# Self-Supervised Learning for Rare Fault Detection in Industrial IoT Systems with Limited Labelled Data

[1]Kismat Chhillar, [2]Dharamdas Kumhar
[1]Assistant Professor, [2]Assistant Professor
[1]Department of Mathematical Sciences and Computer Applications
[1]Bundelkhand University, Jhansi, India

***Abstract*— Rare faults in Industrial Internet of Things (IIoT) systems are difficult to detect because equipment usually operates under normal conditions, leaving few labelled examples of abnormal behaviour. Although industrial sensors continuously generate operational data, annotating these observations requires technical expertise and reliable maintenance records. This paper proposes a self-supervised learning framework that uses unlabelled multivariate sensor data to support rare fault detection with limited labelled samples. Sensor measurements are first cleaned, normalized, and divided into temporal windows. A compact temporal encoder is then pretrained through contrastive learning to capture relationships among sensor variables and patterns over time. Physically plausible augmentations generate complementary views of each window, allowing the encoder to learn representations without requiring fault annotations during pretraining. The pretrained encoder is subsequently fine-tuned with a lightweight classifier using a small labelled subset, while class-weighted learning addresses the imbalance between normal and faulty observations. The evaluation is designed to compare the framework with supervised learning from scratch, autoencoder-based anomaly detection, and conventional classifiers under different label budgets. Performance is assessed using precision–recall area under the curve, fault-class precision, recall, F1-score, false alarm rate, and inference latency. An ablation study examines the contribution of self-supervised pretraining, with data partitions separated by time or operating run to reduce information leakage. The study investigates whether representations learned from unlabelled operational data can improve label efficiency and rare fault sensitivity while maintaining computational requirements suitable for industrial monitoring.**

***Index Terms*—Self-supervised learning, Industrial IoT, rare fault detection, limited labelled data, contrastive learning, condition monitoring.**

## I. Introduction

Industrial Internet of Things (IIoT) systems connect machines, sensors, controllers, and monitoring platforms to provide continuous information about industrial operations. Measurements such as vibration, temperature, pressure, electrical current, and rotational speed support equipment condition assessment and maintenance decisions. Condition-based maintenance relies on interpreting these measurements to identify deterioration and determine when intervention is necessary [1]. However, collecting large volumes of sensor data does not automatically produce reliable fault detection. Although deep learning provides methods for extracting diagnostic features from complex measurements, its practical application depends on the availability and quality of representative training data [2]. Industrial datasets may contain extensive records of normal operation but relatively few labelled fault observations. This limits the development of models capable of recognizing infrequent failures. The present study investigates self-supervised learning as a means of learning useful temporal representations from unlabelled operational data before adapting them to rare fault detection with limited labelled samples.

### 1.1.Industrial IoT Monitoring and Equipment Reliability

Continuous condition monitoring helps industrial operators recognize changes in equipment behaviour before they develop into serious failures. Distributed sensors observe different aspects of machine operation, while communication networks transfer measurements to processing platforms for analysis. For example, vibration signals may indicate changes in the mechanical condition of a rotating component, whereas temperature and current measurements may provide complementary evidence of friction, excessive loading, or electrical problems. Interpreting these measurements together can provide a more complete account of equipment condition than examining individual variables separately. However, the relationship between a sensor pattern and an equipment fault is rarely straightforward. Changes in load, production speed, environmental conditions, and measurement noise can alter the observed signals without indicating equipment damage. A useful detector must therefore distinguish fault-related changes from legitimate operational variation. For this study, these requirements motivate the use of a temporal encoder that processes multivariate sensor windows and learns relationships across both sensor channels and time.

### 1.2.Rare Faults, Class Imbalance, and Limited Annotation

Rare fault detection is challenging because the events of greatest interest are often the least represented in the training data. Equipment may operate normally for long periods, while particular mechanical defects or process disturbances occur only occasionally. This imbalance can cause a learning algorithm to favour the majority class and overlook minority-class observations, a central difficulty in classification with imbalanced data [3]. The problem is compounded by incomplete annotation. Maintenance records may identify that a component was replaced without establishing the precise onset or duration of the preceding fault. Producing reliable labels can therefore require expert interpretation of both sensor measurements and operating records. Furthermore, deliberately generating faults in working machinery may interrupt production or damage equipment. Under these conditions, overall classification accuracy provides an incomplete picture of detection quality. Precision–recall analysis is particularly relevant because it examines the identification of positive observations under class imbalance [4]. Accordingly, the proposed evaluation emphasizes fault-class precision, recall, F1-score, and precision–recall area under the curve, alongside the false alarm rate.

### 1.3.Self-Supervised Learning from Unlabelled Sensor Data

Self-supervised learning offers a way to use operational measurements without requiring a manually assigned fault label for every training observation. The model learns through a task constructed from the data itself, allowing representation learning to precede supervised classification. Earlier work on unsupervised multivariate time-series learning demonstrated that temporal encoders can learn transferable representations using objectives based on relationships between time-series segments [5]. Contrastive learning provides a related approach in which representations of compatible views of the same observation are encouraged to agree. SimCLR established an influential formulation of this approach in computer vision and demonstrated the importance of augmentation and projection-head design [6]. Its application to industrial signals nevertheless requires attention to temporal dependencies and the physical meaning of transformations. Time-series-specific research, including temporal and contextual contrasting, has demonstrated the potential of contrastive representations in settings with few labelled observations [7]. Hierarchical contrastive learning has also been used to learn contextual representations at different temporal resolutions [8]. These studies motivate the present framework, while leaving its effectiveness for the selected rare industrial faults to be established experimentally.

### 1.4.Research Objective and Proposed Approach

The central objective of this study is to determine whether self-supervised pretraining improves rare fault detection when only a small proportion of industrial sensor observations can be labelled. The proposed framework combines preprocessing, contrastive temporal representation learning, and limited-label classification. Sensor measurements are partitioned by operating run, equipment, or time, where the dataset permits, before normalization and window generation. A compact encoder is then pretrained using only the training partition and subsequently fine-tuned with a lightweight classifier using the available labelled

examples. Augmentation selection is treated as a methodological decision because the effects of time-series transformations depend on the dataset and model, as demonstrated by empirical comparisons of augmentation techniques [9]. In this study, candidate transformations will therefore be constrained to preserve plausible operating patterns and evaluated for their effect on fault sensitivity. Class-weighted learning will address imbalance during fine-tuning, and the detection threshold will be selected using validation data. Comparisons across different label budgets will examine whether pretraining offers a consistent advantage over supervised training from scratch.

### 1.5.Main Contributions of the Study

This paper makes three connected contributions. First, it proposes a compact self-supervised framework that learns temporal representations from unlabelled industrial measurements and adapts them to rare fault detection using a small labelled subset. Second, it establishes a controlled comparison across different label budgets using a supervised temporal model trained from scratch, an autoencoder-based anomaly detector, and a conventional feature-based classifier. The same labelled subsets will be used for the proposed classifier and its supervised counterpart, helping isolate the effect of pretraining from differences in annotation availability. Third, it includes an ablation analysis comparing the temporal model with and without self-supervised pretraining, together with measurements of detection performance and inference latency. Actual labelled fault counts will accompany the reported label percentages, and repeated runs will be used to assess variability. These contributions are intended to establish whether pretraining improves label efficiency and rare fault sensitivity under the selected experimental conditions, rather than assuming that benefits observed in other applications automatically transfer to industrial monitoring.

The remainder of the paper is organized as follows. Section 2 reviews related work on industrial fault detection, learning with limited labelled data, and self-supervised time-series representation learning. Section 3 presents the proposed methodology, including data preparation, contrastive pretraining, and limited-label classification. Section 4 describes the dataset, experimental settings, baseline methods, and evaluation measures. Section 5 presents and discusses comparative performance, the influence of label availability, and the ablation analysis. Finally, Section 6 concludes the paper and section 7 outlines limitations and directions for future research.

## II. Related Work

Research on industrial fault detection has progressed from statistical monitoring and manually designed features to models that learn diagnostic representations directly from sensor measurements. However, rare fault detection requires more than an expressive classifier: it also requires a learning strategy that remains useful when labelled abnormal observations are scarce. Relevant studies can be grouped into three areas: conventional and supervised fault detection, learning with limited labelled data, and self-supervised time-series representation learning. These areas address different aspects of the problem, including feature extraction, modelling normal behaviour, transferring knowledge, and exploiting unlabelled measurements. The following review examines their contributions and identifies the experimental questions that motivate the proposed framework.

### 2.1. Conventional and Supervised Fault Detection

Conventional fault detection commonly combines signal processing with statistical features and machine learning classifiers. Measurements are transformed into indicators such as amplitude statistics, spectral components, or time–frequency descriptors, which are subsequently used to distinguish operating conditions. Lei et al. [10] reviewed machine learning applications in machine fault diagnosis and presented a roadmap covering the development of diagnostic methods and their practical challenges. Their review provides a foundation for understanding the transition from feature engineering to representation learning. Ince et al. [11] demonstrated a one-dimensional convolutional neural network that integrates feature extraction and classification for motor fault detection, illustrating how diagnostic features can be learned directly within a compact network. Zhang et al. [12] investigated a convolutional architecture with a wide first-layer kernel for diagnosis from raw vibration signals under noise and changing working loads. These

studies establish the usefulness of learned features, although their reported performance should not be interpreted as evidence that supervised models can recognize rare faults with very few labelled examples. An alternative is to model normality and identify deviations. Schölkopf et al. [13] formulated a support-estimation approach underlying one-class support vector methods, while Liu et al. [14] introduced Isolation Forest, which detects unusual observations through their susceptibility to isolation by random partitioning. These approaches provide useful baselines when fault labels are unavailable. Nevertheless, applying them to industrial streams requires decisions about temporal features, operating-condition coverage, and detection thresholds. For the present study, their relevance lies in establishing whether self-supervised temporal representations provide an advantage over conventional classification and anomaly detection under the same data constraints.

### 2.2. Learning with Limited Labelled Data

Methods for learning with limited labels include deep anomaly detection, semi-supervised learning, transfer learning, and few-shot classification. Ruff et al. [15] introduced Deep Support Vector Data Description, which learns a representation through a one-class objective that encourages compactness around a centre. Zong et al. [16] proposed the Deep Autoencoding Gaussian Mixture Model, combining compressed representations and reconstruction information with density estimation for unsupervised anomaly detection. These methods reduce dependence on labelled anomalies, but detecting a deviation from normality is a different task from assigning a documented fault category. Semi-supervised methods instead combine labelled and unlabelled observations during training. Kingma et al. [17] developed deep generative models for semi-supervised classification, and Rasmus et al. [18] combined supervised learning with layer-wise denoising objectives in Ladder Networks. Tarvainen and Valpola [19] introduced Mean Teacher, using weight-averaged teacher predictions to provide consistency targets, while Sohn et al. [20] combined confidence-based pseudo-label selection with consistency across weakly and strongly augmented inputs in FixMatch. These foundational methods demonstrate ways to exploit unlabelled data, but their original results do not establish effectiveness for rare industrial faults. In particular, the possibility that unreliable pseudo-labels could reinforce errors on underrepresented faults must be examined rather than assumed away. Transfer learning offers another route: Pan and Yang [21] organized its principal settings and challenges, while Ganin et al. [22] developed domain-adversarial training to encourage features that support the source task while reducing domain distinguishability. Few-shot approaches provide a complementary perspective, with Matching Networks [23] predicting labels from a small support set and Prototypical Networks [24] classifying observations through distances to class prototypes. For industrial applications, these approaches motivate careful examination of source–target compatibility, available fault categories, and the number of independent examples supporting each class. The proposed study adopts a simpler pretraining and fine-tuning procedure to evaluate the specific contribution of unlabelled representation learning before introducing more elaborate adaptation mechanisms.

### 2.3. Self-Supervised Time-Series Representation Learning

Self-supervised representation learning is particularly relevant to industrial monitoring because sensor streams contain temporal structure even when diagnostic annotations are unavailable. Van den Oord et al. [25] introduced Contrastive Predictive Coding, which learns representations by predicting future observations in latent space using a contrastive objective. Although its original evaluation was not an industrial rare fault study, it provides an influential basis for learning predictive temporal features without manual labels. Tonekaboni et al. [26] developed Temporal Neighborhood Coding for non-stationary time series, distinguishing representations of neighbouring temporal segments from those of non-neighbouring segments. This approach highlights the value of temporal context, while its use for industrial faults would require checking whether neighbourhood assumptions preserve brief abnormal transitions. Zerveas et al. [27] proposed a transformer-based framework with unsupervised pretraining for multivariate time-series representation learning and evaluated its benefits on downstream classification and regression tasks. Their work supports the investigation of pretraining when labelled datasets are small, although architecture choice and computational requirements remain application-specific decisions. Zhang et al. [28] introduced time–frequency consistency for contrastive pretraining, aligning complementary temporal and spectral representations and evaluating transfer across several datasets, including mechanical fault detection. This

provides a direct connection between general time-series pretraining and machinery-related applications. Taken together, these studies motivate learning representations from unlabelled measurements, but they do not remove the need for a controlled rare fault evaluation. The present study therefore focuses on whether a compact temporal encoder improves detection across explicitly reported label budgets, using the same labelled subsets for comparison with supervised training from scratch. Its intended contribution is the joint assessment of label efficiency, fault sensitivity, false alarms, and inference cost, supported by an ablation that isolates the effect of pretraining. Table 1 summarizes the related work.

**Table 1:** Summary of Related Work

| Ref. | Authors / Study | Method or Approach | Main Contribution | Relevance and Limitation for the Present Study |
|---|---|---|---|---|
| [10] | Lei et al. (2020) | Review of machine learning for fault diagnosis | Organizes diagnostic methods and outlines research challenges. | Provides industrial context; does not experimentally establish performance under the proposed rare fault label budgets. |
| [11] | Ince et al. (2016) | One-dimensional convolutional neural network | Integrates feature extraction and classification for motor fault detection. | Supports compact temporal modelling; limited-label rare fault performance requires separate evaluation. |
| [12] | Zhang et al. (2017) | Wide-kernel convolutional neural network | Investigates diagnosis from raw vibration signals under noise and changing loads. | Informs robust feature learning; does not isolate the benefit of self-supervised pretraining. |
| [13] | Schölkopf et al. (2001) | One-class support estimation | Learns a boundary representing the support of a data distribution. | Provides a normality-based baseline; temporal features and thresholds require application-specific design. |
| [14] | Liu et al. (2008) | Isolation Forest | Detects anomalies through random partitioning and isolation. | Offers an unsupervised baseline; temporal dependencies are not explicitly modelled in its original formulation. |
| [15] | Ruff et al. (2018) | Deep Support Vector Data Description | Learns compact representations using a one-class objective. | Reduces reliance on fault labels; anomaly detection does not directly provide fault-category identification. |
| [16] | Zong et al. (2018) | Deep Autoencoding Gaussian Mixture Model | Combines representation learning, reconstruction information, and density estimation. | Provides a deep anomaly detection baseline; effectiveness on rare industrial faults needs validation. |
| [17] | Kingma et al. (2014) | Semi-supervised deep generative models | Uses labelled and unlabelled observations to | Establishes a limited-label learning strategy; original findings |

| | | | | |
|---|---|---|---|---|
| | | | support classification. | do not establish industrial fault performance. |
| [18] | Rasmus et al. (2015) | Ladder Networks | Combines supervised classification with layer-wise denoising objectives. | Motivates joint learning from labelled and unlabelled data; industrial temporal adaptation remains necessary. |
| [19] | Tarvainen and Valpola (2017) | Mean Teacher | Uses weight-averaged teacher predictions as consistency targets. | Provides a semi-supervised alternative; reliability for underrepresented faults must be assessed. |
| [20] | Sohn et al. (2020) | FixMatch | Combines confidence-based pseudo-labels with augmentation consistency. | Supports label-efficient classification; erroneous pseudo-labels may affect rare classes. |
| [21] | Pan and Yang (2010) | Transfer learning survey | Categorizes transfer learning settings and challenges. | Guides knowledge reuse across equipment or conditions; source–target compatibility requires examination. |
| [22] | Ganin et al. (2016) | Domain-adversarial neural networks | Learns task-relevant features while reducing domain distinguishability. | Motivates adaptation across operating conditions; domain alignment alone does not establish rare fault sensitivity. |
| [23] | Vinyals et al. (2016) | Matching Networks | Predicts classes using a small labelled support set. | Offers a few-shot classification strategy; representative support examples remain important. |
| [24] | Snell et al. (2017) | Prototypical Networks | Classifies observations by distance to learned class prototypes. | Provides a simple few-shot approach; prototypes may poorly represent faults with diverse signatures. |
| [25] | Van den Oord et al. (2018) | Contrastive Predictive Coding | Learns representations by predicting future information in latent space. | Supplies a foundation for temporal self-supervision; original evaluation did not establish industrial rare fault performance. |
| [26] | Tonekaboni et al. (2021) | Temporal Neighborhood Coding | Learns representations from temporal neighbourhood relationships. | Addresses non-stationary time series; neighbourhood assumptions need checking around brief fault transitions. |
| [27] | Zerveas et al. (2021) | Transformer-based time-series pretraining | Learns multivariate representations | Supports pretraining with limited labels; computational suitability |

| | | | | |
|---|---|---|---|---|
| | | | for downstream classification and regression. | requires deployment-specific assessment. |
| [28] | Zhang et al. (2022) | Time–frequency contrastive consistency | Aligns temporal and spectral representations for transferable pretraining. | Includes mechanical fault evaluation; the proposed rare fault budgets, false alarms, and inference costs need separate assessment. |

## III. Proposed Methodology

The proposed methodology uses self-supervised representation learning to detect rare faults in Industrial Internet of Things (IIoT) systems when only a small amount of labelled training data is available. The framework consists of four stages: problem formulation and data preparation, self-supervised temporal representation learning, fault classification with limited labels, and operational fault detection. During pretraining, a compact temporal encoder learns patterns from unlabelled multivariate sensor windows. The encoder is then fine-tuned with a lightweight classifier using a small labelled subset containing normal and faulty observations. Class-weighted learning addresses the unequal representation of these classes, while a validation-based threshold controls the balance between fault sensitivity and false alarms. The methodology is designed to test whether pretraining provides a measurable advantage over supervised training from scratch, without assuming that unlabelled data necessarily improve detection.

### 3.1. Problem Formulation and Data Preparation

The fault detection problem is formulated as binary classification of multivariate industrial sensor windows into normal and faulty operating conditions. Sensor channels may include vibration, temperature, pressure, electrical current, and rotational speed, depending on the selected dataset. Raw records are divided into training, validation, and test partitions by equipment, operating run, or time before preprocessing and window generation. This separation prevents overlapping windows or closely related observations from appearing across partitions. Short missing intervals are handled using a predefined imputation procedure, while records with excessive missingness are excluded. Sensor channels are standardized using statistics calculated exclusively from the training partition. The normalized measurements are then segmented into fixed-length windows within each partition. Window labels are assigned using documented fault intervals and a predefined labelling rule, with uncertain transition windows excluded from supervised evaluation. A small labelled subset is selected from the training windows, while the remaining training labels are hidden. All training windows can participate in self-supervised pretraining, but validation and test windows remain excluded.

**Representation of multivariate sensor data**

The industrial sensor stream is represented as

$$[X = \{x_t\}_{t=1}^{T}, \quad x_t = \left[x_{t,1}, x_{t,2}, \ldots, x_{t,d}\right]^{\top} \in R^{d}] \qquad (1)$$

Here, T is the total number of time steps, d is the number of sensor channels, and $(x_{t,k})$ is the measurement recorded by sensor channel k at time t. The superscript $(\top)$ denotes vector transpose.

**Channel-wise normalization**

Each sensor measurement is standardized as

$$\tilde{x}_{t,k} = \frac{x_{t,k} - \mu_k^{tr}}{\sigma_k^{tr} + \varepsilon}, \quad k = 1, 2, \ldots, d \tag{2}$$

Here $(\mu_k^{tr})$ and $(\sigma_k^{tr})$ are the mean and standard deviation of channel $(k)$, calculated only from the training partition. The constant $(\varepsilon > 0)$ prevents division by zero. The same training-derived statistics are applied to validation, test, and operational measurements.

**Generation of temporal windows**

For window length $(L)$ and $(S)$, the $(i)$-th window within a continuous record is

$$W_i = \begin{bmatrix} \tilde{\mathbf{x}}_{s_i}^{\top} \\ \tilde{\mathbf{x}}_{s_i+1}^{\top} \\ \vdots \\ \tilde{\mathbf{x}}_{s_i+L-1}^{\top} \end{bmatrix} \in \mathbb{R}^{L \times d}, \quad s_i = 1 + (i-1)S \tag{3}$$

Here, $(s_i)$ is the starting sample index of window $(i)$, $(L)$ is the number of samples per window, and $(S)$ is the number of samples between successive window starts. Each row represents one time step, and each column represents one sensor channel. Windows are generated separately for each continuous record and must not cross partition boundaries.

**Number of windows**

For a continuous record containing $(T_r \geq L)$ samples, the number of complete windows is

$$\left[N_r = \left\lfloor \frac{T_r - L}{S} \right\rfloor + 1\right] \tag{4}$$

Here, $(\lfloor \cdot \rfloor)$ denotes the floor function. Records shorter than $(L)$ produce no complete windows unless a separate padding procedure is explicitly defined.

**Labelled and unlabelled training subsets**

The training windows are divided into labelled and unlabelled subsets:

$$\left[D_L = \{(W_i, y_i)\}_{i=1}^{N_L}, \quad D_U = \{W_j\}_{j=1}^{N_U}, \quad N_L \ll N_U\right] \tag{5}$$

Here, $(N_L)$ and $(N_U)$ denote the numbers of labelled and unlabelled training windows, respectively. The binary label is defined as

$$y_i = \begin{cases} 0, & \text{normal operation,} \\ 1, & \text{faulty operation.} \end{cases}$$

The subsets contain distinct training windows. During pretraining, windows from both subsets are used without their labels. During supervised fine-tuning, only $(D_L)$ supplies labelled examples.

**Label-budget definition**

The proportion of training windows with available labels is

$$[\rho = \frac{N_L}{N_{tr}}, \qquad N_{tr} = N_L + N_U] \qquad (6)$$

Here, $(N_{tr})$ is the total number of training windows. Label budgets of $(\rho = 0.01), (0.05), and (0.10)$ correspond to 1%, 5%, and 10% labelled training data, respectively. Each budget is reported alongside its actual normal-window count, fault-window count, and number of independent fault events. Validation labels are counted separately.

### 3.2. Self-Supervised Representation Learning

Self-supervised pretraining enables the model to learn temporal representations from industrial sensor measurements without using fault labels. A compact one-dimensional convolutional encoder processes each multivariate sensor window and extracts a fixed-length feature vector. Two independently augmented views are generated from every window using transformations that preserve plausible equipment behaviour, such as low-amplitude noise or limited masking of short intervals. Both views pass through the same encoder and projection head. A contrastive learning objective encourages representations of the same window to remain similar while distinguishing them from other windows in the batch. Where feasible, overlapping or immediately adjacent windows from the same operating record are excluded from the same batch to reduce comparisons between nearly identical observations. Pretraining uses only the training partition, with all labels hidden. After pretraining, the projection head is discarded, and the encoder is retained for supervised fault classification.

**Generation of augmented views**

For an input sensor window $(W_i)$, two augmented views are generated as follows:

$$[W_i^{(1)} = a_i^{(1)}(W_i), \qquad W_i^{(2)} = a_i^{(2)}(W_i)] \qquad (7)$$

Here, $(a_i^{(1)})$ and $(a_i^{(2)})$ are transformations sampled independently from the predefined augmentation distribution $(A)$. The superscripts identify the two views, rather than powers.

**Temporal feature extraction**

The shared encoder converts each augmented view into a feature vector:

$$[h_i^{(v)} = f_\theta!\left(W_i^{(v)}\right), \qquad v \in \{1,2\}\backslash tag8] \qquad (8)$$

Here, $(f_\theta)$ is the temporal encoder, $(\theta)$ denotes its trainable parameters, and $(h_i^{(v)} \in R^m)$ is an $(m)$ -dimensional representation.

**Projection and normalization**

A projection head maps the extracted features into the space used for contrastive learning:

$$q_i^{(v)} = g_\phi\left(h_i^{(v)}\right), \quad z_i^{(v)} = \frac{q_i^{(v)}}{\left(|q_i^{(v)}|_2, \varepsilon\right)} \tag{9}$$

Here, $(g_\phi)$ is the projection head, $(\phi)$ denotes its parameters, $(|\cdot|_2)$ is the Euclidean norm, and $(\varepsilon > 0)$ prevents division by zero.

**Contrastive learning objective**

A batch containing $(B)$ original windows produces $(2B)$ augmented embeddings. For an anchor embedding indexed by $(a)$ let $(p(a)$ identify the other augmented view of the same original window. The contrastive loss is

$$\ell_a = -\log\left[\frac{\exp\left(\text{sim}\left(\mathbf{z}_a, \mathbf{z}_{p(a)}\right)/\tau\right)}{\sum_{\substack{b=1 \\ b\neq a}}^{2B} \exp\left(\text{sim}\left(\mathbf{z}_a, \mathbf{z}_b\right)/\tau\right)}\right] \tag{10}$$

Here, $(\tau > 0)$ is the temperature parameter. The denominator includes the positive partner and all other embeddings except the anchor itself. Cosine similarity is defined as

$$sim(u, v) = \frac{u^{\top} v}{|u|_2 |v|_2} \tag{11}$$

where $(u)$ and $(v)$ are nonzero embedding vectors.

**Overall pretraining loss**

The self-supervised loss is averaged over all augmented views in the batch:

$$\mathcal{L}_{\text{SSL}} = \frac{1}{2B} \sum_{a=1}^{2B} \ell_a \tag{12}$$

The encoder and projection-head parameters are optimized to minimize the expected loss across sampled training batches and augmentations:

$$(\theta^*, \phi^*) = \arg\min_{\theta,\phi} \mathbb{E}_{\mathcal{B},\mathcal{A}}\left[\mathcal{L}_{\text{SSL}}\right] \tag{13}$$

Here, $(B)$ denotes training-batch sampling. Following optimization, the pretrained encoder parameters $(\theta^*)$ initialize the fault detection model, while the projection head is removed.

### 3.3. Rare Fault Detection with Limited Labels

Following self-supervised pretraining, a lightweight classification head is attached to the temporal encoder to distinguish normal operation from faulty conditions. The encoder and classifier are fine-tuned using only the selected labelled training subset, with a smaller learning rate applied to the encoder to preserve useful pretrained features. Weighted binary cross-entropy addresses the unequal representation of normal and faulty observations by assigning greater importance to the underrepresented class. For a controlled comparison, the supervised baseline uses the same architecture, labelled samples, class weights, and model-selection procedure, but starts from randomly initialized encoder parameters. Early stopping uses validation data, and the detection threshold is selected to maximize validation recall while satisfying a predefined false positive rate limit. Experiments are repeated across label budgets and random seeds to assess performance variability. The primary task concerns rare fault classes represented in the labelled training subset; detecting entirely unseen fault categories requires a separate evaluation.

**Fault probability estimation**

For an input sensor window ($W_i$), the model estimates the probability of faulty operation as

$$[p_i = \sigma\left(c_\psi\left(f_\theta\left(W_i\right)\right)\right) \qquad (14)$$

The sigmoid function is defined as

$$[\sigma(u) = \frac{1}{1+\exp exp\,(-u)}]$$

Here, ($f_\theta$) is the temporal encoder, ($c_\psi$) is the classification head, and ($p_i \in (0,1)$) is the predicted fault probability. The encoder is initialized using the pretrained parameters ($\theta^*$).

Class-weight calculation

Let ($n_0$) and ($n_1$) represent the numbers of labelled normal and faulty training windows, respectively. The class weights are

$$[w_0 = \frac{N_L}{2n_0}, \qquad w_1 = \frac{N_L}{2n_1}] \qquad (15)$$

Where

$$[N_L = n_0 + n_1]$$

Both classes must be present in the labelled subset. These weights give each class equal total weight in the training objective.

**Weighted classification loss**

The model is fine-tuned by minimizing weighted binary cross-entropy:

$$[Lsup = -\frac{1}{N_L}\sum_{i=1}^{N_L}\left[w_1 y_i \log log\left(p_i\right) + w_0\left(1 - y_i\right)\log log\left(1 - p_i\right)\right]] \qquad (16)$$

Here, ($y_i = 1$) indicates a faulty window, ($y_i = 0$) indicates a normal window, and ($w_1$) and ($w_0$) are the corresponding class weights. In implementation, a numerically stable loss calculated directly from classifier logits should be used.

**Classification decision**

The predicted label is obtained by comparing the fault probability with a threshold (δ):

$$\hat{y}_i(\delta) = \begin{cases} 1, & p_i \geq \delta, \\ 0, & p_i < \delta. \end{cases} \quad (17)$$

A prediction of 1 indicates faulty operation, whereas a prediction of 0 indicates normal operation.

**Validation-based threshold selection**

The threshold is selected to maximize validation recall subject to an acceptable validation false positive rate:

$$\delta^* \in \arg arg\,(\delta)$$

$$\text{Subject to} \quad FPR_{val}(\delta) \leq \alpha \quad (18)$$

Here, (α) is the maximum acceptable false positive rate, specified before test evaluation. If multiple thresholds achieve the same recall, the threshold with the lower validation false positive rate is selected. The resulting threshold is kept fixed during testing.

**Detection performance measures**

Fault-class precision and recall are calculated as

$$Precision = \frac{TP}{TP+FP}, \qquad Recall = \frac{TP}{TP+FN} \quad (19)$$

The F1-score and false positive rate are

$$F_1 = \frac{2TP}{2TP+FP+FN}, \qquad FPR = \frac{FP}{FP+TN} \quad (20)$$

Here, ($TP$), ($FP$), ($TN$), and ($FN$) denote true positives, false positives, true negatives, and false negatives, respectively. Precision–recall area under the curve is additionally calculated from prediction scores across thresholds. Any undefined metric caused by a zero denominator is handled using an explicitly reported convention.

**3.4. Detection Workflow and Operational Use**

During operational monitoring, incoming sensor measurements undergo the same alignment, missing-value handling, normalization, and window segmentation procedures used during model development. Each completed window is processed by the fine-tuned encoder and classifier, and the resulting fault probability is compared with the fixed validation-selected threshold. Predictions, scores, and timestamps are recorded for operator review. An optional persistence rule requires several consecutive positive windows before generating an alert, helping suppress isolated predictions. However, this rule may

delay detection or miss short faults, so its parameters are selected using validation data and remain unchanged during testing. Window-level classification results are reported separately from event-level alert performance. Computational assessment includes preprocessing time, inference time, model size, and memory use on identified hardware. Where reliable fault-onset annotations are available, detection delay is measured from fault onset to the first qualifying alert. Model parameters remain fixed throughout test evaluation.

**Window-level fault indicator**

For the $(i)$-th completed sensor window, the binary fault indicator is

$$b_i = \mathbb{I}[p_i \geq \delta^*] \tag{21}$$

Here, $(I[\cdot])$ equals 1 when the condition is satisfied and 0 otherwise. The threshold $(\delta^*)$ is determined using validation data.

**Persistence-based alert generation**

If an alert requires $(K)$ consecutive positive windows, the alert indicator is

$$[A_i = \prod_{j=0}^{K-1} b_{i-j}, \qquad i \geq K] \tag{22}$$

Here, $(K)$ is a positive integer. The indicator $(A_i)$ equals 1 only when all $(K)$ consecutive predictions are positive. Setting $(K = 1)$ produces immediate threshold-based alerts. Consecutive positive alert indicators can be grouped into a single alert episode for event-level evaluation.

**Window duration and output interval**

For sampling frequency $(f_s)$, window length $(L)$, and stride $(S)$, the time spanned by a window and the interval between successive outputs are

$$T_{span} = \frac{L-1}{f_s}, \qquad T_{step} = \frac{S}{f_s} \tag{23}$$

Here, $(T_{span})$ is the elapsed time between the first and last samples in a window, while $(T_{step})$ is the interval between successive window completions. These expressions assume uniformly sampled measurements.

**Processing time per window**

The processing time for a completed window is

$$T_{proc} = T_{pre} + T_{enc} + T_{cls} + T_{rule} \tag{24}$$

Here, $(T_{pre})$ is preprocessing time, $(T_{enc})$ is encoder execution time, $(T_{cls})$ is classification time, and $(T_{rule})$ is alert-rule computation time. Communication or queueing delays, where applicable, are measured separately.

**Processing throughput requirement**

For a single sequential processing worker, a basic condition for keeping pace with incoming windows is

$$T_{proc} \leq T_{step} \qquad (25)$$

This condition concerns processing capacity. It does not guarantee an acceptable detection delay, which also depends on window construction, fault characteristics, persistence settings, and other system delays.

**Event detection delay**

For a detected fault event $(e)$, the detection delay is

$$D_e = t_e^{alert} - t_e^{onset} \qquad (26)$$

Here, $(t_e^{onset})$ is the documented fault-onset time, and $(t_e^{alert})$ is the time of the first qualifying alert associated with that event under a predefined matching rule. Events without a qualifying alert are counted as missed events. Delay statistics are reported for detected events together with the event detection rate, preventing missed events from being concealed by a low average delay. Figure 1 illustrates the proposed methodology.

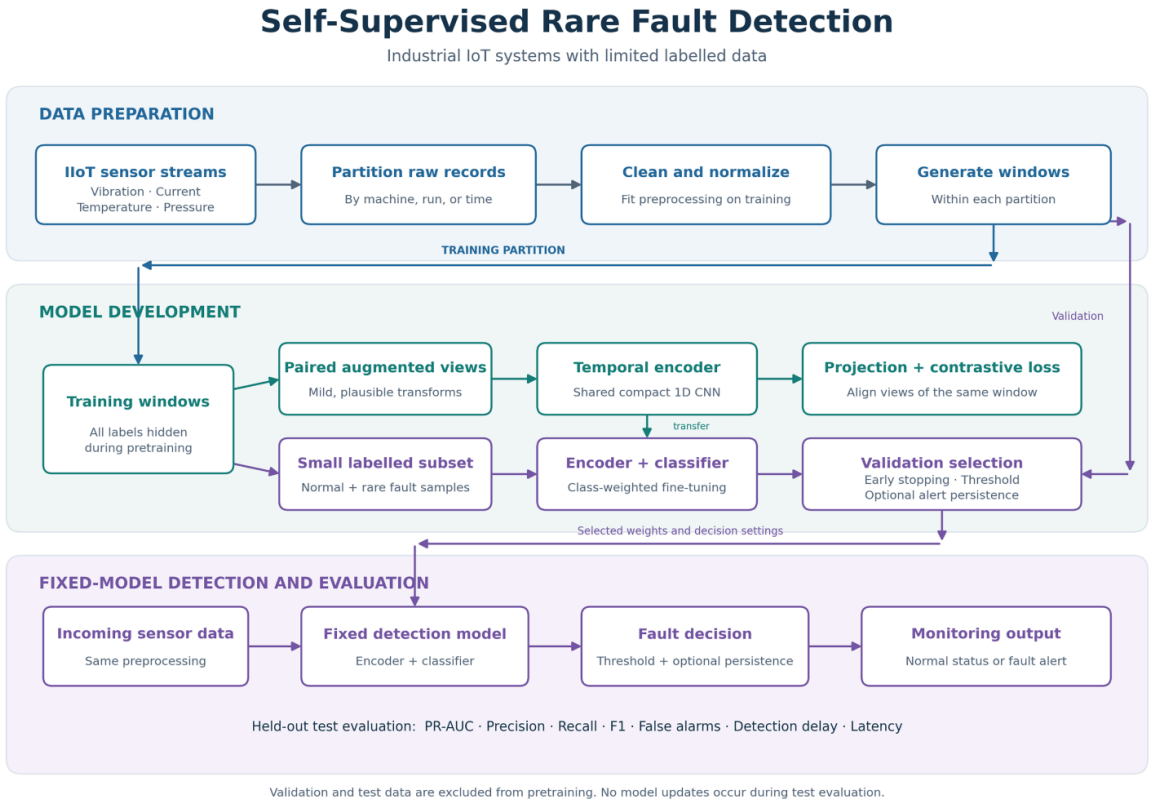


**Figure 1:** Proposed Methodology

## IV. Experimental Setup

The experimental setup is designed to evaluate whether self-supervised pretraining improves rare fault detection when only a small proportion of industrial sensor observations have labels. The evaluation compares the proposed framework with conventional classification, supervised temporal learning, and

reconstruction-based anomaly detection under controlled data conditions. Particular attention is given to the separation of operating records, the availability of labelled fault examples, and the distinction between window-level predictions and independent fault events. Experiments examine detection performance across several label budgets, while an ablation study isolates the contribution of self-supervised pretraining. Computational measurements are included to assess the processing requirements of the resulting detector. All preprocessing decisions, hyperparameter searches, threshold selection procedures, and evaluation rules are established without accessing test labels.

### 4.1. Dataset and Label Budgets

The experiments will use an industrial condition-monitoring dataset containing time-stamped sensor measurements, documented normal and faulty operating conditions, and sufficient record-level information to construct independent training, validation, and test partitions. The selected dataset must be described in terms of equipment type, sensor channels, sampling frequency, operating conditions, fault categories, recording duration, and the number of independent machines or operating runs. A dataset containing only balanced laboratory fault classes will not be described as naturally representing rare industrial faults; if fault prevalence is reduced experimentally, the resulting setting will be identified explicitly as a controlled rare fault scenario. Records will be divided into approximately 60% training, 20% validation, and 20% testing, subject to maintaining independent groups and adequate fault coverage. Equipment-level or run-level separation will be preferred, while chronological separation with an appropriate boundary gap will be used when independent groups are unavailable. Normalization statistics will be estimated from training records, and windows will be generated separately within each partition. Label budgets of 1%, 5%, and 10% will be selected from the training windows, with a fully labelled training condition included as an optional reference. Sampling will retain approximately the training class proportions, subject to the presence of both classes, and any minimum fault-coverage constraint will be disclosed because it uses annotation information. The exact labelled normal count, labelled fault count, number of represented fault categories, and number of independent fault events will accompany each budget. Larger labelled subsets will contain the smaller subsets within each repetition to support consistent comparisons across budgets. All training windows may be used during self-supervised pretraining with their labels hidden, whereas validation and test windows remain excluded. Validation annotation requirements will be reported separately so that the stated training budgets do not conceal additional labelling effort.

### 4.2. Baseline Methods

Three principal baselines will be used to determine whether improvements arise from self-supervised representation learning rather than model capacity or additional labelled examples. The first baseline is a supervised temporal classifier with the same encoder architecture and classification head as the proposed framework, trained from random initialization using exactly the same labelled training subset. Both models will use the same class-weight calculation, fine-tuning search space, early stopping rule, and validation-based threshold procedure. The second baseline is a reconstruction-based autoencoder trained on the available training windows without using their labels, with reconstruction error serving as the anomaly score. Because unlabelled training data may contain faults, the training contamination level will be reported; an additional autoencoder trained on verified normal observations, if included, will be identified as requiring extra normal-status information. The third baseline is a Random Forest classifier trained on statistical features extracted from the same labelled windows, including channel-wise mean, standard deviation, root mean square, minimum, maximum, skewness, and kurtosis. Feature extraction and any feature selection will be fitted using training data only. Each baseline will receive the same validation records and comparable hyperparameter-search effort, while its training time and access to unlabelled observations will be documented. The main ablation will compare the proposed pretrained temporal model against its randomly initialized counterpart. A supplementary comparison with a frozen pretrained encoder may be included to examine whether supervised fine-tuning improves the usefulness of the learned features. These comparisons support a focused assessment of label efficiency without requiring an unnecessarily large collection of models.

### 4.3. Implementation and Training Configuration

The temporal encoder will initially comprise three one-dimensional convolutional blocks with 32, 64, and 128 output channels, followed by global average pooling to produce a 128-dimensional representation. Each block will include normalization and a nonlinear activation, and the pretraining projection head will map the representation through a small multilayer network to a normalized 64-dimensional embedding. These values constitute starting configurations rather than experimentally established optimal settings. Candidate window lengths will be chosen according to the sampling frequency and the duration of relevant equipment behaviour, with the final window duration and stride reported in both samples and seconds. Pretraining will use Adam with an initial learning rate of $(10^{-3})$, a batch size of 128 original windows where memory permits, and a maximum of 100 epochs. A batch therefore contains 256 augmented views. Fine-tuning will use initial learning rates of $(10^{-4})$ for the encoder and $(10^{-3})$ for the classification head, with a maximum of 50 epochs and early stopping after 10 epochs without improvement in validation average precision. The contrastive temperature will be selected from a small predefined set, such as 0.1, 0.2, and 0.5, while augmentation strengths will be constrained by sensor characteristics and selected without test access. Repeated experiments will use five recorded seeds, each controlling labelled-subset sampling, model initialization, augmentation sampling, and training order. The implementation report will specify the software versions, operating system, processor, accelerator, available memory, and numerical precision. Hyperparameter tuning effort and validation access will be disclosed because repeatedly selecting configurations with labelled validation data contributes to the overall supervision required by the approach. Figure 2 summarizes the experimental setup.

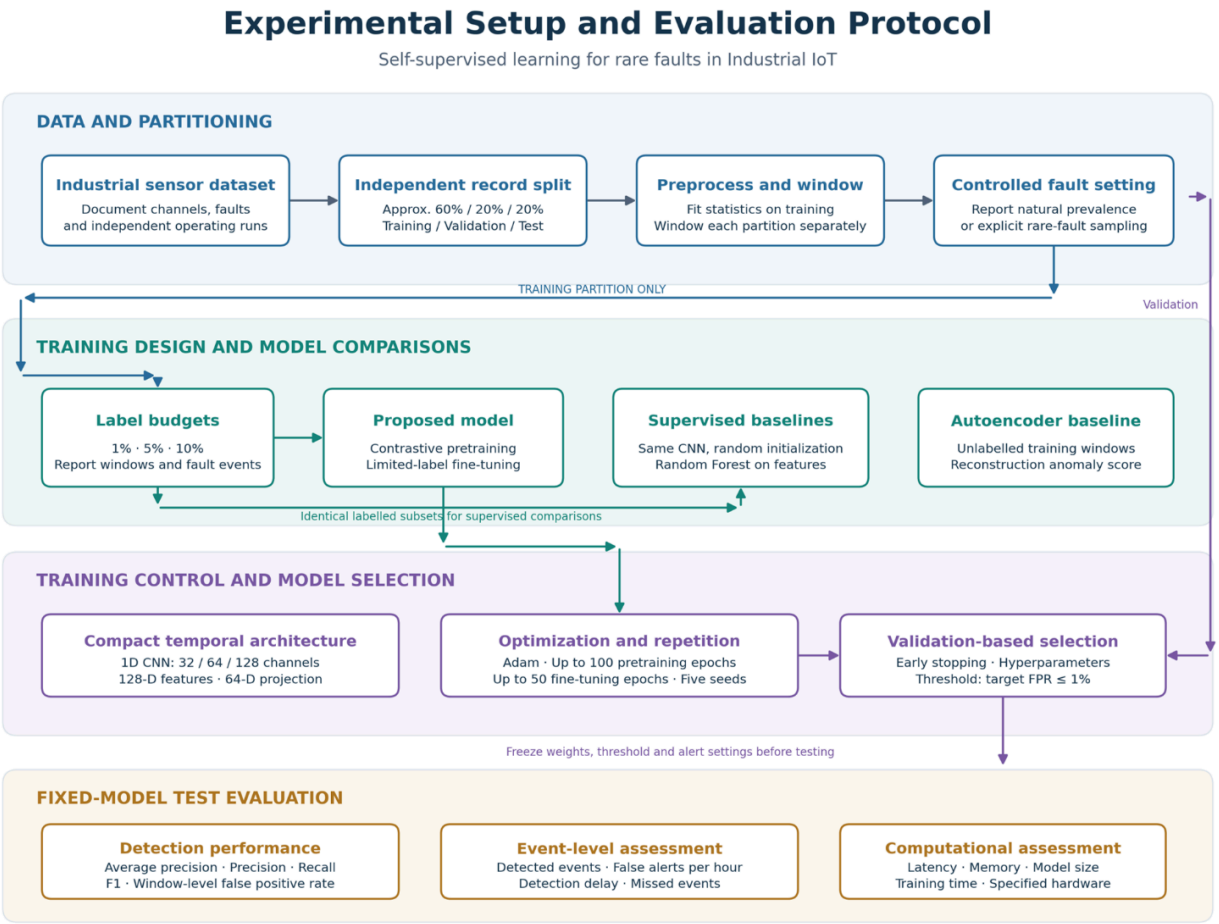


**Figure 2:** Experimental Setup

### 4.4. Evaluation Measures and Statistical Reporting

Detection quality will be assessed primarily using average precision, fault-class precision, recall, F1-score, and window-level false positive rate. Average precision will be reported explicitly as the selected

summary of the precision–recall curve, avoiding ambiguity with trapezoidal precision–recall area. Overall accuracy may be provided as a supplementary measure, but it will not serve as the main criterion because a detector can achieve high accuracy by predicting normal operation in an imbalanced dataset. Each model's threshold will be selected on validation data to maximize recall while maintaining a predefined false positive rate limit, provisionally set at 1%, and then fixed during testing. The validation constraint will not be interpreted as guaranteeing a 1% test false positive rate. Where reliable event annotations exist, evaluation will additionally report the proportion of independent fault events detected, false alert episodes per hour of normal operation, and detection delay for successfully detected events. Any persistence rule will be evaluated separately from raw window predictions, with a fixed event-matching rule and no retrospective adjustment of predictions using test labels. Results will be summarized as mean and standard deviation across the five matched repetitions, and uncertainty intervals, where calculated, will resample independent machines, runs, or events rather than individual overlapping windows. Comparisons will emphasize paired performance differences and their uncertainty. Computational evaluation will measure model size, memory consumption, training time, and batch-size-one inference latency on the stated hardware, including warm-up and repeated timing measurements. Preprocessing and communication delays will be distinguished from neural-network inference, and detection delay will include the effects of window completion and alert persistence. These procedures provide a consistent basis for assessing both rare fault sensitivity and the practical cost of the proposed framework.

## V. Results and Discussion

This section illustrates the evaluation of the proposed self-supervised framework through comparative detection performance, sensitivity to label availability, and ablation analysis. For numerical consistency, the example uses a held-out test set containing 10,000 sensor windows, comprising 9,500 normal windows and 500 faulty windows. The resulting fault prevalence is 5%. The principal comparison assumes a 5% training label budget, with thresholds selected separately on validation data and fixed before testing. Average precision summarizes performance across decision thresholds, while precision, recall, F1-score, and false positive rate describe performance at the selected operating point. The following values represent a hypothetical evaluation scenario and do not establish the effectiveness of the proposed method.

### 5.1. Comparative Detection Performance

In the illustrative comparison, the proposed framework achieves an average precision of 0.861, precision of 84.36%, recall of 82.00%, and F1-score of 83.16%. At its selected threshold, it correctly identifies 410 of the 500 faulty windows and incorrectly flags 76 of the 9,500 normal windows, producing a false positive rate of 0.80%. The supervised temporal CNN achieves an average precision of 0.779 and recall of 76.00%, corresponding to 380 correctly detected faulty windows and 120 missed faulty windows. Its precision and F1-score are 81.55% and 78.67%, respectively. The Random Forest baseline detects 355 faulty windows, while the autoencoder detects 325. Within this hypothetical setting, the proposed model therefore identifies 30 more faulty windows than the supervised CNN, 55 more than Random Forest, and 85 more than the autoencoder. These differences concern windows rather than independent failures: several detected windows may originate from the same fault event.

The comparison illustrates the type of evidence needed to support a claim that self-supervised pretraining improves fault discrimination under limited supervision. Relative to the supervised CNN, the proposed model increases average precision by 0.082 and F1-score by 4.49 percentage points while reducing false positive predictions from 86 to 76. Because these two models use the same architecture and labelled subset, a measured difference of this kind would support examining pretraining as the source of the improvement. Nevertheless, a single comparison would not establish robustness across machines, operating conditions, or fault categories. The autoencoder's hypothetical false positive rate of 1.51%, despite a validation target of at most 1%, also illustrates why a validation constraint cannot guarantee the same operating behaviour on unseen data. Any actual interpretation should therefore consider repeated runs, uncertainty intervals, class prevalence, and differences between validation and test conditions. Table 2 shows the detection

performance. Figure 3 and Figure 4 shows the comparison of F1-score and false positive rate positives respectively.

**Table 2:** Illustrative detection performance at a 5% training label budget.

| Method | Average Precision | Precision (%) | Recall (%) | F1-Score (%) | False Positive Rate (%) |
|---|---|---|---|---|---|
| Random Forest | 0.718 | 78.89 | 71 | 74.74 | 1 |
| Supervised Temporal CNN | 0.779 | 82 | 76 | 78.67 | 0.91 |
| Autoencoder | 0.654 | 69.44 | 65 | 67.15 | 1.51 |
| Proposed Self-Supervised Framework | 0.861 | 84.36 | 82 | 83.16 | 0.8 |

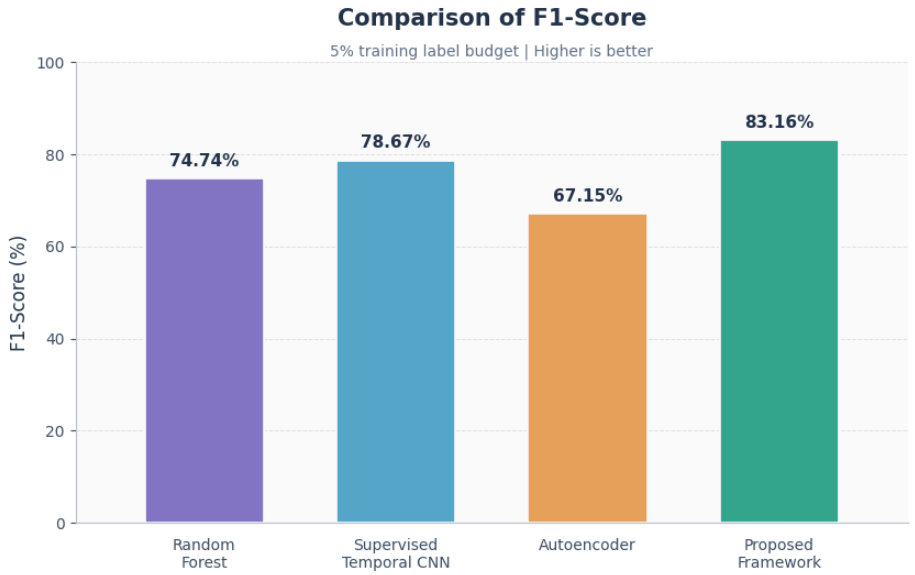


Figure 3: Comparison of F1 Score

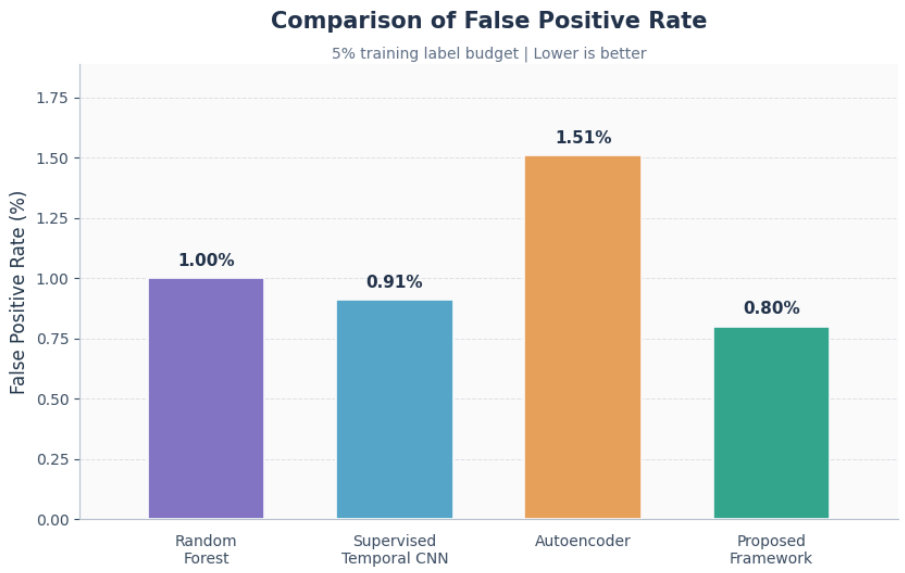


**Figure 4:** Comparison of False Positive Rate

**Table 3:** Illustrative confusion-matrix counts for the same test set.

| Method | True Positives (TP) | False Negatives (FN) | False Positives (FP) | True Negatives (TN) |
|---|---|---|---|---|
| Random Forest | 355 | 145 | 95 | 9,405 |
| Supervised Temporal CNN | 380 | 120 | 86 | 9,414 |
| Autoencoder | 325 | 175 | 143 | 9,357 |
| Proposed Self-Supervised Framework | 410 | 90 | 76 | 9,424 |

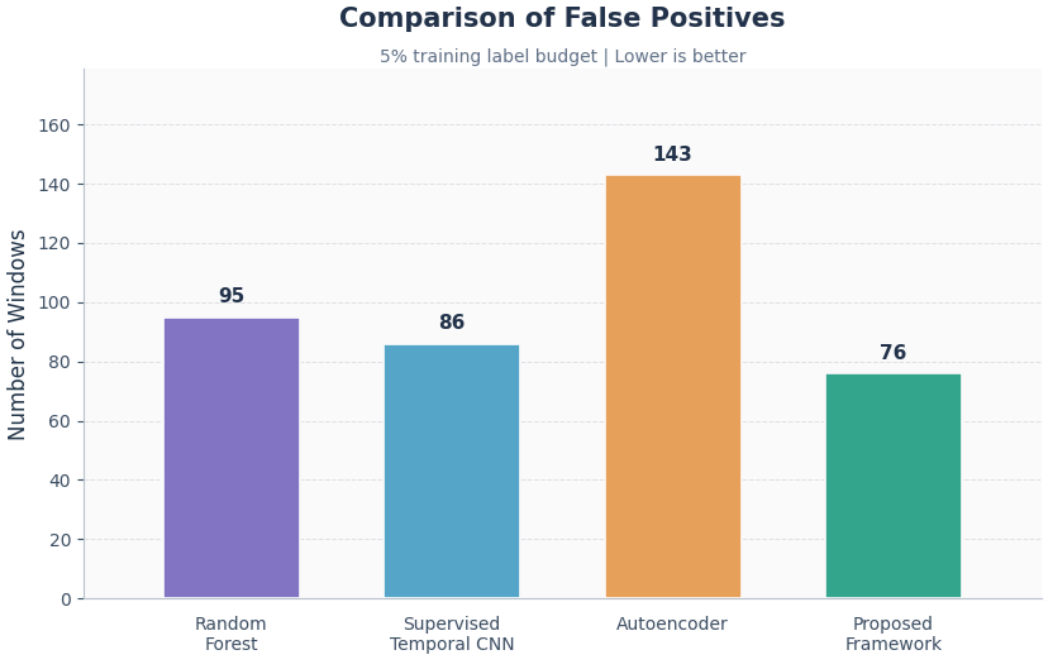


**Figure 5:** Comparison of False Positives

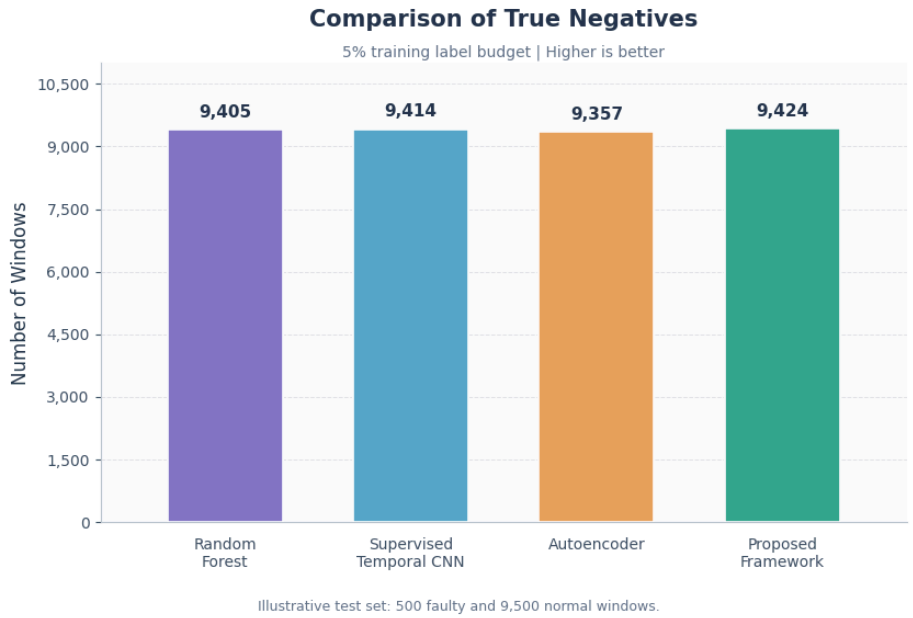


**Figure 6:** Comparison of True Negatives

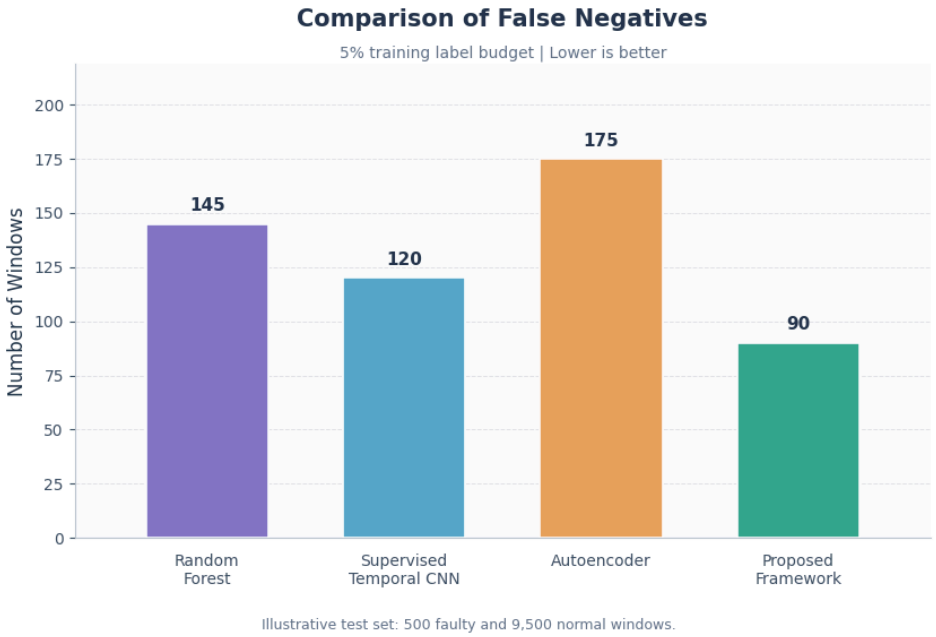


**Figure 7:** Comparison of False Negatives

### 5.2. Effect of Label Availability

The illustrative label-budget comparison shows the proposed framework achieving average precision values of 0.768, 0.861, and 0.903 when 1%, 5%, and 10% of the training windows are labelled, respectively. Corresponding recall values are 68.00%, 82.00%, and 87.00%. The supervised temporal CNN achieves average precision values of 0.548, 0.779, and 0.852, with recall increasing from 49.00% to 76.00% and then to 83.00%. Thus, the hypothetical average-precision advantage of pretraining decreases from 0.220 at the 1% budget to 0.082 at 5% and 0.051 at 10%. At the smallest budget, the proposed model identifies 340 of the 500 faulty test windows, compared with 245 for the supervised CNN. At the largest budget, these counts increase to 435 and 415. The example therefore depicts the greatest benefit when labelled observations are most restricted, followed by a smaller difference as supervised training receives more annotated data.

This pattern would be consistent with pretrained representations providing an informative initialization when supervised examples are insufficient to learn temporal features reliably. It would not demonstrate that pretraining eliminates the need for fault labels. Even in this favourable example, the proposed model misses 160 faulty windows at the 1% budget, compared with 90 at 5% and 65 at 10%. Furthermore, average precision at the 5% budget is slightly higher for the proposed model than for the supervised CNN at the 10% budget, but their recall values are 82.00% and 83.00%, respectively. Consequently, these hypothetical values would support a narrow statement about ranking performance rather than a general claim that half as many labels provide equivalent detection. Actual label-efficiency conclusions must also account for labelled fault counts, independent event coverage, validation annotation, and variability across sampled subsets. Percentages alone are insufficient when small changes in the selected examples can determine whether a fault category is represented at all. Table 4 illustrates the influence of training label availability.

**Table 4:** Illustrative influence of training label availability

| Labelled Training Budget (%) | Supervised CNN Average Precision | Proposed Framework Average Precision | Supervised CNN Recall (%) | Proposed Framework Recall (%) | Faulty Windows Detected by Proposed Framework |
|---|---|---|---|---|---|
| 1 | 0.548 | 0.768 | 49 | 68 | 340 of 500 |
| 5 | 0.779 | 1 | 76 | 82 | 410 of 500 |
| 10 | 0.852 | 0.903 | 83 | 87 | 435 of 500 |

All comparisons assume the same held-out test set and validation-based threshold-selection procedure. Actual labelled training counts remain to be determined from the selected dataset. Figure 8 and figure 9 shows the effect of label availability on average precision and recall respectively.

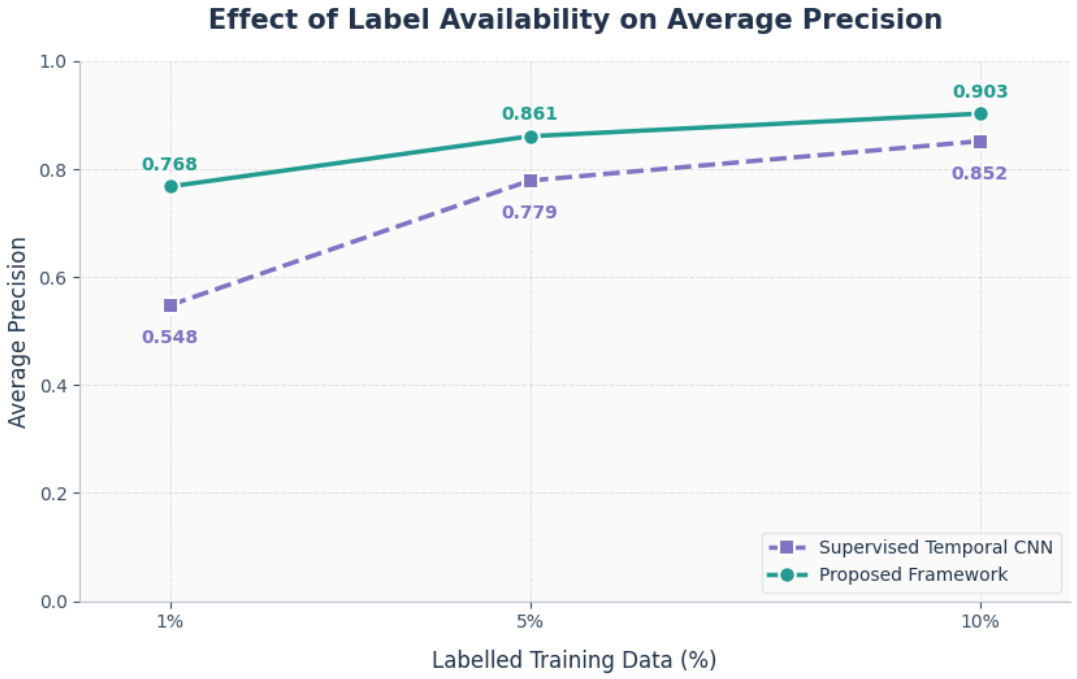


**Figure 8:** Effect of Label Availability on Average Precision

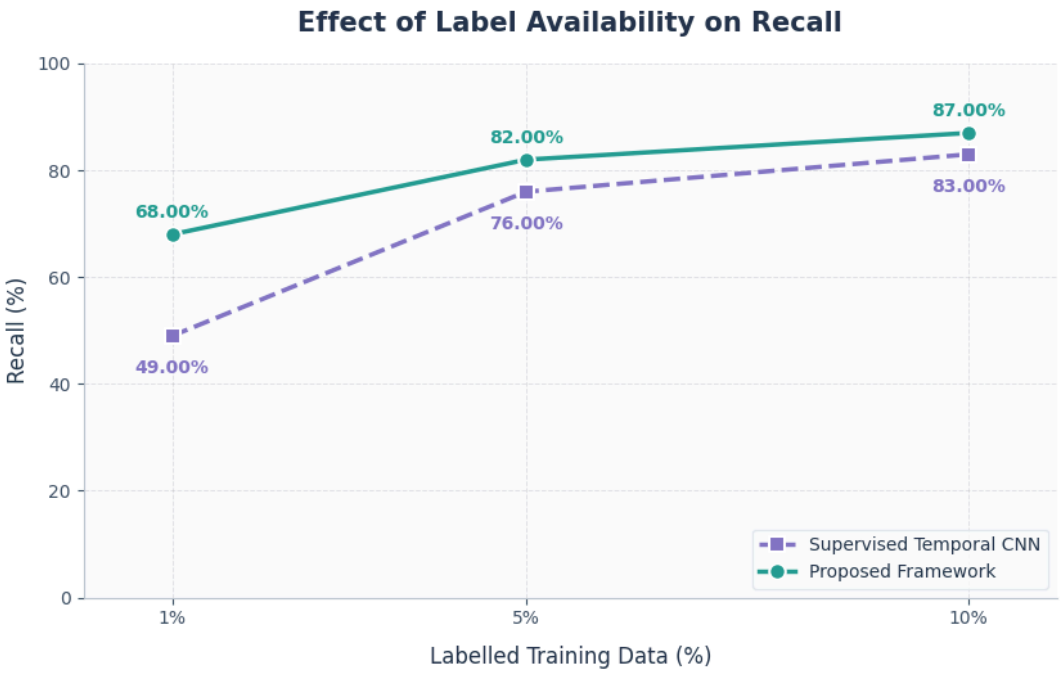


**Figure 9:** Effect of Label Availability on Recall

### 5.3. Ablation Analysis and Practical Implications

The illustrative ablation compares three temporal-model configurations at the 5% training label budget. Training the encoder and classifier from scratch produces an average precision of 0.779 and recall of 76.00%. Using the pretrained encoder as a frozen feature extractor increases these values to 0.832 and 78.00%, respectively. Fine-tuning the pretrained encoder together with the classifier produces the proposed model's average precision of 0.861 and recall of 82.00%. In this example, frozen representations identify 390 faulty windows, compared with 380 for training from scratch and 410 for full fine-tuning. The difference between the frozen and fine-tuned configurations illustrates the potential contribution of adapting the representation to the labelled detection task. However, the comparison does not isolate the effects of individual augmentations, encoder depth, or contrastive temperature. Those factors would require additional controlled experiments before their contributions could be claimed.

For computational illustration, suppose the proposed detector requires 2.4 ms for preprocessing, 4.8 ms for encoder execution, 0.2 ms for classification, and 0.1 ms for threshold and alert-rule computation. The total processing time is therefore 7.5 ms per completed window. With a sampling frequency of 100 Hz, a window length of 256 samples, and a stride of 64 samples, each window spans 2.55 s and successive windows become available every 640 ms. The hypothetical processing time is approximately 1.17% of this output interval, suggesting sufficient processing capacity on the assumed execution platform. This does not establish a 7.5 ms fault detection delay, because detection also depends on the fault's position within a window and the alert rule. For example, requiring three consecutive positive windows adds two output intervals, or 1.28 s, after the first positive window, assuming subsequent windows are also positive. These values illustrate the need to distinguish computational latency from operational detection delay. Actual deployment conclusions require identified hardware, measured timing distributions, independent event annotations, and explicit reporting of missed faults and false alert episodes.

**Table 5:** Illustrative ablation results at a 5% training label budget

| Model Configuration | Average Precision | Recall (%) | Faulty Windows Detected |
|---|---|---|---|
| Randomly Initialized Encoder with Supervised Training | 0.779 | 76 | 380 of 500 |
| Frozen Pretrained Encoder with Trained Classifier | 0.832 | 78 | 390 of 500 |
| Pretrained Encoder and Classifier Jointly Fine-Tuned | 0.861 | 82 | 410 of 500 |

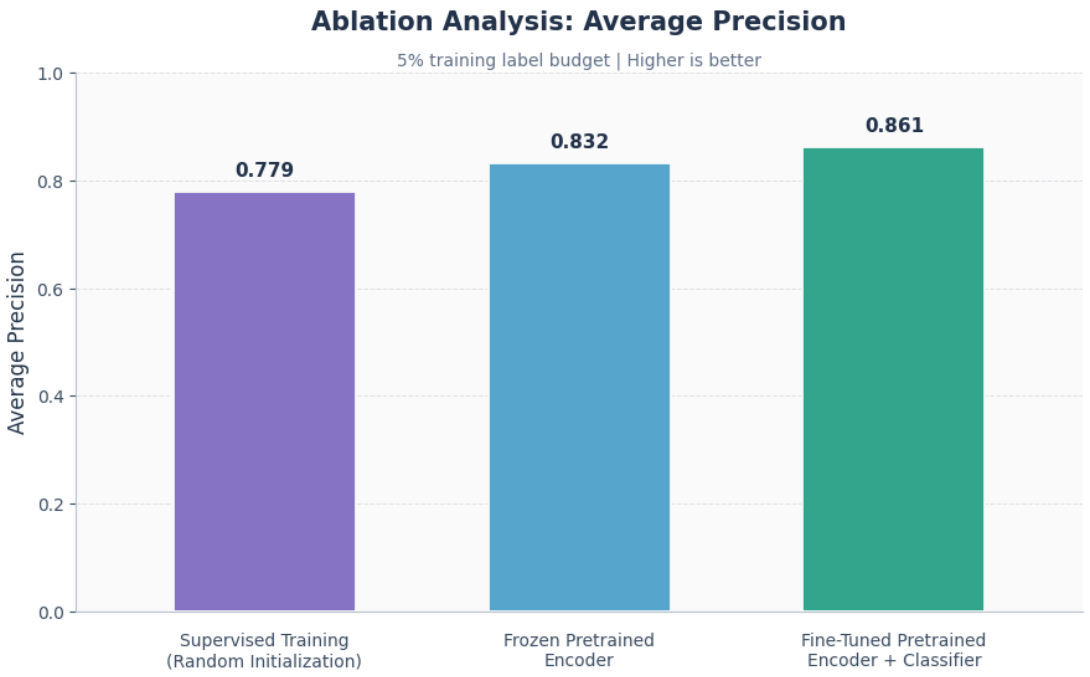


**Figure 10:** Ablation Analysis: Average Precision

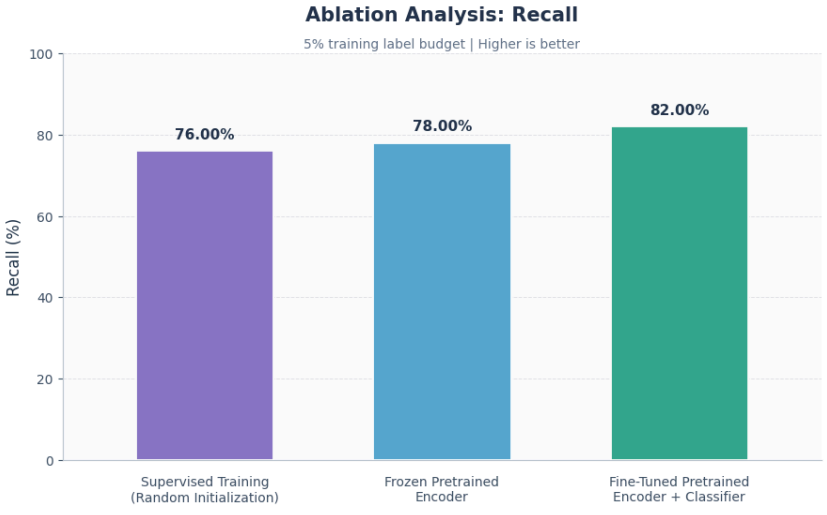


**Figure 11:** Ablation Analysis: Recall

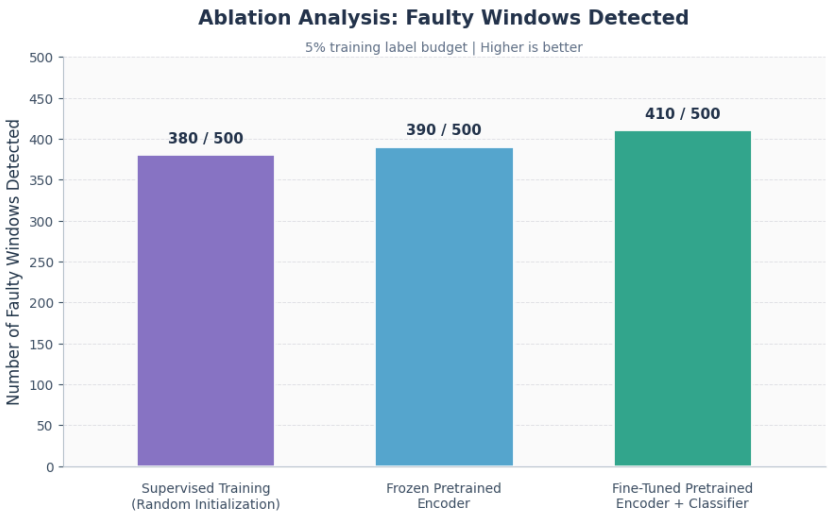


**Figure 12:** Ablation Analysis: Faulty Windows Detected

**Table 6:** Illustrative processing and monitoring configuration.

| Component or Setting | Illustrative Value |
|---|---|
| Preprocessing Time per Window | 2.4 ms |
| Encoder Execution Time per Window | 4.8 ms |
| Classification Time per Window | 0.2 ms |
| Threshold and Alert-Rule Computation Time | 0.1 ms |
| Total Processing Time per Window | 7.5 ms |
| Sensor Sampling Frequency | 100 Hz |
| Window Length | 256 samples |
| Window Stride | 64 samples |
| Elapsed Time Spanned by a Window | 2.55 s |
| Interval Between Completed Windows | 640 ms |
| Additional Persistence Wait for Three Consecutive Positive Windows | 1.28 s |

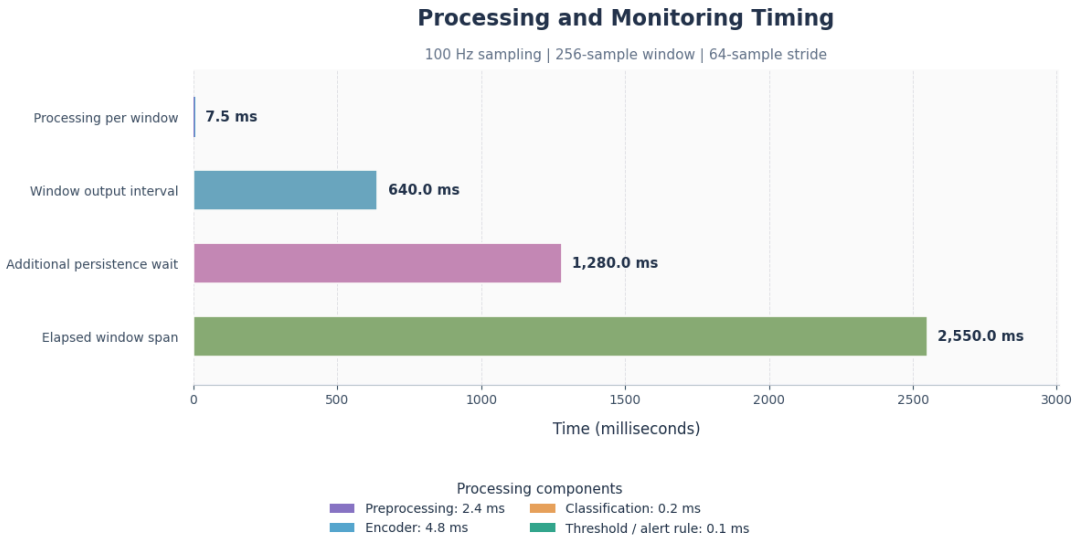


**Figure 13:** Processing and Monitoring Timing

## VI. Conclusion

This paper proposes a self-supervised learning framework for rare fault detection in Industrial Internet of Things systems with limited labelled data. The approach combines contrastive pretraining on unlabelled sensor windows with class-weighted supervised fine-tuning, enabling a compact temporal encoder to learn operational patterns before adapting to fault classification. The illustrative analysis demonstrates how its potential benefits can be assessed through comparisons with supervised temporal learning, Random Forest classification, and autoencoder-based anomaly detection across different label budgets. However, the example values do not constitute experimental evidence, and improvements in detection performance or label efficiency remain to be established through reproducible experiments. The framework emphasizes independent data partitioning, validation-based threshold selection, and separate assessment of classification performance, false alerts, computational latency, and event detection delay. Its practical value will depend on whether these evaluations confirm reliable fault sensitivity across equipment and operating conditions while maintaining manageable annotation and processing requirements.

## VII. Future Work

Future work will focus on validating the proposed framework using industrial datasets and operational equipment with documented fault events, varying loads, and realistic class distributions. Evaluation across independent machines and operating conditions will establish whether pretrained representations remain useful beyond the training environment. Further studies will examine physically informed augmentations and complementary time–frequency objectives to preserve subtle fault signatures during representation learning. Online adaptation may be investigated to accommodate changes in equipment behaviour, with safeguards against learning faults as normal operation. Active learning could help direct expert annotation towards informative and uncertain observations, while open-set detection could extend the framework to previously unseen fault categories. Model compression and quantization will be explored for deployment on resource-constrained edge devices. These extensions should be assessed through repeated experiments that report annotation requirements, uncertainty, false alerts, missed events, and detection delay alongside classification performance.

## References

[1] A. K. S. Jardine, D. Lin, and D. Banjevic, “A review on machinery diagnostics and prognostics implementing condition-based maintenance,” Mechanical Systems and Signal Processing, vol. 20, no. 7, pp. 1483–1510, 2006, doi: 10.1016/j.ymssp.2005.09.012.

[2] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, “Deep learning and its applications to machine health monitoring,” Mechanical Systems and Signal Processing, vol. 115, pp. 213–237, 2019, doi: 10.1016/j.ymssp.2018.05.050.

[3] H. He and E. A. Garcia, “Learning from imbalanced data,” IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 9, pp. 1263–1284, Sep. 2009, doi: 10.1109/TKDE.2008.239.

[4] T. Saito and M. Rehmsmeier, “The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets,” PLOS ONE, vol. 10, no. 3, Art. no. e0118432, 2015, doi: 10.1371/journal.pone.0118432.

[5] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, “Unsupervised scalable representation learning for multivariate time series,” in Advances in Neural Information Processing Systems, vol. 32, 2019, pp. 4650–4661.

[6] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in Proc. 37th International Conference on Machine Learning, vol. 119, 2020, pp. 1597–1607.

[7] E. Eldele, M. Ragab, Z. Chen, M. Wu, C. K. Kwoh, X. Li, and C. Guan, “Time-series representation learning via temporal and contextual contrasting,” in Proc. 30th International Joint Conference on Artificial Intelligence, 2021, pp. 2352–2359, doi: 10.24963/ijcai.2021/324.

[8] Z. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, Y. Tong, and B. Xu, "TS2Vec: Towards universal representation of time series," in Proc. AAAI Conference on Artificial Intelligence, vol. 36, no. 8, 2022, pp. 8980–8987, doi: 10.1609/aaai.v36i8.20881.

[9] B. K. Iwana and S. Uchida, "An empirical survey of data augmentation for time series classification with neural networks," PLOS ONE, vol. 16, no. 7, Art. no. e0254841, 2021, doi: 10.1371/journal.pone.0254841.

[10] Y. Lei, B. Yang, X. Jiang, F. Jia, N. Li, and A. K. Nandi, "Applications of machine learning to machine fault diagnosis: A review and roadmap," Mechanical Systems and Signal Processing, vol. 138, Art. no. 106587, 2020, doi: 10.1016/j.ymssp.2019.106587.

[11] T. Ince, S. Kiranyaz, L. Eren, M. Askar, and M. Gabbouj, "Real-time motor fault detection by 1-D convolutional neural networks," IEEE Transactions on Industrial Electronics, vol. 63, no. 11, pp. 7067–7075, Nov. 2016, doi: 10.1109/TIE.2016.2582729.

[12] W. Zhang, G. Peng, C. Li, Y. Chen, and Z. Zhang, "A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals," Sensors, vol. 17, no. 2, Art. no. 425, 2017, doi: 10.3390/s17020425.

[13] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," Neural Computation, vol. 13, no. 7, pp. 1443–1471, 2001, doi: 10.1162/089976601750264965.

[14] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in Proc. 8th IEEE International Conference on Data Mining, 2008, pp. 413–422, doi: 10.1109/ICDM.2008.17.

[15] L. Ruff et al., "Deep one-class classification," in Proc. 35th International Conference on Machine Learning, vol. 80, 2018, pp. 4393–4402.

[16] [16] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding Gaussian mixture model for unsupervised anomaly detection," in Proc. International Conference on Learning Representations, 2018.

[17] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semi-supervised learning with deep generative models," in Advances in Neural Information Processing Systems, vol. 27, 2014, pp. 3581–3589. proceedings.

[18] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, "Semi-supervised learning with Ladder Networks," in Advances in Neural Information Processing Systems, vol. 28, 2015, pp. 3546–3554.

[19] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in Advances in Neural Information Processing Systems, vol. 30, 2017, pp. 1195–1204.

[20] K. Sohn et al., "FixMatch: Simplifying semi-supervised learning with consistency and confidence," in Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 596–608.

[21] S. J. Pan and Q. Yang, "A survey on transfer learning," IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345–1359, Oct. 2010, doi: 10.1109/TKDE.2009.191.

[22] Y. Ganin et al., "Domain-adversarial training of neural networks," Journal of Machine Learning Research, vol. 17, no. 59, pp. 1–35, 2016.

[23] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in Advances in Neural Information Processing Systems, vol. 29, 2016, pp. 3630–3638.

[24] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in Advances in Neural Information Processing Systems, vol. 30, 2017, pp. 4077–4087.

[25] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," arXiv preprint arXiv:1807.03748, 2018, doi: 10.48550/arXiv.1807.03748.

[26] S. Tonekaboni, D. Eytan, and A. Goldenberg, “Unsupervised representation learning for time series with temporal neighborhood coding,” in Proc. International Conference on Learning Representations, 2021.

[27] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, “A transformer-based framework for multivariate time series representation learning,” in Proc. 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 2114–2124, doi: 10.1145/3447548.3467401.

[28] X. Zhang, Z. Zhao, T. Tsiligkaridis, and M. Zitnik, “Self-supervised contrastive pre-training for time series via time-frequency consistency,” in Advances in Neural Information Processing Systems, vol. 35, 2022, pp. 3247–3263.