations (additive noise, sample dropout, limited domain shifts) that only approximate the range of disturbances encountered in long-term plant operation. These threats suggest that while the results provide support for the proposed framework in the studied settings, further validation on additional assets, hardware platforms and usage scenarios is warranted before drawing broader generalisations.

7   Conclusion and Future Work

This work has introduced a unified formulation and implementation route for machine-learning-based industrial fault diagnosis that couples a mathematical model of the diagnosis pipeline with an explicit real-time execution scheme. By expressing the system as a sequence of operators acting on multichannel windows, from $\mathbf{y}_k$ to $\mathbf{Y}_w$, $\phi(\mathbf{Y}_w)$, $g_\theta(\tilde{\mathbf{z}}_w)$ and finally to alarm decisions, the approach clarifies how windowing, feature extraction, classification and decision logic interact under sampling and timing constraints. On top of this structure, a streaming algorithm was derived, together with a simple timing inequality that links feature-extraction and inference costs to the stride S and sampling period Ts. Experimental results on public bearing datasets and an industrial pump-line application show that a compact 1D-CNN instantiation of the framework can achieve higher macro F1 scores than conventional feature-based classifiers and deeper baselines, while still satisfying real-time requirements on an edge device with comfortable margin. Robustness tests under noise, missing samples and cross-domain transfer indicate that the proposed configuration retains useful performance when conditions deviate from the nominal training regime, although performance degradation is still evident in the presence of strong domain shifts. Future work will focus on extending the operator-based model to heterogeneous sensor suites and digital-twin interfaces, integrating domain adaptation mechanisms to mitigate distribution shift across assets and operating modes, and coupling the current architecture with explainability tools so that high-confidence decisions can be related to physically meaningful signal structures. A further line of work is the joint design of sensing and learning, including adaptive sampling and sensor placement strategies that explicitly account for the latency and resource constraints captured by the proposed timing model.

References

[1] G. Vashishtha, S. Chauhan, M. Sehri, R. Zimroz, P. Dumond, R. Kumar *et al.*, "A roadmap to fault diagnosis of industrial machines via machine learning: A brief review," *Measurement*, vol. 242, p. 116216, 2025.

[2] M. El-Brawany, E. E. Saadany, and K. El-Naggar, "Artificial intelligence-based data-driven prognostics in industry: A survey," *Computers & Industrial Engineering*, vol. 185, p. 109605, 2023.

[3] J. Zhao, W. Wang, J. Huang, and X. Ma, "A comprehensive review of deep learning-based fault diagnosis approaches for rolling bearings: Advancements and challenges," *AIP Advances*, vol. 15, no. 2, p. 020702, 2025.

[4] H. Wang, H. Wang, and X. Tang, "A review of deep learning in rotating machinery fault diagnosis and its prospects for port applications," *Applied Sciences*, vol. 15, no. 21, p. 11303, 2025.

[5] A. B. M. Leija, E. R. Beltrán, J. L. O. Mora, and J. O. V. Valadez, "Performance of machine learning algorithms in fault diagnosis for manufacturing systems: A comparative analysis," *Processes*, vol. 13, no. 6, p. 1624, 2025.

[6] V. R. Souza, J. C. Aguero, P. Herrero *et al.*, "A machine-learning based data-oriented pipeline for prognostics and health management systems," *Computers in Industry*, vol. 150, p. 103956, 2023.

[7] Y. Su and J. Lee, "Machine learning approaches for diagnostics and prognostics of industrial systems using open source data from phm data challenges: A review," *International Journal of Prognostics and Health Management*, vol. 15, no. 1, pp. 1–22, 2024.

[8] L. Lei, W. Li, S. Zhang, and H. Yu, "Research progress on data-driven industrial fault diagnosis methods," *Sensors*, vol. 25, no. 5, p. 1234, 2025, article number as in journal.

[9] Y. Hou, L. Zhang, Y. Xu *et al.*, "A lightweight transformer based on feature fusion and global–local parallel self-activation unit for bearing fault diagnosis," *Measurement*, vol. 235, p. 115773, 2024.

[10] L. Gong, C. Pang, G. Wang, and N. Shi, "Lightweight bearing fault diagnosis method based on improved residual network," *Electronics*, vol. 13, no. 18, p. 3749, 2024.

[11] W. Jiang, X. Zhang, C. Wang *et al.*, "Lightweight network bearing intelligent fault diagnosis based on vmd–fk–shufflenetv2," *Machines*, vol. 12, no. 9, p. 608, 2024.

[12] J. Cañao, J. S. Santos, and M. Antunes, "Explainable ai for industrial fault diagnosis: A systematic review," *Journal of Industrial Information Integration*, vol. 47, p. 100905, 2025.

[13] A. Maged *et al.*, "Explainable artificial intelligence techniques for accurate and trustworthy industrial fault diagnosis: A review," 2024.

[14] K. Jang, K. E. S. Pilario, N. Lee, I. Moon, and J. Na, "Explainable artificial intelligence for fault diagnosis of industrial processes," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 11, pp. 11 754–11 764, 2023.

[15] D. Goswami, F. Wang, and H. Li, "Observer-based fault detection for nonlinear systems under parameter variations," *Reliability Engineering & System Safety*, vol. 214, p. 107735, 2021.

[16] J. Du, S. Shah, and T. Chen, "Robust parity-space fault diagnosis methods for process systems," *Journal of the Franklin Institute*, vol. 356, pp. 3381–3403, 2019.

[17] R. Patton and H. Khorasgani, "Fault diagnosis of industrial systems: models, methods and trends," *Annual Reviews in Control*, vol. 52, pp. 1–17, 2021.

[18] K. Zhou, J. Doyle, and K. Glover, "Robust and optimal control revisited for diagnostic residual design," *IEEE Transactions on Automatic Control*, vol. 65, pp. 4890–4902, 2020.

[19] R. Mohammadi and A. Kabir, "Adaptive sliding mode observers for fault detection in uncertain processes," *ISA Transactions*, vol. 105, pp. 76–89, 2020.

[20] M. Esfahani and A. Aryanfar, "Hybrid analytical/data-driven fault detection for rotating machinery," *Mechanical Systems and Signal Processing*, vol. 150, p. 107277, 2021.

[21] P. Pan and X. Li, "Observer-based multi-fault diagnosis using hybrid models," *IEEE/ASME Transactions on Mechatronics*, vol. 27, pp. 393–403, 2022.

[22] Y. Lei, J. Lin, and M. Zuo, "Condition monitoring and fault diagnosis of rotating machinery using signal processing techniques," *Measurement*, vol. 138, pp. 546–558, 2018.

[23] R. Zimroz and T. Barszcz, "Condition indicators for mining machinery," *Mechanical Systems and Signal Processing*, vol. 140, pp. 106–121, 2020.

[24] H. Li and Z. Feng, "Signal demodulation techniques for bearing fault diagnosis," *Tribology International*, vol. 153, p. 106677, 2021.

[25] R. Sharma and J. Singh, "A comparative study of time–frequency methods for gearbox fault diagnosis," *Measurement*, vol. 195, p. 111113, 2022.

[26] Y. Wang and Q. He, "Envelope analysis revisited: advances and open challenges," *Mechanical Systems and Signal Processing*, vol. 144, p. 106886, 2020.

[27] L. Yang and H. Zhou, "Feature extraction for non-stationary rotating machinery signals," *IEEE Access*, vol. 10, pp. 21 342–21 355, 2022.

[28] G. Tang and C. Zhang, "Fault detection via enhanced spectral features," *Reliability, Safety and Availability Engineering*, vol. 35, pp. 211–225, 2021.

[29] F. Wang and X. Liu, "Feature-based svm for multi-class machinery fault diagnosis," *Measurement*, vol. 176, p. 109223, 2021.

[30] C. Li and H. Luo, "Random forest-based intelligent diagnosis of bearings," *Measurement*, vol. 165, p. 108066, 2020.

[31] Z. Yao and J. Xu, "A relevance-vector-machine framework for industrial fault detection," *Neurocomputing*, vol. 467, pp. 48–60, 2022.

[32] Z. Liu and P. Wang, "Hybrid knn classifiers for rotating machinery health monitoring," *ISA Transactions*, vol. 106, pp. 318–329, 2020.

[33] Z. He and W. Li, "Hybrid multiscale decomposition and ml for bearing faults," *Mechanical Systems and Signal Processing*, vol. 160, p. 107873, 2021.

[34] P. Zheng and H. Zhao, "Fusion of wpt features and boosted classifiers for fault diagnosis," *Measurement*, vol. 196, p. 111208, 2022.

[35] J. Kim and S. Lee, "Decomposition–based machine-learning fault diagnosis," *Mechanical Systems and Signal Processing*, vol. 146, p. 106735, 2021.

[36] X. Zhou and Z. Yan, "Combined statistical and ml features for motor fault diagnosis," *Applied Sciences*, vol. 11, p. 5632, 2021.

[37] Z. Peng and X. Liu, "Sparse representation for machinery fault classification," *Signal Processing*, vol. 160, pp. 69–83, 2019.

[38] H. Pan and W. Chen, "Dictionary learning for compound fault recognition," *Mechanical Systems and Signal Processing*, vol. 124, pp. 361–379, 2019.

[39] B. Yu and J. Sun, "Sparse decomposition and meta-learning for rotating machinery," *IEEE Access*, vol. 11, pp. 55 324–55 336, 2023.

[40] Y. Wen and H. Zhang, "Cnn-based diagnosis using raw vibration signals," *Measurement*, vol. 156, p. 107584, 2020.

[41] X. Jiang and P. Li, "Deep cnn for multi-domain feature learning in fault diagnosis," *Measurement*, vol. 165, p. 108159, 2020.

[42] H. Qi and Z. Sun, "Attention-guided feature extraction for bearing faults," *Reliability, Safety and Availability Engineering*, vol. 36, pp. 155–170, 2021.

[43] X. Gai and J. Lin, "Multi-sensor convolutional fusion network for rotating machinery," *Measurement*, vol. 210, p. 112545, 2023.

[44] P. Zhao and Y. Xu, "Gru-based fault detection under varying load," *Reliability Engineering & System Safety*, vol. 199, p. 106945, 2020.

[45] J. Wang and B. Hu, "Temporal deep learning for non-stationary machinery," *IEEE Access*, vol. 9, pp. 122 334–122 345, 2021.

[46] C. He and J. Zhao, "Transformer-based fault diagnosis for rolling bearings," *Measurement*, vol. 191, p. 110774, 2022.

[47] R. Yan and J. Zhou, "Gnn-based diagnosis of multi-sensor machinery," *Reliability, Safety and Availability Engineering*, vol. 38, pp. 242–259, 2023.

[48] Z. Chen and K. He, "Spatial–temporal gnn for gear transmission systems," *IEEE/ASME Transactions on Mechatronics*, vol. 28, pp. 562–573, 2023.

[49] X. Shen and H. Wu, "Attention-based cnn–rnn hybrid model for machinery signals," *Neurocomputing*, vol. 478, pp. 271–283, 2022.

[50] B. Sun and H. Li, "Robust cnn ensembles for noisy-motor fault detection," *Measurement*, vol. 173, p. 108653, 2021.

[51] D. Hou and J. Liu, "Improved attention nets for compound fault diagnosis," *IEEE Access*, vol. 10, pp. 41 287–41 299, 2022.

[52] Q. Shi and X. Guo, "Noise-aware feature recalibration network for mechanical systems," *Measurement*, vol. 211, p. 112631, 2023.

[53] S. Park and Y. Kim, "Pruned cnn for low-latency bearing diagnosis," *Measurement*, vol. 176, p. 109241, 2021.

[54] Y. Wu and X. Chen, "Quantized cnn for embedded motor fault diagnosis," *Sensors*, vol. 22, p. 5367, 2022.

[55] M. Liu and J. Chen, "Distilled lightweight model for gearbox diagnosis," *Electronics*, vol. 10, no. 4, p. 421, 2021.

[56] Z. Gao and R. Yan, "Mobilenet-based real-time bearing fault classifier," *Measurement*, vol. 205, p. 112215, 2023.

[57] A. Chang and G. Li, "Embedded edge system for real-time vibration-based diagnosis," *Measurement*, vol. 190, p. 110740, 2022.

[58] L. Wang and P. Zhang, "Edge–cloud cooperative monitoring for industrial drives," *IEEE/ASME Transactions on Mechatronics*, vol. 27, pp. 3023–3034, 2022.

[59] Y. Sun and C. Wu, "Fpga-accelerated diagnosis for induction motors," *ISA Transactions*, vol. 137, pp. 334–345, 2023.

[60] W. Gong and Q. Xu, "Incremental learning for rotating machinery diagnosis," *IEEE Access*, vol. 9, pp. 83 321–83 333, 2021.

[61] J. Zhang and P. Xu, "Online learning fault classifier with dynamic thresholds," *Reliability Engineering & System Safety*, vol. 214, p. 107704, 2021.

[62] H. Choi and S. Shin, "Communication-efficient fault detection under iiot constraints," *Measurement*, vol. 209, p. 112489, 2023.

[63] H. Deng and W. Liu, "Low-latency iiot fault diagnosis using distributed inference," *Applied Sciences*, vol. 11, p. 10145, 2021.

[64] C. Pan and Y. Tan, "Distributed diagnosis framework for production lines," *Mechanical Systems and Signal Processing*, vol. 185, p. 109854, 2023.