
# POLITECNICO DI TORINO

*Master's Degree in Mechatronics Engineering*

**Master's Degree Thesis**

# Multi-sensor Channel Classification for Industrial Anomaly Detection at the Edge

**Supervisors:**
Dr. Valentino Peluso

**Candidate:**
Alessandro Giacobetti
S329778

**Abstract**

Industry 4.0 has made predictive maintenance a practical necessity, but the dominant cloud-centric monitoring pipeline is increasingly limited by bandwidth cost, latency, and data confidentiality. This has accelerated a shift toward TinyML, where anomaly detection runs directly on edge microcontrollers. However, deployment on Cortex-M–class devices is constrained by a hard memory ceiling (2 MB), creating a "hardware conflict" for standard reconstruction-based solutions such as autoencoders: the decoder imposes a prohibitive memory tax, and point-wise reconstruction objectives tend to model nuisance detail (e.g., background noise) rather than fault-relevant physics, making detectors brittle under domain shift.

This thesis proposes a decoder-free alternative: a self-supervised classification (SSC) framework for multi-sensor anomaly detection that replaces reconstruction with a discriminative pretext task, *Sensor Channel Identification*. By training a lightweight classifier to predict which sensor channel produced a window (labels available "for free" from acquisition metadata), the model is forced to learn stable channel-specific transfer-function fingerprints from normal data only. At inference, anomalies are scored via *confidence drop* (log-odds of the true channel), enabling detection without anomaly labels and without any decoder.

We validate the approach on the IMAD-DS multi-sensor benchmark under explicit domain shift (speed/load and background-noise changes). The proposed SSC architecture fits the edge budget, reducing parameters from 33.9M to 0.9M ($\sim 38\times$ fewer) and the Int8 peak memory footprint from 34.0 MB to 1.0 MB ($\sim 34\times$ smaller), making it deployable where the reconstruction baseline is structurally infeasible. Crucially, SSC improves robustness: on the stationary brushless-motor case, combined AUROC rises from 59.0% to 98.4%; on the transient robotic-arm case, combined AUROC improves from 91.6% to 95.5%. These results show that robust, high-performance edge anomaly detection can be achieved by replacing reconstruction with an SSC channel-identification objective that is both lightweight and deployment-oriented.

**Keywords:** Anomaly Detection; TinyML; Edge AI; Self-Supervised Learning; Sensor Channel Identification; Domain Shift Robustness; Efficient Deep Learning; Predictive Maintenance.

# Acknowledgments

A chi ha reso possibile questo percorso e ne ha condiviso il cammino.

# Contents

## List of Tables

# List of Figures

# 1 Introduction

## 1.1 Context: The Data Deluge and the Edge

The industrial world is currently undergoing a silent revolution. Under the paradigm of Industry 4.0, factories are transitioning from reactive maintenance, that is, fixing machines only after they break, to *predictive maintenance*, where intelligent systems anticipate failures before they occur [31].

To achieve this, modern industrial assets are blanketed with sensors. High-frequency accelerometers, gyroscopes, and microphones now capture the "heartbeat" of machinery, generating massive streams of vibration and acoustic data. Ideally, this data contains the subtle precursors to failure: a faint rattle in a robotic arm or a harmonic shift in a spinning motor. However, capturing this data has created a new dilemma: where should it be processed?

The traditional approach, often termed "Cloud-Centric", involves streaming all raw sensor data to centralized servers. While computationally powerful, this architecture is increasingly hitting physical limits. Transmitting high-frequency raw waveforms consumes prohibitive network bandwidth, introduces latency that prevents real-time safety stops, and exposes sensitive operational signatures to security risks outside the factory premises [24].

In response, the industry is shifting toward *TinyML*: the deployment of Machine Learning (ML) models directly onto the sensor node itself [5]. By moving the intelligence to the "Edge", smart sensors can process data locally, sending only high-level health status updates to the cloud. This ensures real-time responsiveness, drastically reduces bandwidth costs, and guarantees data confidentiality [1].

## 1.2 Problem Statement: The Hardware Conflict

While the concept of TinyML is compelling, its realization faces a severe **Hardware Conflict**. We are essentially asking for a contradiction: we desire the sophisticated analytical power of deep learning, but we require it to run on microcontrollers that have a fraction of the resources of a smartphone.

Specifically, we target High-Performance Edge nodes (e.g., Cortex-M7 class microcontrollers). While these chips are capable of numerical process-

ing, they are strictly bound by non-volatile storage, typically offering only a few megabytes of on-chip **Flash** with a hard ceiling of **2 MB** for on-device model storage. This creates a physical bottleneck that renders most modern AI architectures structurally undeployable; Chapter 6 quantifies this gap via a direct footprint comparison (Table 3).

This conflict is summarized in Figure 1. The industry standard for detecting anomalies without labels (Self-Supervised Learning) is the **Reconstruction-based** approach, typically using Autoencoders [7].

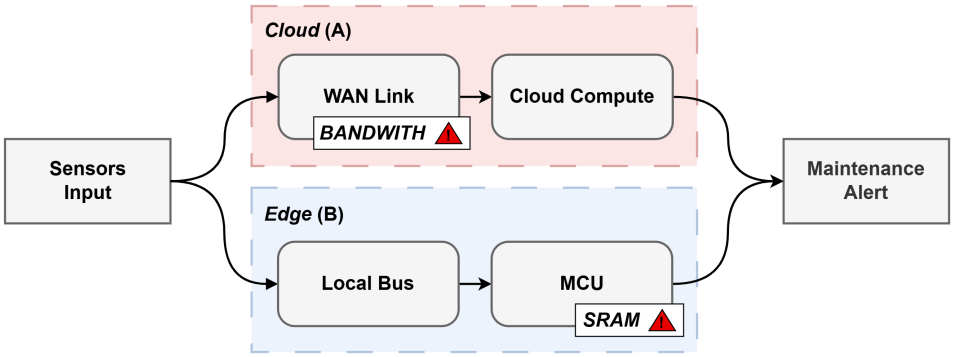



Figure 1: Illustration of the trade-off between cloud-centric processing (A) and microcontroller deployment constraints (B) in industrial monitoring.

Intuitively, these standard models function like a **sketch artist**. The AI tries to learn to draw a perfect picture of the machine's normal sound. If it encounters a sound it cannot draw (reconstruct) effectively, it flags an anomaly. However, this "generative" approach relies on a parameter-heavy component called a *Decoder* to recreate the signal. On the edge, this design creates two practical problems. First, the **Efficiency Gap:** the decoder forces the system to generate a high-dimensional output that is immediately discarded after a simple error check, adding substantial computational overhead without directly improving the anomaly decision (see the feasibility analysis in Chapter 6). Second, the **Semantic Gap:** by trying to reconstruct every detail, the model often wastes capacity learning irrelevant background noise rather than the meaningful physics of the machine, which can make the system brittle to benign environmental changes, such as a factory becoming slightly louder or a motor changing speed [19].

To summarize the architectural shift at a glance, Figure 2 contrasts

the standard reconstruction pipeline with the proposed decoder-free SSC alternative.

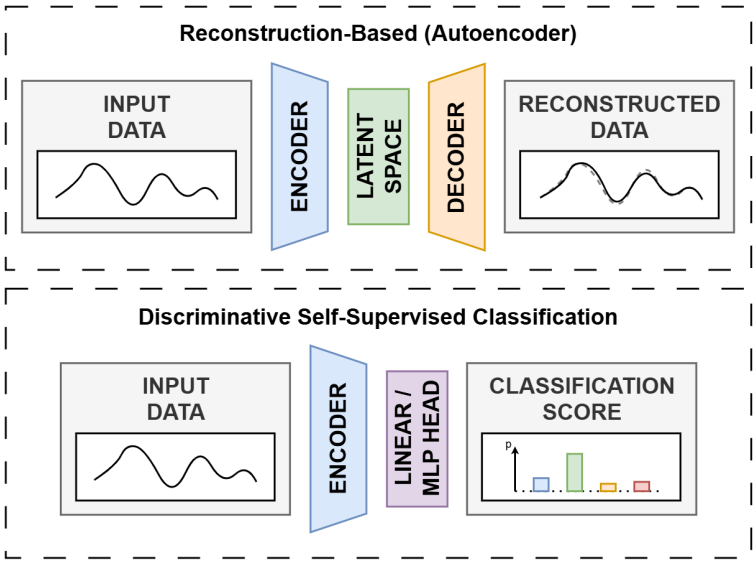



Figure 2: Reconstruction-based autoencoder vs decoder-free self-supervised classification.

### 1.3 Proposed Solution: Identification instead of Reconstruction

This thesis proposes a fundamental shift in how we approach anomaly detection on the edge. We argue that to detect a broken machine, we do not need to be able to "draw" a perfect picture of it. We only need to be able to *recognize* its unique characteristics. We introduce a **Self-Supervised Classification (SSC)** framework that replaces the heavy Autoencoder with a lightweight, decoder-free classifier.

**The Core Insight.** Modern industrial nodes are inherently multi-sensor (e.g., they contain an X-axis accelerometer, a Y-axis accelerometer, and a microphone). We can exploit this hardware fact to train our model. We set the model a pretext task: *"Look at this signal snippet and tell me which sensor channel it came from."*

To distinguish the vibration of the X-axis from the Y-axis on the same

machine chassis, the model is forced to learn the intrinsic physical properties of the signal, such as resonances, damping, and transfer functions. We demonstrate that this discriminative ability translates directly to anomaly detection: when a machine breaks, its physical signature changes, and the model becomes "confused" about the source of the signal.

**Key Contributions.** By discarding the generative decoder in favor of a discriminative objective, we obtain two complementary gains. The first is **Feasibility** (closing the Efficiency Gap): the proposed model reduces the memory footprint by approximately **30**× compared to a standard reconstruction baseline (from $\approx$ 34 MB to $\approx$ 1 MB), enabling deployment within microcontroller-class memory limits; this comparison is reported explicitly in the edge feasibility benchmark (Table 3, Chapter 6). The second is **Improved Detection Performance**: across both stationary and transient machines, the classification-based objective consistently outperforms reconstruction-based baselines under both nominal and shifted operating conditions. This gain can be attributed to the fact that the model is optimized to discriminate stable sensor-channel signatures rather than to reproduce fine-grained signal details, which reduces sensitivity to nuisance variations such as background noise or operating-point changes. These effects are quantified in Chapter 6 (Tables 4–5).

## 1.4 Thesis Structure

The remainder of this thesis is organized as follows: **Chapter 2** provides the technical background on time-frequency analysis and deep learning components. **Chapter 3** reviews the related work, mapping the landscape of industrial anomaly detection. **Chapter 4** details the proposed Self-Supervised Classification method and the mathematical formulation of the pretext task. **Chapter 5** describes the experimental setup and the IMAD-DS benchmark dataset used for validation. **Chapter 6** presents the quantitative results, analyzing both the computational efficiency and the detection performance under domain shift. Finally, **Chapter 7** concludes the work and outlines future research directions for ultra-low-power adaptation.

# 2 Background

This chapter introduces the technical concepts needed to understand the proposed framework. We first review time–frequency representations for industrial audio and vibration, then summarize the deep learning components used in this thesis with an emphasis on memory constraints typical of microcontrollers. Finally, we introduce the formal definitions of anomaly detection and place the proposed method within the broader taxonomy of outlier detection algorithms.

## 2.1 Time–Frequency Representations for Industrial Signals

Industrial monitoring signals (e.g., microphones, accelerometers, gyroscopes) are typically high-rate, noisy, and often non-stationary. A standard approach is to transform raw waveforms into time–frequency representations where mechanically meaningful patterns (harmonics, broadband bursts, periodic impulses) become easier to model.

### 2.1.1 Short-Time Fourier Transform (STFT)

The Short-Time Fourier Transform (STFT) analyzes how frequency content evolves over time by applying a Fourier transform on short, overlapping windows [3].

Unlike the standard Discrete Fourier Transform (DFT), which provides a global frequency summary of the entire signal, the STFT assumes that the signal is "quasi-stationary" over short durations. This allows it to capture transient events and shifting harmonics essential for monitoring dynamic machinery.

Given a discrete-time signal $x[n]$ and a smoothing window function $w[n]$ of length $N$, the STFT is defined as:

$$X(m,k) = \sum_{n=0}^{N-1} x[n+mH]\; w[n]\; e^{-j\frac{2\pi}{N}kn} \tag{1}$$

where $m$ indexes the time frames, $k$ indexes the frequency bins, and $H$ is the hop size (or stride) between successive windows.

**The Time-Frequency Resolution Trade-off.** A critical constraint in STFT analysis is the Heisenberg-Gabor uncertainty principle, which states

that one cannot simultaneously achieve arbitrarily high resolution in both time and frequency. On one hand, **Long Windows ($N \uparrow$)** provide excellent frequency resolution (narrow bins), allowing the separation of closely spaced harmonics (e.g., motor rotation orders). However, they smear distinct temporal events, reducing the ability to localize transient impacts. On the other hand, **Short Windows ($N \downarrow$)** provide excellent temporal localization for transients (e.g., clicks, impacts) but suffer from poor frequency resolution (wide bins). In this work, the window parameters are selected to strike a balance suitable for both stationary vibration and transient mechanical shocks.

**Windowing and Spectral Leakage.** The window function $w[n]$ (typically a Hann or Hamming window) plays a vital role in suppressing *spectral leakage*. Simply slicing the signal (rectangular windowing) introduces sharp discontinuities at the boundaries, creating artificial high-frequency artifacts (sidelobes). Smooth windows taper the signal to zero at the edges, mitigating these artifacts and ensuring a cleaner spectral representation.

**The Spectrogram.** The complex-valued output $X(m, k)$ contains both magnitude and phase information. For standard discriminative tasks, the phase is typically discarded to enforce shift-invariance. The system therefore operates on the **Power Spectrogram**:

$$S(m, k) = |X(m, k)|^2 \tag{2}$$

This 2D representation treats audio as an "image", enabling the use of standard Convolutional Neural Networks (CNNs). However, discarding phase eliminates information regarding the precise shape of the waveform, which motivates the use of dual-branch architectures (like STGram) when waveform fidelity is required.

### 2.1.2 Mel Filterbanks and Log-Mel Spectrograms

The Mel scale is a perceptual scale of pitches judged by listeners to be equal in distance from one another. It mimics the human ear's non-linear perception of sound, effectively compressing frequency resolution at high frequencies and expanding it at low frequencies [25]. This transformation is advantageous because standard spectrograms provide linear frequency resolution, which often allocates excessive dimensionality to high-frequency components that may contain less semantic information for acoustic tasks.

A Mel-spectrogram is generated by applying a bank of overlapping triangular filters to the power spectrogram. These filters are spaced uniformly on the Mel scale, integrating the energy within each frequency band. The conversion between frequency in Hertz ($f$) and the Mel scale ($m$) is typically approximated as:

$$m = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right) \tag{3}$$

Following the filterbank application, logarithmic compression is applied to the feature map. This step mimics the logarithmic nature of human loudness perception (decibels) and compresses the dynamic range of the signal. The resulting Log-Mel spectrogram is defined as:

$$S_{\log}(t, f) = \log(S_{\text{mel}}(t, f) + \epsilon) \tag{4}$$

where $\epsilon$ is a small constant added to ensure numerical stability. Log-Mel spectrograms are widely used in acoustic and vibration monitoring because they provide a compact, perceptually relevant representation with a stable numerical range, making them highly suitable for machine learning classifiers.

#### 2.1.3 Per-Channel Energy Normalization (PCEN) as an Adaptive Compressor

While logarithmic compression reduces dynamic range, it is a static transformation that treats all signals equally, regardless of the background noise level or source distance. Per-Channel Energy Normalization (PCEN) [28, 18] offers an adaptive alternative, functioning as a form of Automatic Gain Control (AGC) inspired by the adaptation mechanisms of the auditory system.

Unlike static compression, PCEN normalizes the instantaneous energy $E(t, f)$ by a smoothed estimate of the local background energy, $M(t, f)$. This background estimate is typically calculated using a first-order infinite impulse response (IIR) filter. The transformation is defined as:

$$\text{PCEN}(t, f) = \left(\frac{E(t, f)}{(\epsilon + M(t, f))^{\alpha}} + \delta\right)^{r} - \delta^{r} \tag{5}$$

Here, the exponent $\alpha \in [0, 1]$ controls the strength of the gain normalization (effectively suppressing stationary background noise), while $r$ applies dynamic range compression to the normalized signal. The bias terms $\epsilon$ and

$\delta$ are included for numerical stability.

In this thesis, PCEN is utilized as a *supporting stabilizer* to ensure feature robustness in highly non-stationary acoustic regimes. As illustrated in Figure 3, standard static compression (Log-Mel) often fails to suppress high-energy background noise, such as fans, leaving transient fault spikes with low contrast. In contrast, PCEN's adaptive gain control normalizes the stationary background floor to near-zero. This significantly enhances the contrast of transient fault impulses, preventing the model from learning noise patterns.

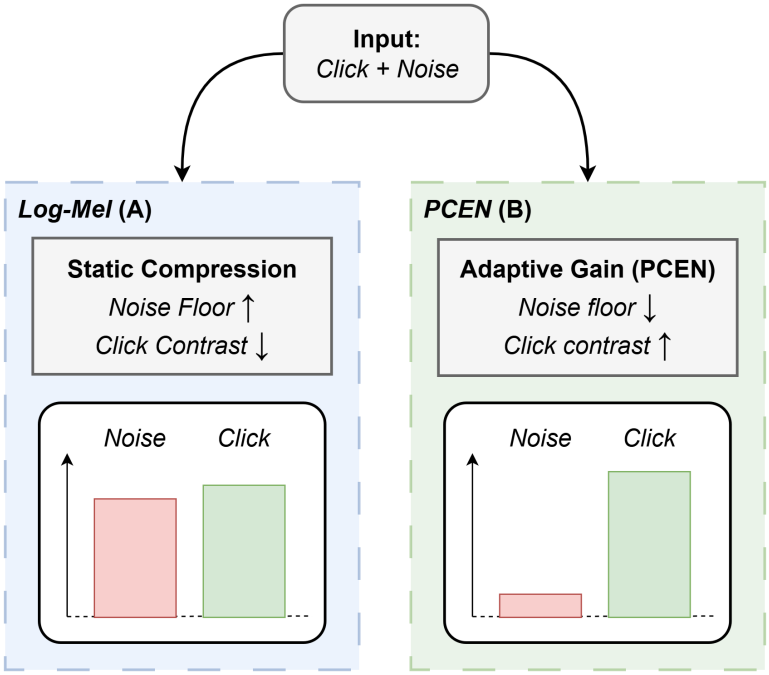



Figure 3: Conceptual comparison of signal representations (Log-Mel vs. PCEN) for transient analysis.

Furthermore, the choice of representation must align with the mechanical regime. Figure 4 compares these features across different machine types. For a Brushless Motor (Top Row), the constant harmonic hum is clearly visible in Log-Mel but is treated as background noise and aggressively suppressed by PCEN, potentially destroying useful signal content. Conversely, for a Robotic Arm (Bottom Row), transient actuation impulses are obscured by

noise in Log-Mel but are significantly enhanced by PCEN. This contrast justifies our regime-specific feature selection: Log-Mel for stationary harmonics and PCEN for transient impulses.

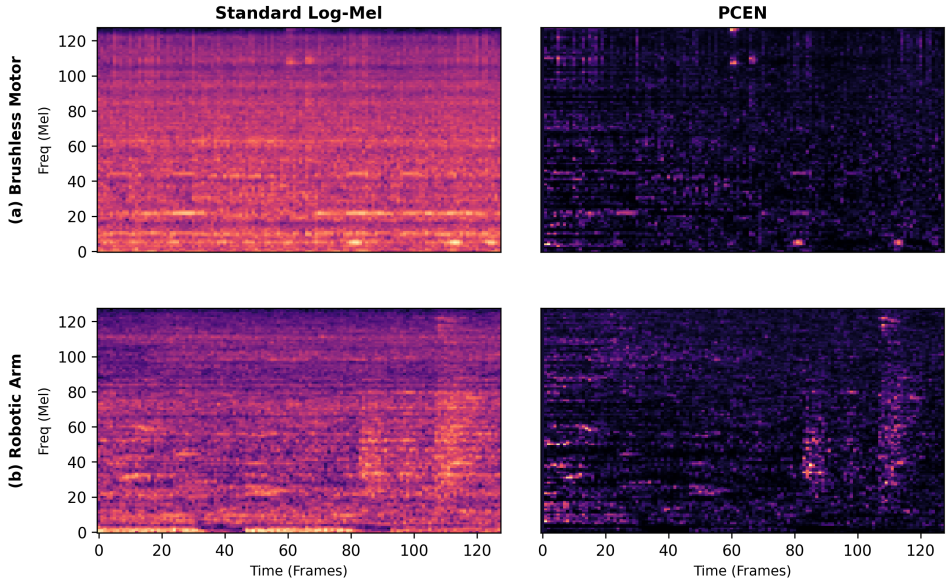



Figure 4: Feature representation comparison for stationary (Brushless Motor) and transient (Robotic Arm) regimes.

### 2.1.4 Spectral–Temporal Fusion and STgram-Style Inputs

Time–frequency features, such as Log-Mel spectrograms or PCEN, are efficient for capturing energy distributions over wider windows. However, they inherently suffer from the time-frequency resolution trade-off imposed by the Short-Time Fourier Transform (STFT) and discard phase information. Consequently, highly *transient* machine behaviors, characterized by onset sharpness, fine temporal textures, or phase-dependent impulses, may be smoothed out or lost in the spectral conversion.

To address this, a practical strategy is to fuse two complementary views of the same signal: a 2D time–frequency representation capturing global spectral structures, and a lightweight 1D temporal branch operating directly on the raw waveform to preserve fine-grained temporal cues.

A representative family of such approaches is **STgram**-style fusion [17].

In this architecture, a 1D Convolutional Neural Network (CNN) extracts high-frequency temporal features from the raw signal, while a parallel 2D CNN processes the spectrogram. The outputs are typically fused via concatenation in a shared latent space, allowing the model to learn correlations between spectral anomalies and their temporal characteristics.

## 2.2 Deep Learning Components and Edge Constraints

Deep learning models operating on time–frequency features rely heavily on convolutional backbones due to their parameter efficiency and ability to exploit local substructures. However, deployment on microcontrollers (edge deployment) fundamentally alters the definition of efficiency. In this thesis, **model size (storage)** is the primary feasibility constraint and is bounded by a hard **2 MB Flash** ceiling for on-device deployment. Consequently, architectures must optimize not just for accuracy, but also for compact parameterization to satisfy this storage limit.

### 2.2.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) [16, 40, 41] process data via learnable kernels applied across spatial dimensions. In the context of spectrogram analysis, they offer distinct advantages over dense architectures. First, they provide **Translation Invariance**, allowing the detection of fault signatures (e.g., a bearing click or frequency shift) regardless of where they occur in the time or frequency domain. Second, they excel at **Local Pattern Extraction**, as mechanical faults typically manifest as localized spectral textures or transient bursts rather than global changes, matching the receptive field logic of convolutional kernels. Finally, they ensure **Parameter Efficiency**; by sharing weights across the input space, CNNs significantly reduce the storage footprint required compared to fully connected networks.

### 2.2.2 Residual Networks (ResNets)

As network depth increases to capture more complex abstract features, standard CNNs often suffer from performance degradation due to the vanishing gradient problem.

This phenomenon occurs during backpropagation, where gradients are computed using the chain rule. As the gradient is propagated backward through many layers, the repeated multiplication of small partial derivatives causes the gradient signal to exponentially decay toward zero. Consequently,

the weights in the initial layers stop updating, preventing the network from learning low-level features effectively.

Residual Networks (ResNets) [13] mitigate this by introducing identity skip connections, allowing the network to learn residual functions. Formally, a building block is defined as $y = \mathcal{F}(x, \{W_i\}) + x$, where $x$ and $y$ are the input and output vectors of the layers considered, and $\mathcal{F}$ represents the residual mapping. This structure eases the optimization of deep networks, ensuring gradients can propagate effectively during training. In industrial monitoring, ResNet-style backbones (e.g., ResNet-8 or ResNet-18) are particularly attractive because they balance hierarchical feature extraction with a controlled computational budget, providing high accuracy without the prohibitive memory costs of wider or densely connected architectures.

### 2.2.3 Normalization Layers: BatchNorm vs InstanceNorm

Normalization layers are a critical component of modern deep learning architectures. By standardizing the input to each layer to have zero mean and unit variance, they reduce internal covariate shift, allowing for higher learning rates and faster convergence. However, different normalization strategies encode different inductive biases regarding the data distribution, which becomes significant in the context of domain generalization.

**Batch Normalization (BN).** Batch Normalization [14] computes statistics across the batch dimension ($N$). For a feature map $x \in \mathbb{R}^{N \times C \times H \times W}$, BN normalizes each channel $c$ independently using the mean $\mu_c$ and variance $\sigma_c^2$ aggregated over the entire mini-batch and spatial locations $(N, H, W)$:

$$\hat{x}_{n,c,h,w} = \frac{x_{n,c,h,w} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}} \cdot \gamma_c + \beta_c \tag{6}$$

During inference, BN uses running statistics accumulated during training.

**Implication for Anomaly Detection:** BN assumes that the test data follows the same global statistics as the training data. If the deployment environment has a shifted noise floor or amplitude (e.g., a louder background), the fixed running mean/variance will mismatch, potentially causing false alarms. BN is thus best suited for **stationary** regimes where operating conditions are stable.

**Instance Normalization (IN).** Instance Normalization [26], originally developed for style transfer, computes statistics for each sample independently. The mean $\mu_{n,c}$ and variance $\sigma^2_{n,c}$ are calculated only across the spatial dimensions $(H, W)$ of the specific sample $n$:

$$\hat{x}_{n,c,h,w} = \frac{x_{n,c,h,w} - \mu_{n,c}}{\sqrt{\sigma^2_{n,c} + \epsilon}} \tag{7}$$

**Implication for Anomaly Detection:** By normalizing each spectrogram based on its own statistics, IN effectively filters out global amplitude shifts and constant offsets (acting as a dynamic high-pass filter for "style"). This makes the model invariant to global gain changes or varying sensor calibration, rendering it highly effective for **transient** or non-stationary environments where absolute energy levels may fluctuate benignly.

### 2.2.4 Data Augmentation as Lightweight Regularization

In self-supervised learning, the pretext task must be challenging enough to force the model to learn semantic features rather than memorizing low-level artifacts (e.g., specific noise patterns). Data augmentation serves as a regularization mechanism, injecting invariance into the training process without increasing the inference-phase parameter count.

**Mixup (Input Space Interpolation)** [30] generates virtual training examples by computing convex combinations of pairs of inputs and their corresponding labels. Given two samples $(x_i, y_i)$ and $(x_j, y_j)$, a new sample is constructed as:

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j, \quad \tilde{y} = \lambda y_i + (1 - \lambda) y_j \tag{8}$$

where $\lambda \sim \text{Beta}(\alpha, \alpha)$. This encourages the model to behave linearly between classes, smoothing the decision boundaries. In the context of industrial monitoring, this prevents the model from becoming overconfident on sharp spectral peaks that may drift slightly due to speed variations.

Complementarily, **SpecAugment (Spectrogram Masking)** [20], originally designed for speech recognition, operates directly on the time-frequency representation. It applies random masks to contiguous blocks of time frames (time masking) and frequency channels (frequency masking). This effectively simulates partial signal occlusion, ensuring the classifier learns robust, distributed representations rather than relying on a single "shortcut" feature.

### 2.2.5 The Feasibility Constraint: 2 MB Flash

In the context of TinyML, hardware constraints manifest primarily as a storage bottleneck: a feasible model must fit within the available on-chip Flash memory.

**Storage Constraint (Flash Memory).** The non-volatile memory stores the model's fixed parameters (weights), which sets a hard upper bound on model complexity. For standard Cortex-M7 microcontrollers, the available Flash typically ranges from 1 MB to 2 MB; in this thesis, we adopt a hard ceiling of **2 MB** for on-device model storage [45]. Any architecture exceeding this size is structurally undeployable regardless of its runtime efficiency.

## 2.3 Anomaly Detection Setting and Taxonomy

Industrial anomaly detection is a specific instance of the broader outlier detection problem. It typically operates under a *semi-supervised* or *unsupervised* assumption (often termed "One-Class" learning), where abundant data is available for the "normal" class ($\mathcal{D}_{\text{normal}}$), while anomalous data is rare, expensive, or completely absent during training [7].

The goal is to learn a decision function $f(x)$ or a scoring function $s(x)$ that assigns high values to samples deviating from the learned normality distribution.

$$\text{State}(x) = \begin{cases} \text{Anomalous} & \text{if } s(x) > \tau \\ \text{Normal} & \text{otherwise} \end{cases} \tag{9}$$

### 2.3.1 Types of Anomalies

To design an effective detection system, it is crucial to categorize the nature of the anomalies expected in the signal [32]. The most fundamental category is **Point Anomalies**, where an individual data instance is considered anomalous with respect to the rest of the data distribution. In our context, this corresponds to global shifts in behavior, such as a motor continuously vibrating at a forbidden frequency due to misalignment.

In contrast, **Contextual Anomalies** occur when a data instance is anomalous only within a specific context. For example, a sudden high-energy impulse might be normal if it corresponds to a robotic actuation (context: "moving"), but anomalous if it occurs during an idle period (context: "waiting"). Detecting these conditional faults requires models capable

of capturing temporal dependencies and contextual state, rather than evaluating frames in isolation.

### 2.3.2 Domain Shift

A central challenge in real-world deployment is domain shift, formally defined as the scenario where the joint probability distribution of the training data (Source domain, $\mathcal{D}_S$) differs from that of the testing data (Target domain, $\mathcal{D}_T$), i.e., $P_S(X) \neq P_T(X)$ [6]. In industrial environments, this non-stationarity is pervasive; machines frequently undergo changes in operating conditions such as variable rotational speeds, fluctuating loads, environmental noise, or mechanical settling over time.

For an anomaly detector to be practically viable, it must exhibit *selective invariance*. It must remain robust to these benign shifts (covariate shift) to prevent false alarms, while maintaining high sensitivity to true structural faults (concept shift). This thesis explicitly evaluates this robustness using the IMAD-DS dataset [2], which segregates data into distinct Source and Target conditions to simulate realistic deployment gaps.

### 2.3.3 Discriminative Self-Supervised Classification (SSC)

Domain shift also motivates training objectives that avoid reconstructing low-level signal statistics. To circumvent the limitations of reconstruction-based approaches, recent works have explored discriminative self-supervised methods. Instead of reconstructing the input, the model is trained on a surrogate (pretext) classification task where labels are automatically derived from the data structure [11].

This paradigm offers three decisive advantages for edge deployment. Architecturally, it is **Decoder-Free**: the model requires only an encoder and a lightweight head, drastically reducing memory usage compared to autoencoders. Semantically, the classification objective forces the network to learn **robust features** rather than pixel-level noise statistics. Finally, it enables **Uncertainty-Based Scoring**, where anomalies are detected by a drop in classification confidence (high entropy) when the input violates the learned pretext rules. This thesis adopts the SSC paradigm, specifically leveraging *Sensor Channel Identification*, to enable deployment on Cortex-M microcontrollers.

### 2.3.4 Taxonomy of Approaches

The literature on anomaly detection is vast, spanning from classical statistics to modern deep learning. We can broadly categorize the standard approaches into three families:

**1. Probabilistic and Density Estimation.** These methods explicitly estimate the probability density function (PDF) of the normal data, $P(x)$. Anomalies are identified as samples lying in low-density regions. Classic examples include **Gaussian Mixture Models (GMM)** and **Kernel Density Estimation (KDE)**. While effective for low-dimensional data, these methods often struggle with high-dimensional inputs (like spectrograms), since distance metrics lose discriminative power.

**2. One-Class Classification (Boundary Methods).** Rather than estimating the full density, these methods map the data into a feature space and attempt to find a compact boundary that encloses the normal samples. Prominent algorithms include the **One-Class SVM (OC-SVM)** [33], which separates data from the origin with a hyperplane, and **Support Vector Data Description (SVDD)** [34], which encloses data within a hypersphere. While powerful, scaling kernel methods to the massive datasets typical of high-frequency sensor streams remains computationally challenging.

**3. Reconstruction-Based (Generative) Models.** This family, dominating the recent deep learning literature, relies on the hypothesis that a model trained to compress and reconstruct normal data will fail to reconstruct anomalies. **Autoencoders (AEs)** [21] are the standard archetype. The anomaly score is defined as the reconstruction error, $\|x - \hat{x}\|^2$. However, as discussed in Chapter 1, this approach imposes a significant "Decoder Tax." The requirement to generate a high-dimensional output introduces substantial computational overhead and inflates the parameter footprint, creating a structural conflict with a strict on-device storage ceiling (2 MB Flash), which motivates decoder-free discriminative alternatives such as SSC.

# 3 Related Works

This chapter maps the research landscape around industrial anomaly detection (IAD). We first review the broader paradigm of Self-Supervised Learning (SSL), then discuss its specific applications in industrial monitoring. We subsequently broaden the scope to examine efforts in architectural optimization and feature selection, which attempt to mitigate computational costs within existing paradigms. Then, we analyze the dominant reconstruction-based approaches (Autoencoders), highlighting their structural incompatibility with edge constraints. Finally, we narrow our focus to discriminative approaches, specifically "Machine ID" classification, which serves as the direct precursor to our proposed method.

## 3.1 Self-Supervised Learning (SSL)

Self-Supervised Learning (SSL) has revolutionized representation learning by enabling models to learn rich features from unlabeled data. The core principle is to define a *pretext task*, that is, a surrogate problem where the supervisory labels are automatically generated from the data structure itself, such that solving it requires the model to capture semantic properties of the input [38].

### 3.1.1 General Paradigms

In the computer vision and audio domains, SSL typically manifests in two primary forms. The first is **Contrastive Learning**, where methods like **SimCLR** [35] and **MoCo** [36] learn representations by maximizing the similarity between differently augmented views of the same sample (positive pairs) while minimizing similarity with other samples (negative pairs). While highly effective, these methods often require large batch sizes or extensive memory banks to maintain negative sample diversity, which can be computationally expensive.

The second form relies on **Pretext Tasks**, such as Transformation Prediction. A more direct approach involves predicting applied transformations. For instance, Golan et al. [37] demonstrated that training a network to predict the rotation angle ($0°, 90°, 180°, 270°$) applied to an image forces it to learn high-level semantic features (e.g., "up vs. down") that are highly effective for anomaly detection. This "classification-based" SSL is particularly relevant to our work as it is inherently decoder-free.

## 3.2 SSL for Industrial Anomaly Detection

In the specific context of Industrial Anomaly Detection (IAD), SSL is the standard operating mode due to the scarcity of labeled fault data. Recent literature has largely focused on adapting contrastive and geometric principles to time-series and audio data.

### 3.2.1 Contrastive Audio Representation

State-of-the-art approaches often adapt contrastive learning to the spectral domain. For example, **CLP-SCF** [12] employs a machine-ID based contrastive learning pretraining phase to disentangle machine-specific features from environmental noise. Similarly, **GRLNet** [23] combines regional and local adversarial learning to capture long-term dependencies in audio. While these methods achieve high detection rates on benchmarks like DCASE [15], they typically rely on heavy backbones (e.g., ResNet-50 or Vision Transformers) [13, 9] to extract sufficiently robust embeddings for the contrastive loss, making them difficult to deploy on microcontrollers.

### 3.2.2 Distribution Modeling

Another branch of SSL for IAD focuses on normalizing flows and density estimation. Methods like **Glow-Aff** [8] use flow-based models to learn the exact likelihood of normal data. While theoretically elegant, these models require reversible architectures that maintain high dimensionality throughout the network depth, preventing the spatial downsampling that makes CNNs efficient.

## 3.3 Architectural Efficiency and Input Representation

Parallel to the development of new learning paradigms, significant research has focused on optimizing the deployment pipeline itself [1, 10]. These efforts can be broadly categorized into architectural modifications and preprocessing strategies designed to maximize efficiency on constrained hardware.

### 3.3.1 Efficient Neural Architectures

To address the computational bottleneck of deep learning on the edge, several works have proposed utilizing lightweight convolutional backbones. Architectures such as **MobileNetV2** [22], **EfficientNet** [44], and compact

audio-specific residual designs (e.g., **ERANNs**) [27] utilize depthwise separable convolutions or carefully constrained residual blocks to drastically reduce the parameter count and FLOPs (Floating Point Operations) required for inference, often achieving performance comparable to heavier ResNet models. In the specific domain of vibration analysis, researchers have also explored ad-hoc topologies, such as **1D-CNNs** [43]. These models operate directly on temporal sequences with compact 1D kernels, offering an extremely low memory footprint suitable for real-time monitoring on microcontrollers, albeit sometimes at the cost of the rich frequency-domain features captured by 2D spectrogram analysis.

### 3.3.2 Input Representation Strategies

Complementary to architectural search, the choice of input representation remains a subject of active debate. While time-frequency representations (e.g., Mel-spectrograms) are the standard for audio anomaly detection, they impose a fixed resolution governed by the STFT parameters. To bypass this limitation, end-to-end learning approaches [42, 28] propose feeding raw waveforms directly into the neural network, allowing the model to learn its own optimal filterbanks. However, this often results in high-dimensional input layers that strain the input buffers of embedded devices. Conversely, other works focus on enhancing the spectrogram itself, using techniques like PCEN [18] or wavelet transforms [29] to improve signal-to-noise ratios before the data reaches the classifier. Our work builds upon these insights by adopting a hybrid approach (STGram) that fuses efficient spectral features with raw temporal cues.

## 3.4 Reconstruction-Based Approaches (The Standard)

Despite the improvements in architecture and preprocessing, the industrial standard for deployment remains the **Reconstruction-based** paradigm. The fundamental premise relies on the *reconstruction hypothesis*: a model trained exclusively on normal data learns to project input samples onto a low-dimensional "normality manifold". Consequently, when presented with an anomalous sample, the model fails to reconstruct it, yielding a high error [7].

### 3.4.1 Autoencoders and Their Limitations

**Autoencoders (AEs)** [21] and their variants (e.g., Variational AEs, USAD [4]) minimize a pixel-wise error objective ($\mathcal{L} = \|x - \hat{x}\|^2$). While intu-

itive, this approach faces two fundamental limitations when deployed on constrained embedded systems.

**The Decoder Tax.** In a standard Autoencoder, the model compresses the input into a low-dimensional latent vector $z$, only to expand it back to the original input dimension $\hat{x}$ via a decoder. Although the latent representation is compact, the decoder adds substantial overhead in both computation and parameterization to regenerate a high-dimensional output. This imposes a structural inefficiency: resources are spent generating high-fidelity data that is immediately discarded after computing a single scalar error value.

**Semantic Misalignment.** Reconstruction losses operate at the pixel or spectral bin level. This incentivizes the model to allocate capacity to modeling high-frequency nuisance details (e.g., background noise texture) rather than abstract mechanical states. As noted by Micheleri et al. [19], this often results in models that are brittle to benign domain shifts, where a change in noise floor causes a spike in error (false positive).

## 3.5 Discriminative Approaches and Machine ID

To circumvent the "Decoder Tax" and the focus on pixel-level details, recent research has pivoted towards **Discriminative Self-Supervised Classification (SSC)**. Instead of generating data, these models are trained to classify metadata labels associated with the normal operating conditions.

### 3.5.1 Deep SVDD and One-Class Classification

Foundational work by Ruff et al. [39] introduced Deep Support Vector Data Description (Deep SVDD), which trains a network to map normal data into a compact hypersphere. While efficient, it often suffers from "hypersphere collapse" (mapping all data to a single point) unless carefully regularized, and typically requires a pre-training phase (e.g., using an Autoencoder) to initialize the weights, introducing a substantial layer of complexity to the design phase.

### 3.5.2 Machine-ID Classification

The most direct precursor to this thesis is the **Machine-ID Classification** framework proposed by Giri et al. [11]. In this setting, the pretext task is to identify which specific machine instance (e.g., "Pump A" vs. "Pump B") produced a given audio segment. During training, the model learns

the specific acoustic signature of each machine instance to minimize cross-entropy loss. Subsequently, during inference, anomalous events distort these learned signatures, causing the model's classification confidence to drop. This approach is highly attractive for the edge because it uses a standard classifier backbone (e.g., MobileNet) without a decoder.

### 3.5.3 The Single-Machine Gap

However, a critical limitation of Machine-ID SSC is its dependence on multi-instance deployment. In many practical industrial scenarios, only a *single* machine is available for training ("fleet-of-one" setting). In such cases, the label set collapses to a single class ($K = 1$), making classification impossible. This limitation motivates our proposed **Sensor Channel Identification** task, which adapts the efficient SSC paradigm to single-machine environments by exploiting the multi-sensor nature of modern industrial nodes.

# 4 Method

This chapter details the proposed **Self-Supervised Classification (SSC)** framework. Our primary objective is to replace the heavy, reconstruction-based architectures standard in industrial anomaly detection with a lightweight, decoder-free classifier that can operate within the strict memory constraints of edge microcontrollers.

The core premise is to shift the detection paradigm: instead of learning to *reconstruct* multi-sensor inputs to detect errors, the model learns to *classify the sensor channel identity* using only normal data and automatically available metadata. By solving this discriminative task, the model implicitly learns the unique physical signatures of the machine's nominal state.

## 4.1 Overview

### 4.1.1 Problem Setup and Notation

We consider a multi-sensor edge node that acquires synchronous signals from $C$ distinct channels (e.g., accelerometer axes $x, y, z$, gyroscope axes $x, y, z$, microphone). Let $\mathbf{x}_{\text{raw}} \in \mathbb{R}^{L \times C}$ denote a multi-sensor segment of length $L$ acquired on-device.

For efficient processing, these raw signals are converted into a time–frequency representation and partitioned into fixed-size windows. Formally, let $x \in \mathbb{R}^{F \times T}$ represent a single-channel input window, where $F$ is the number of frequency bins and $T$ is the temporal width. Let $y \in \{0, \ldots, C-1\}$ denote the **sensor channel ID** associated with $x$. Crucially, this label $y$ is intrinsic to the hardware configuration (e.g., hardwired pinout) and is available as "free" metadata without manual labeling effort.

At inference time, the system produces a scalar anomaly score for each window (and an aggregated score for a monitoring period). Intuitively, the score should remain low for nominal samples and increase when the observed signal deviates from the learned healthy behavior. The concrete scoring rule is defined in Section 4.3.

### 4.1.2 The Self-Supervised Classification Principle

The central contribution of this work is the architectural shift from **Generative Reconstruction** to **Discriminative Classification**. Standard

approaches, such as Autoencoders, detect anomalies by measuring the *reconstruction error* $\|x - \hat{x}\|^2$. In contrast, our framework detects anomalies by measuring the *classification uncertainty*, asking instead: "Does this signal look like it belongs to its claimed sensor channel?"

This is achieved via a pretext task we term **Sensor Channel Identification**. The model is trained to identify which specific sensor channel an input window originates from. To distinguish vibration from the X-axis versus the Y-axis on the same machine chassis, the network is forced to internalize stable channel-specific features, or "transfer-function fingerprints", including resonance modes, spectral coloring, and noise-floor characteristics.

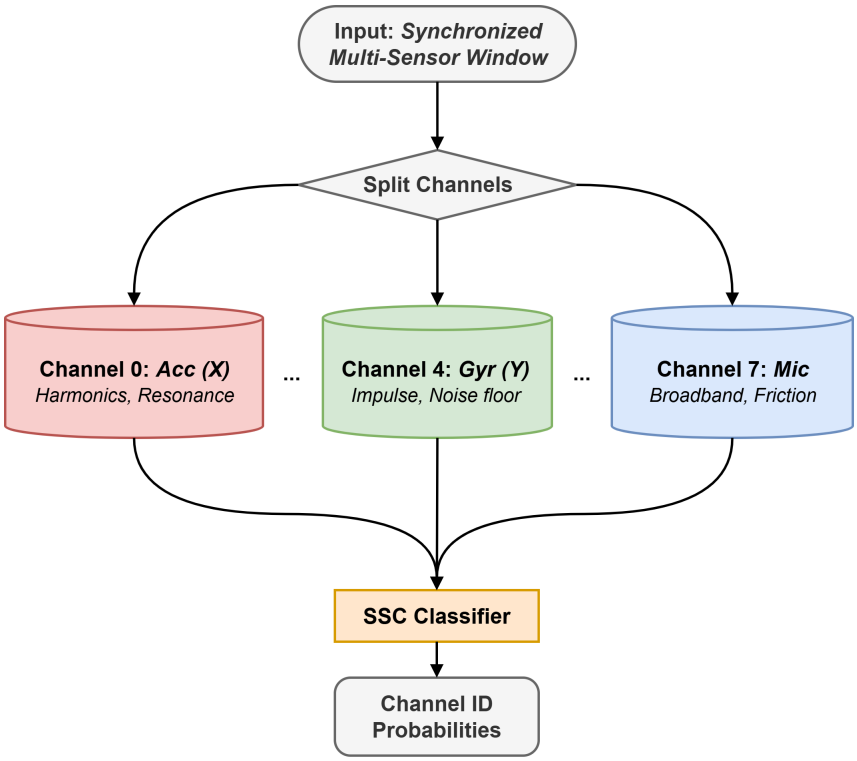



Figure 5: Conceptual visualization of the Sensor Channel Identification pretext task.

This concept is visualized in Figure 5. Unlike reconstruction models that attempt to reproduce the entire input, the SSC framework splits the

synchronized signal to learn these discriminative fingerprints based on the sensor's physical characteristics.

## 4.2 Training Phase

The training process is designed to be fully self-supervised, requiring only "normal" operating data collected from the target machine. This addresses the common "fleet-of-one" scenario where a model must be trained on a single machine instance without access to a large database of failure modes.

As illustrated in Fig. 5, the learning signal is obtained by turning each sensor-channel window into an independent classification instance whose label is the channel identity. This explicit framing links the pretext task to the training mechanics described below: the model is trained to separate channel-specific nominal signatures, thereby forcing it to learn representations of healthy operating behavior without any anomaly labels.

### 4.2.1 Batch Construction and Forward Pass

The training dataset $\mathcal{D}_{\text{train}}$ consists of multi-channel recordings of the machine in its healthy state. Although the sensors are physically synchronized, during training we treat them as independent classification instances.

As detailed in Algorithm 1, data is processed in mini-batches. At each training step, the data loader constructs a batch $\mathcal{B} = \{(x_i, y_i)\}_{i=1}^{B}$ by sampling windows from different sensor channels. The classifier $f_\theta$ processes these inputs and produces a vector of logits $z = f_\theta(x) \in \mathbb{R}^C$. The probability that a given input window $x$ belongs to channel $c$ is computed via the softmax function (Equation 11).

### 4.2.2 Optimization Objective

The model parameters $\theta$ are optimized to minimize the confusion between sensor channels. We employ the standard Cross-Entropy Loss:

$$\mathcal{L}_{\text{CE}}(x, y) = -\log(p_\theta(y \mid x)) \tag{10}$$

Gradients are computed via backpropagation ($\nabla_\theta \mathcal{L}$) and applied to update the weights using gradient-based optimizer (Adam). This optimization process is essential for anomaly detection: by forcing the model to minimize $\mathcal{L}_{\text{CE}}$ on normal data, the network effectively learns a decision boundary that

encapsulates the **nominal operating conditions** of the machine. It captures the exact spectral and temporal correlations that define "normality" for each sensor.

## 4.3 Inference Phase

During inference, the model parameters $\theta$ are frozen. The system monitors the machine by continuously analyzing incoming sensor streams and generating an anomaly score.

### 4.3.1 Synchronized Batch Processing

To maximize throughput and ensure low latency, we exploit the batched nature of neural network inference. At any discrete monitoring period, the system acquires a stream of $K$ windows for each of the $C$ channels (e.g., 6 IMU axes + 1 Microphone).

Rather than processing each time window sequentially, the windows from a **single sensor's recording** are stacked into a single inference batch. The model performs a batched forward pass, efficiently generating predictions for the entire timeline of that sensor. This process is repeated sequentially for each sensor channel. This strategy maximizes accelerator utilization by filling the batch dimension with temporal windows, ensuring efficient execution on constrained hardware.

### 4.3.2 Anomaly Scoring via Confidence Drop

We use the prediction confidence for the true channel ID $y$ as the primary anomaly indicator. Concretely, let $z = f_\theta(x) \in \mathbb{R}^C$ be the logits produced by the classifier for an input window $x$. The corresponding class probabilities are computed via the softmax function:

$$p_\theta(c \mid x) = \frac{\exp(z_c)}{\sum_{j=0}^{C-1} \exp(z_j)} \tag{11}$$

Under nominal conditions, the signal matches the learned fingerprint and the model identifies the channel correctly with high confidence ($p_\theta(y \mid x) \approx 1$). Conversely, fault-induced perturbations (e.g., a rattle or frequency shift) degrade the specific channel fingerprint, which confuses the classifier and causes the predicted probability $p_\theta(y \mid x)$ to drop.

To transform this probability into an unbounded score suitable for thresholding, we utilize the **Log-Odds Anomaly Score**:

$$A(x) = \log\left(\frac{1 - p_\theta(y \mid x)}{p_\theta(y \mid x)}\right) \tag{12}$$

A high score indicates a high probability of anomaly. For numerical stability, probabilities are clamped using machine epsilon $\epsilon$ before computation. In the remainder, we use $s(x)$ to denote this window-level anomaly score, i.e., $s(x) \equiv A(x)$.

### 4.3.3 Sliding-Window Aggregation

To handle continuous monitoring, we adopt a sliding-window strategy. For a given monitoring period $X$ consisting of $K$ time steps:

1. **Patch Scoring:** For each time step $k$, and for each sensor channel $c$, we compute the score $s_{k,c}$ using Equation 12.
2. **Channel Aggregation:** We compute the mean score across the time dimension for each channel.
3. **Global Aggregation:** Finally, the anomaly score for the entire machine is computed as the average of the channel-wise scores.

This hierarchical aggregation is detailed in Figure 6. By averaging scores across diverse physical modalities (acoustic and inertial), the system becomes robust to localized noise that might affect only a single sensor.

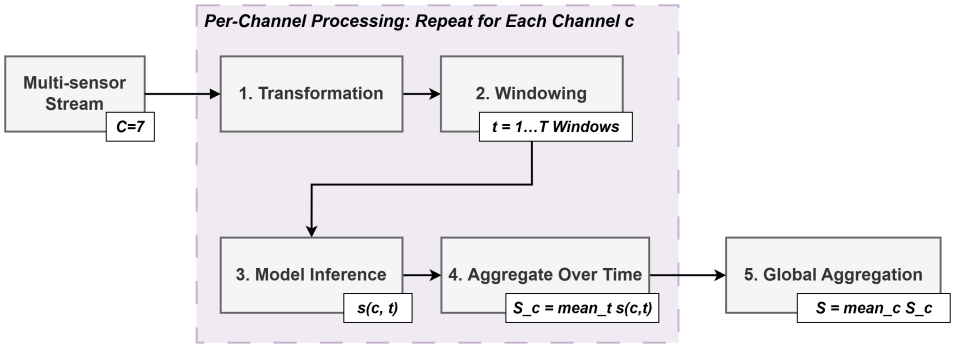



Figure 6: Schematic of the hierarchical inference and aggregation strategy.

### 4.4 End-to-End Procedure

To provide a clear implementation reference, we decouple the workflow into two distinct algorithms: the offline optimization phase and the online inference phase.

#### 4.4.1 Phase 1: Offline Training

The training procedure (Algorithm 1) is performed once. Note that the dataset $\mathcal{D}_{\text{train}}$ contains only healthy data. The supervisory labels $y$ correspond to the hard-wired sensor indices.

**Algorithm 1** Phase 1: Self-Supervised Training Procedure

**Require:** Dataset of normal tracks $\mathcal{D}_{\text{train}}$; Labels $y \in \{0, \dots, C-1\}$
**Initialization:** Randomly initialize network parameters $\theta$
**Preprocessing:** Convert raw streams to T-F representations
**for** epoch $e = 1$ **to** $E$ **do**
  **Shuffle** $\mathcal{D}_{\text{train}}$ into minibatches $B$
  **for** each batch $(x, y) \in B$ **do**
    $z \leftarrow f_\theta(x)$ {Forward pass}
    $p \leftarrow \text{Softmax}(z)$
    $\mathcal{L} \leftarrow -\log(p_y)$ {Cross-Entropy Loss}
    $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$ {Gradient Update}
  **end for**
**end for**
**return** Trained model parameters $\theta^*$

#### 4.4.2 Phase 2: Online Inference

The inference procedure (Algorithm 2) describes how the trained model assesses new data. The test recording is segmented into $K$ windows for each of the $C$ channels, and inference is performed by batching the $K$ windows of one channel at a time.

**Algorithm 2** Phase 2: Inference and Anomaly Scoring

**Require:** Test recording $X$; Trained model $\theta^*$

**Setup:** Freeze $\theta^*$ (eval mode)
**Segmentation:** Slice $X$ into $K$ windows per channel
Initialize score list $S \leftarrow []$
**for** each channel $c \in \{0, \ldots, C-1\}$ **do**
  Construct batch $x_{batch}$ containing all $K$ windows for channel $c$
  $P_{all} \leftarrow p_{\theta^*}(x_{batch})$ {Batched Forward Pass}
  **for** $k = 1$ **to** $K$ **do**
    $p \leftarrow P_{all}[k, c]$ {Extract prob for target class $c$}
    $s \leftarrow \log\left(\frac{1-p}{p}\right)$ {Log-Odds}
    Append $s$ to $S$
  **end for**
**end for**
$A_{\text{final}} \leftarrow \text{Mean}(S)$
**if** $A_{\text{final}} > \tau$ **then**
  **return** **Anomalous**
**else**
  **return** **Normal**
**end if**

# 5 Experimental Setup

This chapter details the experimental environment designed to validate the proposed Self-Supervised Classification (SSC) framework. We describe the IMAD-DS benchmark dataset, the preprocessing pipeline for multi-sensor synchronization, the specific model architectures for stationary and transient regimes, and the evaluation metrics used to quantify both edge feasibility and detection robustness. The experimental design benchmarks the proposed decoder-free architecture against a standard reconstruction baseline, focusing on the trade-off between computational footprint and generalization capability under domain shift.

## 5.1 Dataset: IMAD-DS

To evaluate the proposed framework, we utilize the **IMAD-DS (Industrial Multi-sensor Anomaly Detection under Domain Shift)** dataset [2]. IMAD-DS provides synchronized high-frequency streams from both acoustic (microphone) and inertial (accelerometer/gyroscope) sensors, which is essential for validating the *Sensor Channel Classification* pretext task across $C = 7$ channels. In addition, IMAD-DS incorporates domain shift by design by explicitly partitioning data into nominal operating conditions (Source) and rarely seen operating conditions (Target), enabling a controlled evaluation of robustness to changes in speed/load and background noise.

### 5.1.1 Sensors, Sampling, and Channel Definition

Each recording consists of 7 synchronized sensor channels, which serve as the ground-truth class labels $y \in \{0, \dots, 6\}$ for the SSC model. The Inertial Measurement Unit (IMU) contributes 6 channels comprising a 3-axis accelerometer (`acc_x`, `acc_y`, `acc_z`) and a 3-axis gyroscope (`gyro_x`, `gyro_y`, `gyro_z`); the microphone provides 1 channel capturing air-borne acoustic emissions (`mic`). In the raw data, the microphone is sampled at 16 kHz, while the industrial-grade IMU operates at approximately 7063 Hz. For our pipeline, all modalities are temporally aligned and resampled to a common time-base before windowing.

### 5.1.2 Machine Scenarios and Physical Anomalies

The dataset captures two distinct physical environments, allowing us to evaluate the method on both stationary and transient machinery. Crucially, all anomalies are *physically induced* on the real hardware, ensuring realistic

fault signatures rather than synthetic noise injections.

**Scenario A: Brushless Motor (Stationary Dynamics).** This setup features a controlled DC motor driving a belt system at constant rotational speeds, producing quasi-stationary signals dominated by harmonic components. The induced anomalies include **Mechanical Stress**, created by overtightening the drive belt to induce friction and alter harmonic balance, and **Magnetic Interference**, caused by placing a strong magnet near the chassis to disrupt the electromagnetic field and induce coil vibration.

**Scenario B: Robotic Arm (Transient Dynamics).** This setup involves a multi-joint manipulator executing repetitive pick-and-place cycles. The resulting signals are highly non-stationary, characterized by impulsive events and silence intervals linked to specific motion phases. Here, the primary anomaly is **Structural Instability**, simulated by selectively loosening screws at the base or joints. This creates subtle mechanical looseness that manifests as high-frequency vibrations or "rattles" only during specific actuation moments.

### 5.1.3 Domain Shift Protocol

To assess generalization, IMAD-DS partitions the data based on operating parameters and environmental conditions. The shift from Source to Target involves changes in background noise sources (from real factory recordings) and machine states.

**Source Domain:** Primary operating modes mixed with factory background noises A, C, D, and F (SNR $-4$ dB). For the robotic arm, this corresponds to lighter attached loads (indices $00, 10, 15, 20$). For the brushless motor, this corresponds to medium-to-high rotation speeds (1500, 1600, 1700, 1800, 1900, 2000, 2400, 2800, 3000 RPM).

**Target Domain:** Shifted operating modes mixed with factory background noises B, E, and G (SNR $-4$ dB). For the robotic arm, this corresponds to heavier attached loads (indices $25, 30, 35$). For the brushless motor, this corresponds to low rotation speeds (1000, 1100, 1200, 1300, 1400 RPM).

The following table details the specific data partition between Source and Target domains, including the sample counts for training and testing

across Normal and Anomaly classes.

Table 1: Number of samples for each class in source and target domains.

| | | Source Domain | | Target Domain | |
|---|---|---|---|---|---|
| **Machine** | **Split** | **Normal** | **Anomaly** | **Normal** | **Anomaly** |
| Robotic Arm | Train | 1812 | 0 | 27 | 0 |
| | Test | 116 | 116 | 116 | 116 |
| Brushless Motor | Train | 1263 | 0 | 18 | 0 |
| | Test | 78 | 78 | 78 | 78 |

## 5.2 Preprocessing and Input Representation

To ensure rigorous evaluation and reproducibility, all raw sensor streams are preprocessed into standardized time–frequency representations and stored in temporally aligned HDF5 containers. The complete extraction workflow is illustrated in Figure 7.

As depicted, asynchronous raw sensor streams (IMU at 7063 Hz, Microphone at 16 kHz) are temporally synchronized and resampled prior to segmentation. The pipeline employs a regime-specific normalization strategy: standard **Log-Mel** spectrograms are used for stationary machinery, while **Per-Channel Energy Normalization (PCEN)** is selected for transient regimes to suppress background noise.

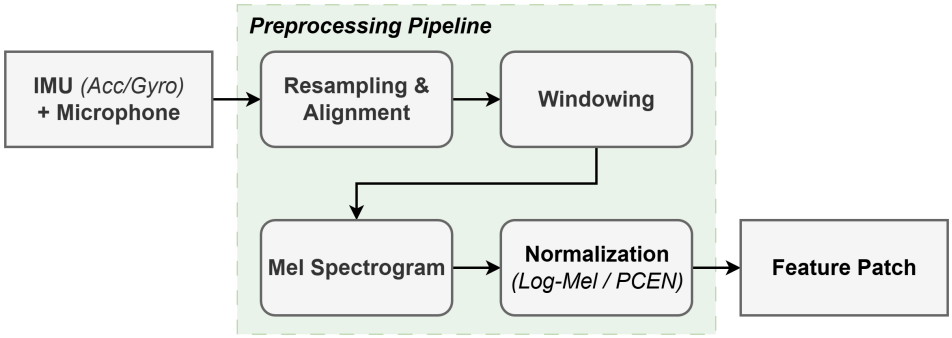



Figure 7: Schematic of the proposed multi-sensor feature extraction pipeline.

### 5.2.1 Time–Frequency Parameters and Synchronization

For all sensor channels, we generate spectrograms using the parameters detailed below. The window size of 64 ms was selected to provide sufficient frequency resolution for low-frequency mechanical vibrations (approx. 15 Hz resolution) while maintaining adequate temporal localization. Concretely, we use a frame length of 64 ms (1024 samples at 16 kHz) with hop length 32 ms (50% overlap), a Mel filterbank with 128 bins, and power exponent 2.0 (power spectrogram).

**Multi-Rate Synchronization:** A challenge in the IMAD-DS dataset is the sampling rate mismatch between the microphone (16 kHz) and the inertial sensors (7063 Hz). To enable strictly synchronized multi-channel learning, we employ a "minimum common frame" strategy:

1. A short initial settling period (35 ms) is discarded from all raw waveforms to remove sensor turn-on artifacts.
2. Spectrograms are computed independently for each stream.
3. All resulting feature maps are truncated to the minimum number of time frames available across the modalities, ensuring that the $t$-th column in the microphone spectrogram corresponds exactly to the $t$-th column in the accelerometer/gyroscope spectrogram.

### 5.2.2 Feature Selection: Log-Mel vs. PCEN

We adapt the feature compression strategy to the mechanical behavior of the target machine. For the **Stationary Regime (Brushless Motor)**, we employ standard **Log-Mel** features. Since the motor operates at constant speeds, the signal energy is stable, making static logarithmic compression ($S_{\log} = \log(S + \epsilon)$) sufficient and effective.

Conversely, for the **Transient Regime (Robotic Arm)**, we employ **Per-Channel Energy Normalization (PCEN)**. The robotic arm exhibits wide dynamic range variations due to start-stop cycles and varying distance from the sensors. PCEN acts as an adaptive gain control, enhancing the onset sharpness of mechanical events while suppressing background shifts. We use standard parameters for the IIR smoothing filter:

$$s = 0.025, \quad \alpha = 0.98, \quad \delta = 2.0, \quad r = 0.5, \quad \epsilon = 10^{-6} \tag{13}$$

### 5.2.3 STGram Inputs (Spectrogram + Waveform)

For the discrete-domain pipeline applied to the transient scenario, the model benefits from having access to fine-grained phase information typically lost in the spectral magnitude. The preprocessing pipeline therefore stores a dual representation consisting of a **Spectral Branch** (the aligned PCEN spectrogram, $128 \times T$) and a **Temporal Branch** (the corresponding raw waveform, resampled to a common 16 kHz clock). Critically, the waveform segments are cropped and padded to strictly match the temporal duration of the spectral windows, ensuring that the 1D convolution (time) and 2D convolution (frequency) branches view the exact same physical event.

## 5.3 Models and Baselines

To validate the effectiveness of the proposed approach, we benchmark the lightweight Self-Supervised Classification (SSC) models against the reconstruction-based baseline established for the IMAD-DS dataset.

### 5.3.1 Baseline: Reconstruction Autoencoder (FC-AE)

As a representative of standard unsupervised anomaly detection on IMAD-DS, we utilize the **Fully-Connected Autoencoder (FC-AE)** baseline provided with the benchmark [2]. This model operates on a "late fusion" principle: it flattens and concatenates features from all synchronized sensors into a single high-dimensional vector before processing.

**Input Representation.** The baseline processes **multi-rate, multi-sensor data** by forming an input vector from a **shared temporal window** of **100 ms**. Let each sensor stream be $x_s \in \mathbb{R}^{L_s \times C_s}$ with $s \in \mathcal{S} \triangleq \{\text{mic}, \text{acc}, \text{gyr}\}$, where $L_s$ is the number of samples in the 100 ms window (depending on sampling rate) and $C_s$ is the number of channels (e.g., $C_{\text{acc}} = C_{\text{gyr}} = 3$ for the three axes). Each channel is **z-score normalized** independently, yielding $\tilde{x}_s$, and all channels are stacked into a single column vector. The final fused AE input is

$$x = \left[\tilde{x}_{\text{mic}}^\top, \tilde{x}_{\text{acc}}^\top, \tilde{x}_{\text{gyr}}^\top\right]^\top \in \mathbb{R}^{\sum_{s\in\mathcal{S}} L_s C_s}. \tag{14}$$

Concretely, for IMAD-DS sampling rates, a 100 ms window corresponds to $L_{\text{mic}} = 1600$ and $L_{\text{acc}} = L_{\text{gyr}} \approx 670$, giving an input dimensionality of roughly $1600 \cdot 1 + 670 \cdot 3 + 670 \cdot 3 \approx 5620$ values.

**Architecture.** A symmetrical encoder–decoder network composed of dense (fully connected) layers. The encoder consists of three ReLU-activated layers followed by a low-dimensional bottleneck: $\mathrm{FC}\big(\sum_{s\in\mathcal{S}} L_s C_s, 2048, \mathrm{ReLU}\big)$, $\mathrm{FC}(2048, 2048, \mathrm{ReLU})$, $\mathrm{FC}(2048, 2048, \mathrm{ReLU})$, and a bottleneck $\mathrm{FC}(2048, 16, \cdot)$. The decoder mirrors the encoder to reconstruct $x$ from the 16-D latent code.

**Detection Logic.** Anomaly scores are derived from the reconstruction error. The benchmark uses a **dimension-normalized MSE**:

$$\mathcal{L}(\theta_e, \theta_d) = \frac{1}{\sum_{s\in\mathcal{S}} L_s C_s} \left\| x - \hat{x} \right\|_2^2, \quad \hat{x} = D(E(x \mid \theta_e) \mid \theta_d). \tag{15}$$

**Edge Suitability.** The baseline requires approximately **34 million parameters** (approx. 130 MB storage in FP32, 34 MB in Int8). This footprint, combined with the need to run the full decoder during inference, makes it computationally prohibitive for standard microcontrollers.

### 5.3.2 Proposed: Decoder-Free SSC Classifiers

We evaluate the proposed SSC framework using two architecture variants tailored to the signal dynamics, both significantly smaller than the baseline. Crucially, both are trained on the exact same pretext task: *identifying the source channel* among the 7 available sensors.

The **Stationary Pipeline (ResNet-SSC)**, designed for the Brushless Motor scenario, utilizes a lightweight ResNet-18 (**Batch Normalized**) backbone adapted for single-channel spectrogram inputs. To handle shifting motor speeds, it employs **Mixup** regularization ($\alpha = 0.4$), which enforces smooth decision boundaries in the frequency domain.

The **Transient Pipeline (STGram-ResNet-SSC)** [17], designed for the Robotic Arm scenario, employs a dual-branch STGram architecture. This fuses PCEN spectrograms with raw waveform segments using a lightweight ResNet-18 (**Instance Normalized**) backbone. To prevent overfitting to background noise harmonics, we apply **SpecAugment** with heavy frequency masking as a regularization strategy.

### 5.3.3 Architecture Details and Inductive Biases

To ensure feasibility on edge microcontrollers, the proposed method relies exclusively on compact, discriminative classifiers. Unlike Autoencoders which

require mirroring the encoder capacity in a decoder, SSC architectures terminate at an encoder embedding followed by a minimal classification head.

**General Decoder-Free Topology.** All SSC models share the following macro-architecture: an encoder backbone (convolutional feature extractor), followed by global average pooling (GAP) to collapse the spatial dimensions into a single vector, and a single dense classification layer producing $C = 7$ logits (one per sensor channel).

For the transient scenario specifically, we employ the **STGram-ResNet-SSC** variant, visualized in Figure 8. This dual-branch topology captures fine-grained temporal impulses (via 1D convolution on raw audio) and global spectral structures (via 2D ResNet) simultaneously.

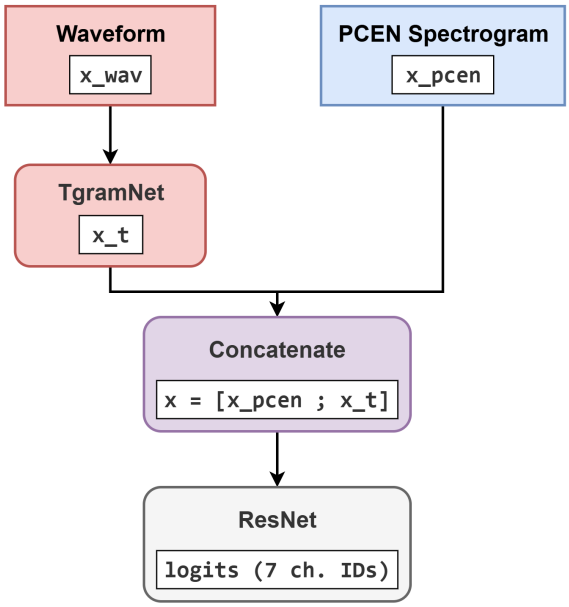



Figure 8: The STGram-ResNet-SSC architecture employed for the transient robotic arm scenario.

## 5.4 Training Implementation

All models were implemented in PyTorch and trained on a standard CUDA-enabled GPU. Training is performed in a fully self-supervised manner: the

dataset $\mathcal{D}_{\text{train}}$ consists exclusively of *Normal* operating data, and the supervisory labels $y$ are derived automatically from the sensor metadata (channel identity).

### 5.4.1 Patch Extraction and Dataset Construction

To create a fixed-size input tensor suitable for CNN processing, we do not train on full-length recordings, which can often vary in duration. Instead, we extract overlapping time–frequency patches using a sliding window approach parameterized by a **Patch Size ($T_w$)** of 64 time frames and a **Stride ($T_h$)** of 16 time frames.

Given a per-channel input spectrogram $S \in \mathbb{R}^{F \times T_{\text{total}}}$, the $k$-th input window $x_k$ is formally defined as:

$$x_k = S[:, k \cdot T_h : k \cdot T_h + T_w], \quad k = 0, \ldots, K-1 \tag{16}$$

where the number of extractable windows from a single spectrogram is:

$$K = \left\lfloor \frac{T_{\text{total}} - T_w}{T_h} \right\rfloor + 1 \tag{17}$$

Given hop length $\Delta t_{\text{hop}} = 32\,\text{ms}$ and window length $\Delta t_{\text{window}} = 64\,\text{ms}$, the physical duration of each input patch is:

$$T_{\text{phys}} = (T_w - 1) \cdot \Delta t_{\text{hop}} + \Delta t_{\text{window}} = (63 \cdot 0.032) + 0.064 \approx 2.08\,\text{s} \tag{18}$$

A single output spectrogram contains **232** frames for the Robotic Arm scenario and **162** frames for the Brushless Motor scenario. Using 64-frame patches with a 16-frame hop yields $N_{\textbf{robot}} = 11$ and $N_{\textbf{motor}} = 7$ full patches, respectively. All samples within each scenario produce spectrograms of identical length.

Each extracted patch $x_i$ is paired with its corresponding channel label $y_i \in \{0, \ldots, 6\}$. A random split is applied to the normal training data, allocating 80% for training and 20% for validation to monitor convergence.

**Waveform Alignment for Dual-Input Models (STGram).** For the STGram pipeline, the model requires a dual input: the spectrogram chunk described above and its corresponding, perfectly aligned raw waveform segment. Let the STFT hop length be $H_{\text{samples}}$ and the window length be

$W_{\text{samples}}$. Given a spectrogram chunk starting at frame index $t_0$, the corresponding waveform slice starts at:

$$n_0 = t_0 \cdot H_{\text{samples}} \tag{19}$$

The total waveform length $L$ required to cover the spectral window of $T_w$ frames is:

$$L = (T_w - 1) \cdot H_{\text{samples}} + W_{\text{samples}} \tag{20}$$

This ensures the temporal (1D) and spectral (2D) branches observe the same physical event, enforced in both the dataloader and inference engine.

### 5.4.2 Optimization Setup

We employ a large batch size strategy ($B = 512$) to stabilize the gradients of the channel classifiers. We use Adam with initial learning rate $\eta = 10^{-3}$, and train for 20 epochs, where each epoch is defined as 200 gradient steps (sampling with replacement) to ensure a consistent update count across dataset sizes. Gradient norms are clipped at $\|\nabla\|_2 = 1.0$ to prevent divergence during the initial phase.

For the learning-rate schedule, we apply (i) a linear warm-up to $\eta$ over the first 5 epochs, followed by (ii) `ReduceLROnPlateau` monitoring validation loss, with decay factor 0.1 and patience 2 epochs.

### 5.4.3 Regularization Strategies

To prevent the model from memorizing Source-domain background noise (overfitting) and to improve robustness against domain shift, we apply regime-specific regularization.

**Stationary Pipeline (Mixup).** We apply input Mixup [30], where virtual training examples are constructed as convex combinations of input pairs: $\tilde{x} = \lambda x_i + (1 - \lambda)x_j$. This encourages smooth decision boundaries across motor speeds and reduces overconfidence on narrow spectral peaks.

**Transient Pipeline (SpecAugment).** We apply SpecAugment [20] directly to spectrograms, using aggressive **frequency masking** (10 masks)

to reduce shortcut learning on static background noise during inactive moments. Time masking is kept minimal (1 mask) to preserve the temporal structure of transient onsets.

Table 2: Summary of training hyperparameters used for the two SSC pipelines.

| **Hyperparameter** | **Stationary** | **Transient** |
|---|---|---|
| Batch size | 512 | |
| Epochs | 20 | |
| Optimizer | Adam | |
| Base Learning Rate | $10^{-3}$ | |
| LR Warm-up | 5 epochs (Linear) | |
| LR Scheduler | Reduce on Plateau | |
| Gradient Clipping | 1.0 | |
| **Differences** | | |
| Regularization Method | **Mixup** | **SpecAugment** |
| Parameters | $\alpha = 0.4$ | **Time:** $N = 1, W = 6$ |
| | | **Freq:** $N = 10, W = 9$ |

## 5.5 Inference Protocol and Anomaly Scoring

The inference-time sliding-window procedure and the Log-Odds anomaly scoring rule are part of the proposed SSC methodology and are therefore defined in Chapter 4.3, Section 4.3.3. We follow that protocol unchanged for all experiments reported in this chapter.

## 5.6 Evaluation Metrics

The proposed framework is evaluated along two orthogonal axes: computational feasibility on the edge and fault detection reliability.

### 5.6.1 Edge Feasibility (The "Hardware Conflict")

To quantify the efficiency gap between reconstruction-based and classification-based methods, we measure the **parameter count** (total learnable weights) and the **estimated parameter Flash footprint**. The latter represents the space occupied by model parameters assuming standard **Int8 quantization**

(1 parameter $\approx$ 1 byte). We benchmark this footprint against a hard **2.0 MB Flash** ceiling for on-device model storage.

### 5.6.2 Detection Performance (AUROC)

We quantify anomaly detection accuracy using the Area Under the Receiver Operating Characteristic curve (AUROC). Unlike standard classification accuracy, which can be misleading in anomaly detection due to severe class imbalance (where "Normal" samples vastly outnumber "Anomalous" ones), AUROC provides a threshold-independent measure of separability.

It is computed by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) as the decision threshold $\tau$ is varied:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{21}$$

A score of 0.5 indicates random ranking, while 1.0 indicates perfect separability.

To provide a granular view of generalization, we report performance across three subsets. **Source AUROC** measures accuracy on nominal operating conditions seen during training. **Target AUROC** measures accuracy on shifted operating conditions (e.g., different speeds/loads) and is the primary metric for domain generalization. **Combined AUROC** reports performance on the union of Source and Target sets.

### 5.6.3 Comparative Metrics

To interpret the results relative to the baseline, we compute two derived quantities: detection-performance gain and footprint reduction.

**Performance Gain.** We report the AUROC improvement of SSC over the baseline as an absolute difference in AUROC points:

$$\Delta\text{AUROC} = \text{AUROC}_{\text{SSC}} - \text{AUROC}_{\text{Baseline}} \tag{22}$$

**Memory Reduction.** Let $M_{\text{Baseline}}$ and $M_{\text{SSC}}$ denote the Int8 parameter footprints (in bytes). We report both (i) a reduction factor and (ii) a percentage reduction relative to the baseline:

$$R_{\times} = \frac{M_{\text{Baseline}}}{M_{\text{SSC}}} \qquad R_{\%} = 100 \cdot \frac{M_{\text{Baseline}} - M_{\text{SSC}}}{M_{\text{Baseline}}} \tag{23}$$

## 5.7 Reproducibility Notes

To ensure the reliability of our findings, we adhere to a strict experimental protocol regarding randomness and model selection.

### 5.7.1 Deterministic Execution

Training and inference pipelines are designed to support deterministic execution. For the reported results, we seed the random number generators for `Python`, `NumPy`, and `PyTorch` with a fixed global seed and configure the backend to avoid non-deterministic algorithms.

# 6 Experimental Results

This chapter presents the quantitative evaluation of the proposed Self-Supervised Classification (SSC) framework. The analysis is structured to validate two core claims regarding the deployment of deep learning on constrained industrial edge nodes.

**Feasibility (RQ1).** We demonstrate that the proposed decoder-free architecture can satisfy the strict memory and latency constraints of standard Cortex-M microcontrollers, whereas traditional reconstruction baselines cannot.

**Performance (RQ2).** We show that, despite its significantly reduced footprint, the SSC model delivers improved detection performance overall, including under the evaluated domain shift, matching or exceeding heavier baselines on unseen operating conditions.

All results are reported for the model checkpoint yielding the **lowest validation loss** on the held-out normal dataset.

## 6.1 Edge Feasibility Analysis

Before evaluating detection capability, we must first establish deployability. A model with perfect accuracy is useless if it cannot physically fit onto the target hardware.

Table 3 benchmarks the computational footprint of our proposed SSC architecture against the standard dataset baseline (FC-AE) [2]. Estimations assume **8-bit integer (Int8) quantization**, which is the standard deployment format for microcontroller inference (e.g., via TensorFlow Lite for Microcontrollers or ST X-CUBE-AI).

Table 3: Edge Feasibility: Parameter Flash Footprint vs. 2 MB Flash Ceiling

| **Model** | **Parameters** | **Flash (Int8)** | **Deployable?** |
|---|---|---|---|
| Baseline (AE) | 33.9 M | 34.0 MB | ✗ **No** |
| **Ours (SSC)** | **0.9 M** | **1.0 MB** | ✓ **Yes** |
| *Improvement* | *37× Smaller* | *34× Smaller* | – |

**Analysis of the "Efficiency Gap."** The results highlight a structural mismatch between reconstruction-based architectures and edge hardware.

**The Autoencoder Bottleneck.** The baseline FC-AE requires approximately 34 MB of storage for its weights alone. This exceeds the available Flash memory of a high-end STM32H7 microcontroller (2 MB) by a factor of 17×. This bloat arises because the model must maintain dense connections to reconstruct the high-dimensional input vector, resulting in a parameter explosion.

**The SSC Advantage.** In contrast, the proposed SSC ResNet relies on convolutional weight sharing and a Global Average Pooling (GAP) layer. By collapsing the spatial dimensions before the final classification head, the SSC architecture decouples the input size from the parameter count. The resulting parameter footprint ($\approx 1$ MB in Int8) fits comfortably within the **2 MB Flash** ceiling. This confirms that the proposed shift from Generative Reconstruction to Discriminative Classification effectively closes the *efficiency gap*, transitioning anomaly detection from a server-class workload to a microcontroller-feasible task.

## 6.2 Comparative Detection Performance

Having established the hardware feasibility of the proposed approach, we now evaluate its detection reliability under domain shift. This is the critical test for industrial applicability: a model must distinguish between benign operational changes (e.g., speed variations) and true mechanical faults.

### 6.2.1 Case Study 1: Brushless Motor (Stationary Regime)

This scenario evaluates the model's ability to handle continuous shifts in the operating manifold, such as variations in rotation speed (RPM). The Autoencoder baseline is compared against our proposed **ResNet-SSC** pipeline. Quantitative results are presented in Table 4.

Table 4: Brushless Motor: Anomaly Detection Performance (AUROC)

| | Domain Split | | |
|---|---|---|---|
| **Model** | **Source** | **Target** | **Combined** |
| Baseline (FC-AE) | 73.6% | 55.6% | 59.0% |
| **Ours (SSC ResNet)** | **99.5%** | **96.3%** | **98.4%** |
| **Gain** | **+25.9%** | **+40.7%** | **+39.4%** |

Table 4 shows that the SSC model is markedly more robust, achieving a combined AUROC of **98.4%** compared to the baseline's 59.0%. The contrast is most pronounced on the Target domain: the baseline drops to 55.6% (close to random ranking), whereas SSC maintains near-perfect separation at 96.3%. This suggests that the reconstruction objective struggles to generalize across speed-induced shifts in spectral structure, while the discriminative channel-identification objective remains stable.

#### 6.2.2 Case Study 2: Robotic Arm (Transient Regime)

This scenario evaluates the model's robustness under highly non-stationary dynamics, where the signal is characterized by short, impulsive actuation events. Quantitative results are presented in Table 5.

Table 5: Robotic Arm: Anomaly Detection Performance (AUROC)

| | Domain Split | | |
|---|---|---|---|
| **Model** | **Source** | **Target** | **Combined** |
| Baseline (FC-AE) | 93.3% | 90.5% | 91.6% |
| **Ours (SSC ResNet)** | **98.9%** | **95.4%** | **95.5%** |
| **Gain** | **+5.6%** | **+4.9%** | **+3.9%** |

In the transient robotic arm setting (Table 5), the gap is narrower but still consistent: SSC improves the combined AUROC by **+3.9%**. The degradation from Source to Target is small for both models, indicating that the evaluated domain shift has a limited impact in this scenario. In this regime, the discriminative objective provides an additional margin without requiring reconstruction capacity.

## 6.3 Diagnostic Analysis

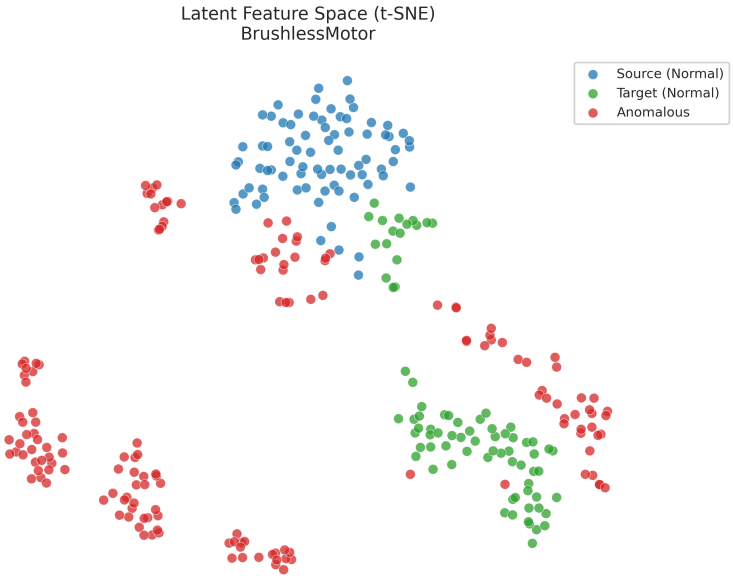
To understand the mechanism behind the quantitative performance differences, we perform a diagnostic analysis of the learned representations and score distributions.

### 6.3.1 Latent Space Visualization (t-SNE)

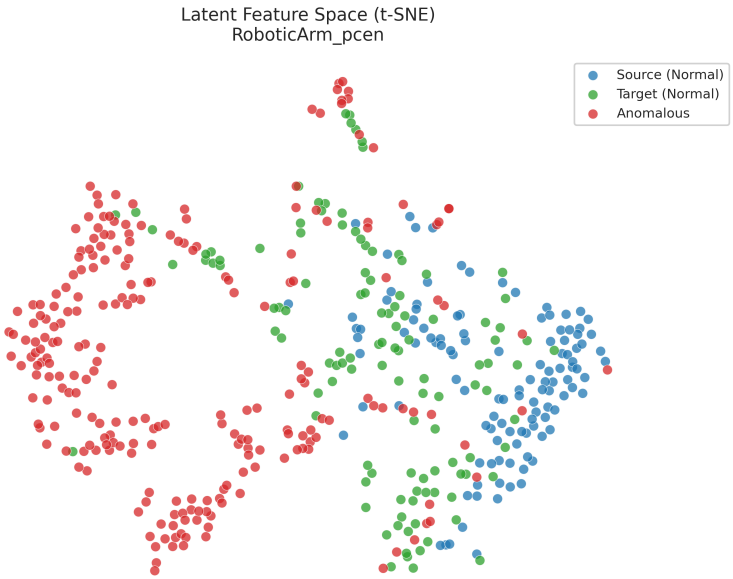
We visualize the high-dimensional features learned by the SSC encoder using t-SNE projections. Figure 9 illustrates the latent spaces for both scenarios.

For the **Brushless Motor (Left)**, the plot reveals three distinct clusters: Source Normal (Blue), Target Normal (Green), and Anomalies (Red). Crucially, the model separates the two operating speeds rather than merging them, but maintains a decisive margin between both normal states and the anomalous clusters.

For the **Robotic Arm (Right)**, the Source and Target normal samples cluster closely together, indicating that the architecture successfully normalizes the domain shift. Anomalies are pushed to the periphery, though they exhibit higher variance, reflecting the diverse nature of transient structural faults.




(a) Brushless Motor (Stationary)




(b) Robotic Arm (Transient)

Figure 9: t-SNE visualization of the learned embeddings.

### 6.3.2 Analysis of Failure Modes

The most significant finding in Section 6.2 is the collapse of the Autoencoder baseline on the Motor Target domain (AUROC 55.6%, Table 4). We attribute this failure to the **"Reconstruction Trap."**

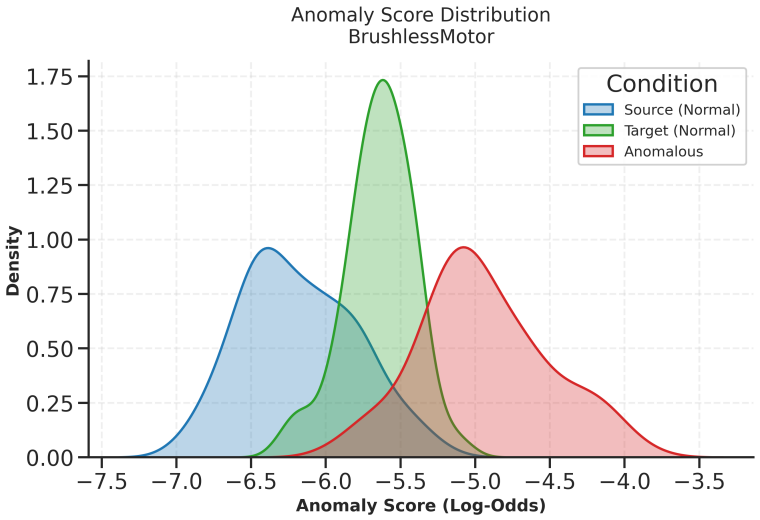
The failure mechanism stems from the Autoencoder's training objective, which minimizes pixel-wise error on the Source domain (high speeds). When the machine shifts to the Target domain (low speeds) and the fundamental harmonic frequencies shift, the model, having learned specific spectral textures rather than underlying physics, fails to reconstruct the new signal. This inability results in a high reconstruction error for *normal* Target samples. Consequently, the system incorrectly flags these operating points as anomalous, driving the False Positive Rate up and rendering the detector effectively random (AUROC $\approx 0.5$).

In contrast, the SSC model learns to identify the sensor channel. Since the physical transfer function of the sensor mounting does not change with motor speed, the discriminative model remains robust.

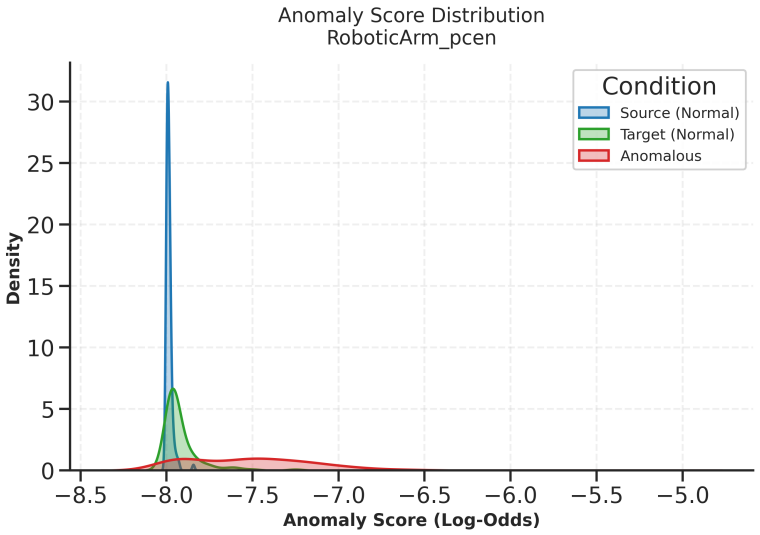
### 6.3.3 Anomaly Score Distributions

Finally, we examine the distribution of the Log-Odds anomaly scores to verify the separation margins.

Figure 10 confirms the robustness of the SSC metric. In the **Motor** case (a), although the Target Normal distribution (Green) shifts slightly right compared to Source (Blue), it remains completely disjoint from the Anomalous distribution (Red), explaining the near-perfect AUROC. In the **Robot** case (b), the overlap is larger due to the impulsive nature of the signal, but the bulk of the anomalous mass is still well-separated from the normal operating modes.




(a) Stationary Regime (Brushless Motor)




(b) Transient Regime (Robotic Arm)

Figure 10: Distribution of Anomaly Scores (Log-Odds) for stationary and transient regimes.

## 6.4 Summary of Findings

The experimental results presented in this chapter provide quantitative evidence supporting the central thesis: **Self-Supervised Classification (SSC) offers a superior trade-off between computational feasibility and detection robustness compared to traditional reconstruction methods.**

**Answer to RQ1 (Feasibility).** Table 3 confirms that the reconstruction baseline is structurally undeployable on standard Cortex-M7 hardware, requiring $\approx 34$ MB of parameter memory against a $\approx 1$ MB budget. In contrast, the proposed SSC architecture fits within the target memory envelope (approximately **1.0 MB** in Int8), enabling practical on-device deployment.

**Answer to RQ2 (Performance).** Across both case studies, SSC delivers improved detection performance overall, including under the evaluated domain shift. In the stationary motor scenario, SSC avoids the reconstruction failure mode highlighted in Table 4, maintaining a high combined AUROC (**98.4%**) while the baseline drops toward random ranking on the Target domain. In the transient robotic arm scenario, SSC achieves strong performance (combined AUROC **95.5%**) and improves upon the baseline despite using orders of magnitude fewer parameters.

Overall, these outcomes validate the proposed framework as a practical, high-performance solution for predictive maintenance, capable of bringing robust anomaly detection to the extreme edge of the industrial IoT.

# 7 Conclusions

This chapter first outlines the practical limitations of the current implementation and a set of concrete future research directions. It then closes with a final synthesis of the core contributions, positioning Self-Supervised Classification as a practical bridge between deep anomaly detection and constrained edge hardware.

## 7.1 Limitations

While the proposed Self-Supervised Classification framework successfully closes the feasibility gap for edge deployment and significantly improves generalization under domain shift, one practical limitation remains regarding its deployment workflow: a **Heuristic Configuration Dependency**.

Although the training process itself is self-supervised and automated, the *design* of the inference pipeline currently relies on a priori domain knowledge. Specifically, the choice of the optimal feature representation (e.g., Log-Mel for stationary motors vs. PCEN for transient robotics) and the appropriate regularization strategy (Mixup vs. SpecAugment) must be selected manually by an engineer based on the machine's mechanical behavior. This creates a "human reopening-the-box" requirement that prevents the system from being a truly universal, plug-and-play solution for arbitrary sensor nodes.

## 7.2 Future Directions

Building upon the foundations laid by this thesis, several research avenues could further mature TinyML anomaly detection, moving from *feasible* to *optimal* and eventually *autonomous*.

### 7.2.1 Towards Zero-Touch Deployment: Automated Signal Characterization

To address the configuration limitation identified above, future work should focus on automating pipeline selection. A lightweight **Meta-Controller** (or *Signal Characterization Module*) could analyze a short buffer of raw data to estimate low-level descriptors such as *Spectral Flux* (to detect transience) and *Energy Modulation Statistics* (to detect periodicity). Based on these metrics, the system could automatically select the appropriate feature

extractor (e.g., switching from Log-Mel to PCEN) and the best-suited regularization strategy. This would evolve the current framework into a plug-and-play edge agent capable of adapting to stationary pumps and transient robotic mechanisms without manual reconfiguration.

### 7.2.2 From Feasibility to Efficiency: On-Device Benchmarking & Optimization

This thesis established feasibility primarily through the lens of a **2 MB Flash** storage ceiling, which is the binary pass/fail criterion for fitting a model onto a microcontroller. The next step is to optimize **operational efficiency** (latency and energy). Future work should therefore include physical benchmarking on real targets (e.g., STM32H7 or Cortex-M4F) to measure inference latency and energy consumption per decision. In parallel, compiler-aware optimization using toolchains such as STMicroelectronics `X-CUBE-AI` or TensorFlow Lite Micro can be used to exploit kernel-level acceleration (e.g., CMSIS-NN) and operator fusion.

### 7.2.3 Extreme Compression: Quantization and Pruning Beyond Int8

While this thesis demonstrated feasibility on Cortex-M7 class devices using standard 8-bit integer (Int8) quantization, the edge spectrum extends down to even more constrained devices (e.g., Cortex-M4F or Cortex-M0+) with smaller Flash budgets. To bridge this gap, future work should explore more aggressive compression strategies: Quantization-Aware Training (QAT) to preserve accuracy under low precision, structured pruning to reduce FLOPs by removing entire filters, and sub-byte quantization (e.g., 4-bit) to potentially reduce the footprint by an additional factor of $2\times$ to $4\times$.

### 7.2.4 Lifelong Learning: Unsupervised Online Adaptation

A fundamental challenge in industrial deployment is **concept drift**. Over long time horizons, "normal" machine behavior can drift due to mechanical wear, lubricant degradation, or seasonal temperature shifts. A static model trained once may eventually interpret this benign drift as anomalous (false positives). A critical research direction is therefore unsupervised online adaptation protocols that can run directly on the microcontroller. Promising mechanisms include adaptive normalization (updating running statistics to track slow changes) and dynamic thresholding (calibrating the decision threshold using a sliding window of recent scores). These methods

must be designed to avoid model contamination, ensuring the system adapts to gradual drift without learning to ignore sudden true faults.

## 7.3 Concluding Remarks

The rapid proliferation of Industry 4.0 has created an urgent demand for predictive maintenance solutions that are not only intelligent, but also deployable at the extreme edge directly on the sensor node. This thesis addressed the challenge of enabling **generalizable, multi-sensor anomaly detection** on **resource-constrained microcontrollers (TinyML)**.

We argued that the prevailing deployment pattern of reconstruction-based architectures (e.g., Autoencoders) is fundamentally ill-suited for microcontroller-class devices. This approach creates a prohibitive **Efficiency Gap**, where the computational and memory cost of the decoder prevents deployment on standard embedded hardware.

In response, we established a **Self-Supervised Classification (SSC)** framework for industrial anomaly detection. By shifting the learning objective from *generative, pixel-wise reconstruction* to *discriminative Sensor Channel Identification*, we demonstrated that it is possible to learn anomaly-sensitive representations of machinery *without* the massive computational and memory overhead of a decoder.

Concretely, this thesis supports three primary conclusions:

**1) The decoder is unnecessary for edge anomaly detection.** We demonstrated that the decoder component of standard Autoencoders constitutes dead weight for deployment-focused anomaly detection: it is required for reconstruction, but it provides no additional value for the binary decision of fault detection, yet it dominates the memory footprint. By adopting a decoder-free, classifier-only architecture, we reduced the model size by a factor of $\approx 30$ (from $\approx 34$ MB to $\approx 1$ MB), enabling deployment under a **2 MB Flash** storage ceiling (see Table 3). This reduction is not merely an optimization but an enabler: it makes on-device anomaly detection physically feasible on Cortex-M7 class hardware where the baseline architecture is undeployable.

**2) Discriminative learning improves robustness under domain shift.** Beyond efficiency, we showed that the SSC objective yields superior robust-

ness in shifted operating conditions. Reconstruction models can fall into a reconstruction trap, where benign operational changes (e.g., speed changes) cause high reconstruction errors and therefore false alarms. In contrast, SSC learns features that remain stable across these benign shifts while remaining sensitive to true faults. Empirical evaluation on IMAD-DS confirms this advantage: in the stationary motor scenario, SSC achieves a combined AUROC of 98.4% versus 59.0% for the baseline (Table 4); in the transient robotic arm scenario, SSC improves the combined AUROC from 91.6% to 95.5% (Table 5).

**3) Sensor Channel Identification closes the single-machine gap of SSC.** Finally, we showed that the SSC paradigm can be made practical in the common fleet-of-one setting. Where Machine-ID SSC requires multiple distinct machine instances to define classes, our formulation uses the multi-sensor nature of industrial nodes to generate labels for free. This enables self-supervised training on a single machine while still inducing a meaningful discriminative task, providing a practical path to deployment when only nominal data from one target asset is available.

# References


[1] Abadade, Y., et al. (2023). *A comprehensive survey on TinyML.* IEEE Access, 11, 96892–96922.

[2] Albertini, F., et al. (2024). *IMAD-DS dataset overview.* In *Proceedings of the DCASE 2024 Workshop.*

[3] Allen, J. (1977). *Short-time spectral analysis, synthesis, and modification by discrete Fourier transform.* IEEE Transactions on Acoustics, Speech, and Signal Processing, 25(3), 235–238.

[4] Audibert, J., et al. (2020). *USAD: Unsupervised anomaly detection on multivariate time series.* In *Proceedings of KDD 2020.*

[5] Banbury, C., et al. (2021). *MICRONETS: Neural network architectures for deploying TinyML applications on commodity microcontrollers.* In *Proceedings of MLSys 2021.*

[6] Ben-David, S., et al. (2010). *A theory of learning from different domains.* Machine Learning, 79(1), 151–175.

[7] Chalapathy, R., & Chawla, S. (2019). *Deep learning for anomaly detection: a survey.* arXiv preprint arXiv:1901.03407.

[8] Dohi, K., et al. (2021). *Flow-based self-supervised density estimation for anomalous sound detection (Glow-Aff).* In *Proceedings of ICASSP 2021.*

[9] Dosovitskiy, A., et al. (2020). *An image is worth* $16 \times 16$ *words: transformers for image recognition at scale.* arXiv preprint arXiv:2010.11929v2.

[10] Garg, S., et al. (2023). *Data-centric and model-centric AI: twin drivers of compact and robust Industry 4.0 solutions.* IEEE Access, 11, 16233–16259.

[11] Giri, R., et al. (2020). *Self-supervised classification for detecting anomalous sounds.* In *Proceedings of the DCASE 2020 Workshop.*

[12] Guan, J., et al. (2023). *Anomalous sound detection using audio representation with machine-ID-based contrastive learning pretraining (CLP-SCF).* In *Proceedings of ICASSP 2023.*

[13] He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition.* In *Proceedings of CVPR 2016*.

[14] Ioffe, S., & Szegedy, C. (2015). *Batch normalization: accelerating deep network training by reducing internal covariate shift.* In *Proceedings of ICML 2015*.

[15] Koizumi, Y., et al. (2020). *Description and discussion on DCASE 2020 Challenge Task 2: unsupervised anomalous sound detection for machine condition monitoring.* arXiv preprint arXiv:2006.05822.

[16] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-based learning applied to document recognition.* Proceedings of the IEEE, 86(11), 2278–2324.

[17] Liu, Y., et al. (2022). *Anomalous sound detection using spectral–temporal information fusion (STgram-MFN).* In *Proceedings of ICASSP 2022*.

[18] Lostanlen, V., et al. (2018). *Per-channel energy normalization: why and how.* IEEE Signal Processing Letters, 26(1), 39–43.

[19] Micheleri, F., et al. (2023). *Towards differentiating between failures and domain shifts in industrial data streams.* In *Proceedings of the CIKM 2023 Workshops*.

[20] Park, D. S., et al. (2019). *SpecAugment: a simple data augmentation method for automatic speech recognition.* In *Proceedings of Interspeech 2019*.

[21] Sakurada, M., & Yairi, T. (2014). *Anomaly detection using autoencoders with nonlinear dimensionality reduction.* In *Proceedings of the MLSDA 2014 Workshop on Machine Learning for Sensory Data Analysis*.

[22] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). *MobileNetV2: inverted residuals and linear bottlenecks.* In *Proceedings of CVPR 2018*.

[23] Sha, Y., et al. (2022). *Regional–local adversarially learned one-class classifier anomalous sound detection in global long-term space (GRL-Net).* In *Proceedings of KDD 2022*.

[24] Shi, W., et al. (2016). *Edge computing: vision and challenges.* IEEE Internet of Things Journal, 3(5), 637–646.

[25] Stevens, S. S., Volkmann, J., & Newman, E. B. (1937). *A scale for the measurement of the psychological magnitude of pitch.* Journal of the Acoustical Society of America, 8(3), 185–190.

[26] Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2016). *Instance normalization: the missing ingredient for fast stylization.* arXiv preprint arXiv:1607.08022.

[27] Verbitskiy, S., et al. (2022). *ERANNs: efficient residual audio neural networks for audio pattern recognition.* Pattern Recognition Letters, 161, 38–44.

[28] Wang, Y., Getreuer, P., Hughes, T., Lyon, R. F., & Saurous, R. A. (2017). *Trainable frontend for robust and far-field keyword spotting.* In *Proceedings of ICASSP 2017.*

[29] Wang, Y., et al. (2024). *A mel spectrogram enhancement paradigm based on continuous wavelet transform.* arXiv preprint arXiv:2406.12164.

[30] Zhang, H., et al. (2018). *mixup: beyond empirical risk minimization.* In *Proceedings of ICLR 2018.*

[31] Zonta, T., et al. (2020). *Predictive maintenance in Industry 4.0: a systematic literature review.* Computers & Industrial Engineering, 150, 106889.

[32] Chandola, V., Banerjee, A., & Kumar, V. (2009). *Anomaly detection: a survey.* ACM Computing Surveys (CSUR), 41(3), 1–58.

[33] Schölkopf, B., et al. (2001). *Estimating the support of a high-dimensional distribution.* Neural Computation, 13(7), 1443–1471.

[34] Tax, D. M. J., & Duin, R. P. W. (2004). *Support vector data description.* Machine Learning, 54, 45–66.

[35] Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). *A simple framework for contrastive learning of visual representations (SimCLR).* In *Proceedings of ICML 2020.*

[36] He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). *Momentum contrast for unsupervised visual representation learning (MoCo).* In *Proceedings of CVPR 2020.*

[37] Golan, I., & El-Yaniv, R. (2018). *Deep anomaly detection using geometric transformations.* In *Proceedings of NeurIPS 2018.*

[38] Jing, L., & Tian, Y. (2020). *Self-supervised visual feature learning with deep neural networks: a survey.* IEEE Transactions on Pattern Analysis and Machine Intelligence.

[39] Ruff, L., et al. (2018). *Deep one-class classification.* In *Proceedings of ICML 2018*.

[40] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *ImageNet classification with deep convolutional neural networks.* In *Proceedings of NeurIPS 2012*, 25.

[41] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* MIT Press.

[42] Dieleman, S., & Schrauwen, B. (2014). *End-to-end learning for music audio.* In *Proceedings of ICASSP 2014*.

[43] Kiranyaz, S., et al. (2021). *1D convolutional neural networks and applications: a survey.* Mechanical Systems and Signal Processing, 151, 107398.

[44] Tan, M., & Le, Q. (2019). *EfficientNet: rethinking model scaling for convolutional neural networks.* In *Proceedings of ICML 2019*.

[45] STMicroelectronics. (n.d.). *STM32H7 series - Arm Cortex-M7 and Cortex-M4 MCUs (480 MHz).* STMicroelectronics Product Page. (Embedded Flash memory ranging from 64 KB to 2 MB.)