applied sciences

MDPI

Article

A Two-Stage Parallel Inception 1D-CNN for Event-Driven Condition Monitoring

Dávid Bodnár and Károly Jármai *

Faculty of Mechanical Engineering and Informatics, University of Miskolc, 3515 Miskolc, Hungary; david.bodnar@student.uni-miskolc.hu
* Correspondence: jarmai@uni-miskolc.hu

Abstract

Digital twins (DTs) of collaborative robots need continuous synchronization, but streaming high-rate kinematic telemetry to a central DT can create bandwidth bottlenecks. Protocol overhead exceeds 70% of the packet volume for raw sensor data in a projected 1000 Hz scenario. This work proposes an Edge-Enabled Digital Twin Networked Control System (E-DTNCS) in which an edge node classifies IMU data windows and sends a low-rate heartbeat in normal operation and a raw-data burst when an anomaly is detected. A compact two-stage Parallel Inception 1D-CNN (PINC-noBN, 37,154 parameters) is evaluated as a classifier on the public CASPER dataset (20 Hz UR3e IMU data) with a chronological split and three seeds. With the given labels, it reaches a test accuracy of 0.86 ± 0.02, slightly below a CASPER-style 1D-CNN with kernel size 50 (0.88 ± 0.01, 169,794 parameters). Errors concentrate near class transitions, where labels are offset from the signal by several minutes; excluding these zones, accuracies are 0.97 and 0.99. An analytical projection including false-positive bursts indicates bandwidth reductions of 85–92% (labels as given) and 90–99.8% (transition zones excluded) for anomaly prevalences between 10% and 0.1%. In float32, PINC-noBN needs at least 2 MB of activation memory.

Keywords: digital twin; edge computing; machine learning; 1D-CNN; condition monitoring; TinyML; protocol overhead; network bandwidth

1. Introduction

The Fourth Industrial Revolution (Industry 4.0) is fundamentally characterized by the convergence of physical manufacturing systems with advanced computing models, leading to the widespread implementation of Cyber-Physical Systems (CPS) and digital twin (DT) technologies. A digital twin is not simply a static 3D representation or a passive dashboard; it is a dynamic, living computational model that requires continuous, bidirectional synchronization with its physical counterpart. Closed-loop integration is a basic prerequisite for supporting real-time condition monitoring, predictive maintenance, and self-governing control optimizations [1]. The fidelity, predictive power, and overall utility of a DT are inextricably linked to the granularity and frequency of the physical phenomena it monitors [2]. In highly dynamic industrial applications—particularly those involving multi-axis collaborative robots (cobots) such as the Universal Robots UR3e (Universal Robots A/S, Odense, Denmark)—this synchronization depends heavily on capturing high-frequency micro-kinematics. Continuous vibration and acceleration telemetry, derived from both internal motor encoders and external Inertial Measurement Units (IMUs), provide the data

Check for updates

Academic Editor: Iwona Grobelna

Received: 31 July 2026
Revised: 1 November 2026
Accepted: 2 November 2026
Published: 5 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

necessary to detect mechanical failures, such as bearing wear or gear backlash, possibly before catastrophic failure.

However, the pursuit of ultra-high-fidelity digital twins has unintentionally engineered a severe structural bottleneck within modern Operational Technology (OT) networks. Traditional DT frameworks operate predominantly on centralized architectures, relying on cloud-based servers or localized high-performance computing (HPC) clusters to ingest, parse, and analyze massive volumes of machine data. Recent reviews of edge and cloud computing approaches document [3] that cloud-centric architectures introduce unpredictable communication latency and bandwidth saturation for time-critical industrial applications [4], making closed-loop, self-governing control difficult.

1.1. The Protocol Overhead Bottleneck

This structural bottleneck stems from the fundamental mechanics of industrial data transmission. Streaming raw, high-density sensor data—often sampled at frequencies exceeding 1000 Hz to capture transient mechanical shocks [5]—directly from the physical asset to the centralized digital twin effectively saturates standard industrial communication buses. Modern industrial control systems frequently use protocols such as Real-Time Data Exchange (RTDE) layered over standard IPv4 and Transmission Control Protocol (TCP) infrastructure [6].

When transmitting micro-payloads, such as a 24-byte array of 6-axis floating-point values from an IMU, the required network routing headers (Ethernet, IP, and TCP [7]) consume a disproportionate share of the total packet size. Transmitting these top-heavy, fragmented packets at a theoretical 1000 Hz means routing headers consume a large share of network bandwidth (about 71% at 1000 Hz, Section 2.7) rather than physical data. This continuous micro-transaction volume can overwhelm factory switch buffers, increasing queueing latency (t_{queue}) and degrading the DT's responsiveness. In environments where dozens of robotic arms operate simultaneously on a shared subnet, this overhead can cause network congestion and packet loss.

1.2. Related Work

The digital twin (DT) paradigm has evolved from a static, simulated representation of aerospace structures to a dynamic computational model central to Industry 4.0. As Lu et al. [8] note, a genuine DT in smart manufacturing surpasses traditional simulation by requiring high-fidelity modeling of functional dynamics, enabled by near-real-time, bi-directional synchronization between physical and cyber domains. Lu et al. emphasize that DT architecture is fundamentally constrained by its information models, communication mechanisms, and data processing functions. Although server-based architectures aggregate data centrally to exploit cloud-computing economies of scale, latency and unpredictable routing delays limit them. This latency impedes the deployment of DTs for delay-sensitive, closed-loop control of multi-axis collaborative robots. Consequently, transitioning to data-driven, edge-integrated digital twins is necessary to manage the continuous flow of high-frequency physical telemetry.

Friederich et al. [9] further emphasize the need to transition from manually crafted simulation models to fully data-driven digital twins. In reconfigurable manufacturing systems, rapidly changing market demands require frequent modifications to factory layouts and production parameters, making traditional expert-built simulation models costly to update. Friederich et al. propose a generic data-centric framework that depends on continuous data collection, real-time validation, and automated process discovery. Their research also identifies the sheer volume of continuous high-frequency data logs. Streaming uncompressed, raw condition-monitoring data from hundreds of IoT sensors across the

entire Product Life Cycle directly to the cloud can overwhelm standard networking infrastructure. Furthermore, if noise, dropped packets, or transmission latency compromise this data, the accuracy of subsequent machine learning models and process mining algorithms is significantly reduced.

He et al. [10] addressed the need for high-fidelity, real-time data processing by integrating data-driven digital twins for optimized fault-tolerant control in process systems. Their research shows that complex industrial structures operating under demanding conditions require localized, adaptive system identification to accurately model closed-loop virtual systems. By developing adaptive Subspace Identification Methods (SIMs) and incorporating parameterization, He et al. decoupled sensor-fault diagnosis from actuator failures, enabling dynamic reconfiguration of control gains. However, their findings also highlight the major computational burden associated with these optimizations. While effective in localized simulations, scaling continuous, high-frequency residual evaluations across an entire factory floor requires a decentralized approach that places intelligence as close as possible to the physical data source to prevent delays in automated fault-tolerant responses.

The digital twin paradigm has expanded beyond mechanical assets to encompass socio-technical systems, as Preuss et al. demonstrate [11]. The future of smart manufacturing requires integrating physical machines with digital user representations, leading to the concept of the Cognitive digital twin. Preuss et al. demonstrate that incorporating high-frequency sensor data to construct a Digital Shadow of human-machine interaction is essential for early-stage human-centered design and adaptive ergonomics. As the digital twin's scope broadens to include continuous cognitive and kinematic mapping of human operators alongside industrial cobots, the resulting exponential increase in telemetry renders centralized processing impractical. This socio-technical expansion intensifies the vulnerabilities of centralized cloud architectures, necessitating localized filtering mechanisms to prevent network overload.

The need for edge-based decentralization is closely tied to the historical development of data exchange standards in manufacturing. Pratt [12] describes how product data exchange was originally established on comprehensive semantic frameworks such as ISO 10303 (STEP), which replaced earlier geometric specifications like IGES. STEP introduced robust Application Protocols (APs) for transferring detailed boundary representations, assembly structures, and configuration-controlled design data. Although standards such as STEP AP242, MTConnect, and OPC UA [8] have advanced CAD/CAM interoperability, their schemas are primarily designed for batch-oriented engineering file transmission. Applying these heavy semantic frameworks to continuous, high-frequency micro-telemetry, such as 1000 Hz data from an Inertial Measurement Unit (IMU) on a robotic arm, creates a significant mismatch. Encapsulating a small floating-point array within extensive TCP/IPv4 headers and XML/JSON schemas wastes network bandwidth on routing overhead, severely impairing the deterministic latency required for cyber-physical synchronization.

Beyond the mathematical inefficiency of protocol overhead, streaming continuous raw telemetry across factory networks introduces significant cybersecurity risks. Implementing Information Security Management Systems (ISMS), particularly ISO/IEC 27001, is now mandatory for both IT and OT networks. Kitsios et al. [13] emphasize the necessity for rigorous risk assessment and treatment strategies to safeguard the Confidentiality, Integrity, and Availability (CIA triad) of data. At the same time, addressing the unique vulnerabilities of Industrial Automation and Control Systems (IACS) requires frameworks such as IEC 62443. Shaaban et al. [14] addressed the challenge of securing dispersed IoT components by proposing an IEC 62443-based optimization algorithm for security attributes. Their research demonstrates that redundant security measures across all nodes degrade system functionality and availability. Instead, they support identifying critical gate components and

establishing distinct security zones and conduits to intercept threats before they propagate. In smart factories, exposing continuous raw kinematic data streams over shared industrial networks significantly increases the attack surface, enabling spoofing and Distributed Denial of Service (DDoS) attacks. Deploying a localized semantic check-valve aligns with ISO 27001's optimized security zoning and risk-avoidance mandates. This approach analyzes data locally and transmits only low-frequency, event-driven state updates, thereby reducing the exposure of sensitive operational data and insulating the centralized digital twin from network delays and cyber threats.

Achieving localized semantic translation at the extreme edge requires a fundamental shift in how we compute physical phenomena. Traditional computational mechanics depend on dense mathematical simulations, which are too resource-intensive for microcontrollers. Capuano and Rimoli [15] addressed this computational bottleneck by introducing "Smart Finite Elements." In multiscale finite element formulations, calculating the internal displacement field for each element requires iterative numerical solvers, which become prohibitive in nonlinear scenarios. To overcome this, Capuano and Rimoli employed machine learning techniques—specifically Support Vector Regression (SVR) and Ordinary Least Squares—as surrogate models. By training these algorithms on data from existing finite elements, they established a direct relationship between an element's macroscopic state and its internal forces, eliminating the need to compute the internal displacement field. Their findings show that machine learning can compress and replicate detailed physical behavior into lightweight, computationally efficient surrogate models, provided training enforces physical constraints such as conservation of linear and angular momentum to prevent erroneous outputs.

Udrescu and Tegmark [16] advanced the use of artificial intelligence as a surrogate for detailed physics by developing the "AI Feynman" algorithm. To address the NP-hard problem of symbolic regression—identifying mathematical formulas that accurately represent experimental data—they showed that neural networks can uncover fundamental physical properties such as translational symmetries, scaling symmetries, and multiplicative separability. Traditional genetic algorithms, such as Eureqa, often fail or become trapped in local optima when processing complex datasets. In contrast, Udrescu and Tegmark's approach combines neural network fitting with physics-inspired techniques to recursively decompose complex, multi-variable problems into simpler, lower-dimensional components. This research marks a paradigm shift: properly constrained neural networks are not merely black-box pattern matchers but effective tools for isolating the governing physical properties of dynamic systems, making them well-suited to translating raw kinematic data into semantic states.

Physics-Informed Neural Networks (PINNs) have attracted significant attention for integrating deep learning with fundamental physical laws. Lawal et al. [17] systematically review how PINNs bridge machine learning and scientific computing by embedding Partial Differential Equations (PDEs) directly into the neural network's loss function. This approach forces the network to produce solutions consistent with physical laws, substantially reducing the training data required compared with conventional deep learning models. However, Lawal et al. also document major limitations of standard PINNs, including high training costs, slow convergence, and susceptibility to vanishing gradient issues in deep architectures. Although Extended PINNs (XPINNs) and Hybrid PINNs address these challenges through domain decomposition and ensemble methods, these solutions remain computationally intensive. Deploying continuous, iterative PINNs or complex sequence models (such as LSTMs or Transformers) on resource-constrained hardware like an ARM Cortex-M4 microcontroller with only 0.06 MB of SRAM is impractical. Therefore, while the physics-informed approach is desirable, implementation at the extreme edge requires a more efficient

architecture, such as a specialized 1D Convolutional Neural Network (1D-CNN) capable of hierarchical spatial down-sampling without exceeding memory constraints.

The need for specialized, lightweight architectures is especially pronounced in condition monitoring and cross-domain fault diagnosis. Industrial equipment often operates under highly variable speeds and loads, making it difficult to train reliable fault classifiers with limited, domain-specific data. Ye et al. [18] addressed this challenge by introducing an Adaptive Threshold Network integrated with Model-Agnostic Meta-Learning (ATN-MAML). Their approach uses the Synchrosqueezing Wavelet Transform (SWT) to convert 1-D time-series vibration signals into high-resolution 2-D time-frequency images, effectively limiting energy leakage. Importantly, Ye et al. incorporated an adaptive threshold acquisition module to filter out irrelevant environmental noise that commonly affects smart factory data. Extensive testing on the CWRU and SDU datasets showed that their meta-learning approach maintains high diagnostic accuracy even when source and target domains differ significantly in rotational speed and have high Signal-to-Noise Ratios (SNRs).

However, converting 1-D signals into 2-D images using SWT and executing meta-learning algorithms requires substantial computational resources, typically available only on localized GPUs or cloud infrastructure. For an Edge-Enabled digital twin Networked Control System (E-DTNCS) to operate without overloading the network, this diagnostic capability must be distilled into a form executable directly at the sensor level. The proposed architecture uses a two-stage Parallel Inception 1D-CNN (PINC) operating on 700-time-step multivariate windows, enforcing channel-wise independence to isolate spatial features and avoiding computationally expensive 2-D wavelet transformations. This semantic check valve at the edge is intended to bridge the gap between heavy industrial data standards and the low-latency demands of some digital twins; whether it does so remains to be shown, because this study evaluates only the classifier and an analytical bandwidth projection.

1.3. Restrictions of Current Edge Implementations

To mitigate latency and bandwidth constraints, the industrial sector has increasingly adopted edge computing. Although recent studies have examined data compression and Physics-Informed Neural Networks (PINNs) to enhance digital twin (DT) performance [19], most current edge implementations primarily serve as localized data gateways. These systems typically perform basic down-sampling, filtering, or batching before forwarding raw telemetry to centralized servers for analysis. A significant gap remains in architectural strategies that enable true semantic translation at the point of data ingestion. Existing frameworks generally lack a localized, resource-constrained intelligence layer that fully decouples the physical data generation rate from the digital twin communication rate. For scalable deployment across factory floors with numerous robotic agents, such decoupling is essential [20]. Edge nodes must move beyond simple pass-through routing; they must interpret data streams algorithmically and transmit only meaningful semantic state changes to the DT. Despite recent integration of machine learning into digital twin ecosystems, most implementations still treat edge nodes as basic data conduits. As Table 1 summarizes, state-of-the-art architectures still rely on continuous streaming or batching of raw telemetry to centralized cloud servers, limiting scalability due to substantial TCP/IPv4 overhead during high-frequency transmission. The proposed E-DTNCS framework differs from this model by moving convolutional intelligence to the extreme edge using the hardware-constrained PINC model. This approach turns the edge node into an active semantic check-valve rather than a passive gateway, shifting from continuous raw data streaming to event-driven semantic updates. This architectural transition is expected to substantially reduce bandwidth; Section 4.6 quantifies this analytically.

Table 1. Comparison of the proposed E-DTNCS architecture against state-of-the-art digital twin implementations.

Approach	Deployment Architecture	AI Methodology	Telemetry Transmission	Bandwidth Optimization
Lawal et al. [17]	Cloud-Edge Hierarchy	PINN	Continuous Batching	Moderate
Kayan et al. [21] (CASPER)	Benchmark dataset (sensor node)	1D-CNN anomaly detection	Not the focus	Not evaluated
Niu et al. [22]	Centralized Cloud	1D-CNN	Continuous Streaming	None
Proposed E-DTNCS	Edge node	PINC-noBN	Event-driven (heartbeat + burst)	Design Consideration

1.4. Contributions

To address the safety, efficiency, and scalability difficulties of self-governing digital twins in complex industrial networks, this study proposes the Edge-Enabled digital twin Networked Control System (E-DTNCS). This framework integrates compact edge inference—categorized under the "TinyML" [23] paradigm—directly adjacent to the physical asset. A compact Parallel Inception 1D-Convolutional Neural Network (PINC) is evaluated as the edge classifier for a Universal Robots UR3e; its working memory exceeds the 0.06 MB SRAM of the ARM Cortex-M4 sensor node used for data acquisition (Section 5.3).

The explicit contributions of this paper are formulated as follows:

1. Architecture (E-DTNCS): The formulation of a decentralized digital twin architecture that decouples high-frequency data generation from the network routing bus: a low-rate heartbeat in normal operation and raw bursts on detected anomalies, with an analytical bandwidth model that includes false-positive bursts.
2. Compact two-stage classifier (PINC-noBN): The design of a compact two-stage channel-wise inception 1D-CNN (37,154 parameters) with an ablation against a single-stage variant, evaluated against CASPER-style CNNs and an InceptionTime-type network over three seeds. It uses 22% of the best baseline's parameters (kernel size 50) but achieves slightly lower accuracy and requires more activation memory than the CASPER-style CNNs.
3. Documented evaluation protocol: A chronological, seeded evaluation on the CASPER benchmark with all variants disclosed, the correction of the original experiment (Section 2.6) and an analysis of the offset between labels and signal (Section 4.3); results are reported both as labeled and with the transition zones excluded.

Empirical validation on the public CASPER benchmark (20 Hz, chronological split, three seeds) shows that the PINC-noBN classifier reaches a test accuracy of 0.86 ± 0.02 with 37,154 parameters when the labels are used as given (0.97 outside the zones around the class transitions, where labels and signal are offset), slightly below a CASPER-style CNN with kernel size 50 (0.88 ± 0.01 and 0.99; 169,794 parameters); no superiority is claimed. An analytical projection that includes false-positive bursts indicates bandwidth reductions of about 85–92% (labels as given) and 90–99.8% (transition zones excluded) at anomaly prevalences between 10% and 0.1%. These findings indicate that edge-side classification can reduce the protocol overhead bottleneck in the projected 1000 Hz scenario; however, the projection and the current implementation's memory requirement (at least 2 MB of activations) still need experimental validation.

The remainder of this paper is structured as follows. Section 2 details the CASPER dataset and its pre-processing, the two-stage PINC classifier, the training protocol and

baselines, the model selection procedure, the correction of the original experiment, and the protocol-overhead model. Section 3 presents the E-DTNCS architecture layer by layer. Section 4 reports the results, including evaluation metrics, training behavior, alignment, classification performance, edge footprint, and projected bandwidth. Section 5 discusses the design choices, the operational significance of false alarms, and the study's limitations. Section 6 concludes the paper.

2. Materials and Methods

To rigorously address the bandwidth limitations inherent in centralized digital twin architectures, this section describes the physical testbed and dataset, the data pre-processing pipeline, the mathematical formulation of the two-stage machine learning model, and the proposed E-DTNCS architecture.

2.1. Physical Setup and the CASPER Dataset

To reproduce and validate the proposed architecture, we use the publicly available CASPER (Context-Aware IoT Anomaly Detection System for Industrial Robotic Arms) dataset. The experimental testbed centers on a Universal Robots UR3e, a 6-Degree-of-Freedom (DoF) collaborative manipulator commonly employed in smart manufacturing for precise assembly and pick-and-place operations. The basic kinematics were captured using an external Arduino Nicla Sense ME (Arduino S.r.l., Monza, Italy) edge-computing sensor node, rigidly mounted near the end-effector (Tool Center Point) of the UR3e to maximize sensitivity to multi-axis kinematic deviations. This device features a 64 MHz ARM Cortex-M4 microcontroller and a highly constrained 0.06 MB of SRAM, representing the true edge of industrial computing where memory constraints could strictly limit software architecture. We used the sensor-node recording (Nicla IMU) together with the robot controller log. We derive window labels from the controller's anomaly-state flag. As the CASPER authors note, the two recordings are not hardware-synchronized, and the sensor board does not run at exactly 20 Hz, so the recordings drift apart over time [21]; we aligned them with a fixed time offset, which cannot compensate for drift. We discarded the first approximately one million samples, which contain no anomalies; we used the remaining 728,006 samples (about 10 h at 20 Hz).

The sensor node uses an integrated 6-axis Inertial Measurement Unit (IMU) to generate a continuous telemetry stream, capturing 3-axis linear acceleration (a_x, a_y, a_z) and 3-axis angular velocity $(\omega_x, \omega_y, \omega_z)$ at 20 Hz. While industrial predictive maintenance typically requires high-frequency sampling (e.g., 1000 Hz to 12,000 Hz) to detect transient micro-vibrations [24], the 20 Hz CASPER dataset instead benchmarks macro-kinematic anomalies over longer operational cycles. A 700×6 tensor has the same memory footprint at 20 Hz and at 1000 Hz. This does not imply equal signal content or detectability: we report all accuracy results in this paper at 20 Hz for macro-kinematic anomalies, and we still need to validate performance at 1000 Hz.

CASPER is labeled for binary classification ($K = 2$). The normal class is the UR3e executing its programmed cyclic pick-and-place trajectory at nominal speed. In the anomaly class, the joint speed was changed relative to nominal: the recording contains eight anomaly episodes of 36 min each, with speed changes of +10, +35, +65, +100, −50, −5, +20 and −25%. The task is therefore to detect tempo changes in an otherwise unchanged trajectory; the benchmark contains no mechanical faults or collisions.

2.2. Data Pre-Processing and Temporal Windowing

The continuous raw IMU data stream generated at the edge is systematically segmented into temporal windows to capture sufficient functional context for the neural network to

identify mechanical anomalies. Figure 1 illustrates the specific process used in this study. The raw multivariate sensor stream is defined as a matrix $\mathbf{S} \in \mathbb{R}^{T \times C}$, where T denotes the total number of recorded time-steps and $C = 6$ represents the individual sensor channels (three for acceleration, three for angular velocity).

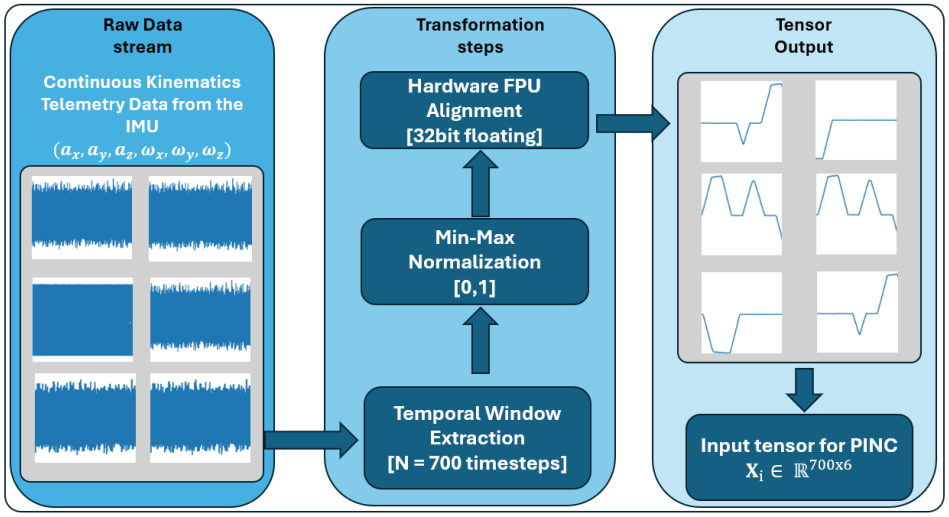


Figure 1. The localized data pre-processing pipeline. Continuous telemetry is ingested, segmented into windows ($N = 700$ time-steps), subjected to Min-Max normalization, and cast to 32-bit floating-point precision, forming the input matrix $\mathbf{X}_i \in \mathbb{R}^{700 \times 6}$.

To ensure equal feature weighting across differing sensor scales (m/s^2 versus $^\circ/s$) and to optimize gradient descent convergence, Min-Max normalization is applied independently to each channel $c \in \{1, \ldots, 6\}$:

$$\mathbf{S}_{norm}^{(c)} = \frac{\mathbf{S}^{(c)} - \min(\mathbf{S}^{(c)})}{\max(\mathbf{S}^{(c)}) - \min(\mathbf{S}^{(c)})} \tag{1}$$

To strictly prevent information leakage, we computed the min and max scaling parameters exclusively from the training dataset. We then fixed these constants and applied them to both the validation set and the live edge inference pipeline. We explicitly cast the normalized data to 32-bit floating-point precision (`float32`), matching the ARM Cortex-M4 microcontroller's native Hardware Floating-Point Unit (FPU) and avoiding precision-conversion overhead during real-time edge inference. Because the constants come only from the training block, validation and test values can fall outside $[0, 1]$.

After normalization, we use a sliding-window technique to extract discrete matrices from the continuous stream. At the dataset's native sampling frequency of $f_s = 20$ Hz ($\Delta t = 50$ ms), a standardized window size of $N = 700$ time-steps (encompassing a 35.0-s trajectory cycle) is established. The window length is of the order of one pick-and-place cycle. We extract windows with a stride of 10 samples (0.5 s), so consecutive windows overlap by 98.6%, and each window carries the label of its last time-step.

Assuming a stride length of s, the i-th extracted window is defined as:

$$\mathbf{X}_i = \mathbf{S}_{norm}[i \cdot s : i \cdot s + N, :] \quad \in \mathbb{R}^{700 \times 6} \tag{2}$$

To prevent temporal leakage, we split the recording chronologically into 70/15/15 training, validation, and test blocks before windowing; no window crosses a block boundary. Because the stride is much shorter than the window, neighboring windows are strongly correlated, so the number of windows (training 50,891; validation 10,851; test 10,851) overstates the number of independent samples. Class counts (normal/anomaly) are 28,810/22,081 for training, 4320/6531 for validation, and 5041/5810 for test. The test block (about 91 min)

contains only about two anomaly episodes, so test metrics carry large uncertainty that window counts do not reflect.

2.3. Two-Stage Parallel Inception 1D-CNN (PINC)

Standard sequential CNNs can be memory-hungry for edge nodes. We therefore examine a compact design in which we first process each IMU channel separately and fuse the channels only in a second stage. We motivate this design but do not claim it is optimal; we evaluate the second stage against a single-stage variant in Section 4.4.

To keep the parameter count small (37,154 in the final variant), this study proposes a compact Parallel Inception design (PINC) for edge-class devices with more memory than the 0.06 MB Nicla used for acquisition (Section 5.3). The architecture is shown in Figure 2.

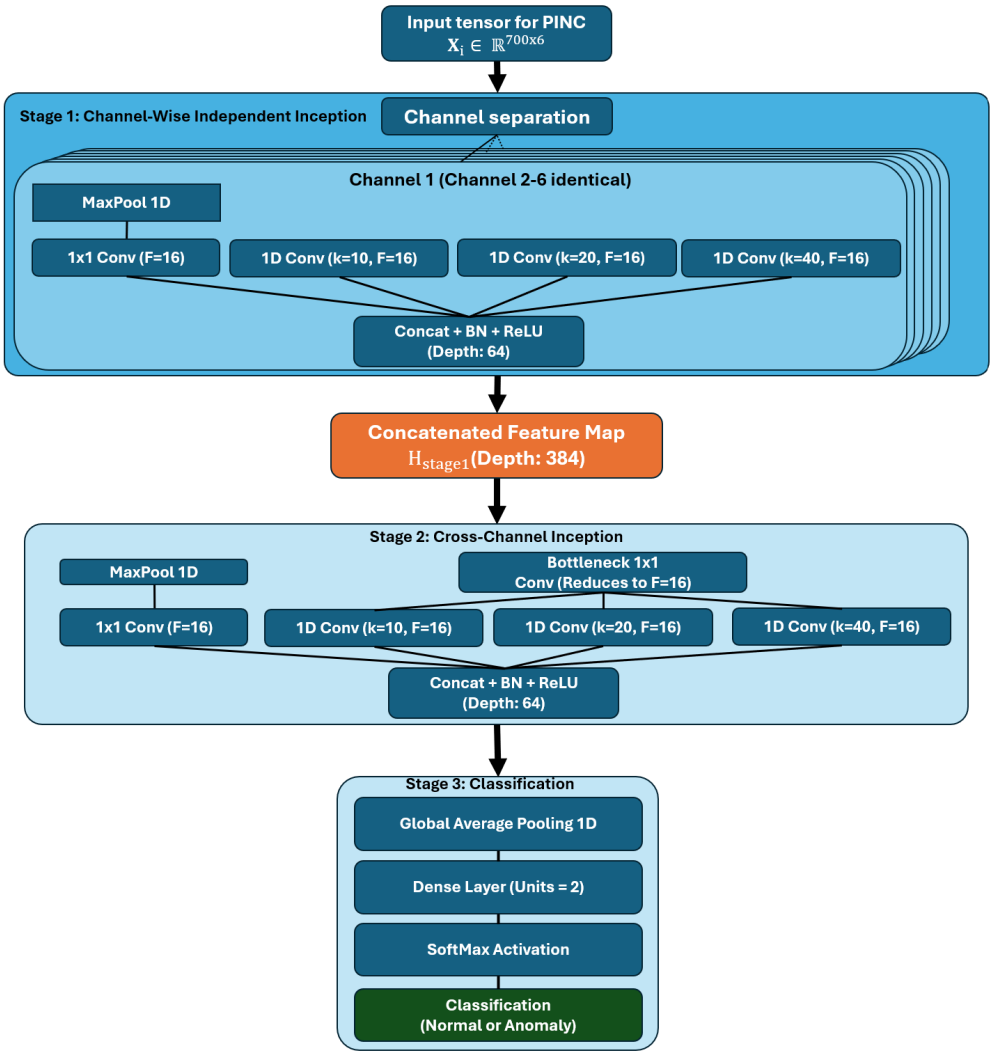


Figure 2. The proposed Two-Stage Parallel Inception 1D-CNN (PINC) architecture. Stage 1 enforces channel-wise independence, processing each of the 6 IMU axes through isolated inception blocks. Stage 2 executes cross-channel fusion via a secondary inception block, followed by Global Average Pooling (GAP) and Softmax classification. The figure shows the design with batch normalization (Concat + BN + ReLU); the final variant (PINC-noBN) is identical except that the batch-normalization layers are omitted.

2.3.1. Stage 1: Channel-Wise Independent Inception

Rather than immediately convolving all $C = 6$ channels, the input window $\mathbf{X}$ is sliced along the feature axis into individual 1D vectors $\mathbf{x}^{(c)} \in \mathbb{R}^{700\times1}$.Channel c is selected by slicing the input window along the feature axis, so each of the six branches sees exactly one channel.

For each independent channel $c \in \{1, 2, \ldots, 6\}$, an inception block is applied independently. A spatial MaxPooling operation is applied directly to $\mathbf{x}^{(c)}$ to generate a pooled feature map $\mathbf{z}_{pool}^{(c)}$.

In parallel with the max pooling, three independent 1D convolutional paths use multiscale kernel sizes $k \in \{10, 20, 40\}$ with stride 1 and `same` padding. At the benchmark dataset's 20 Hz sampling rate, these kernel sizes correspond to 0.5 s, 1.0 s, and 2.0 s physical time windows. This lets the network capture both immediate kinematic deviations ($k = 10$) and longer, multi-second operational drifts ($k = 40$). For path $p \in \{1, 2, 3\}$ applied to channel c:

$$\mathbf{z}_p^{(c)} = \mathbf{x}^{(c)} * \mathbf{W}_p \tag{3}$$

where $*$ denotes the 1D convolution operation. A fourth path applies a 1×1 convolution to the max-pooled tensor to match filter depths:

$$\mathbf{z}_4^{(c)} = \mathbf{z}_{pool}^{(c)} * \mathbf{W}_{1\times1} \tag{4}$$

All convolutions are bias-free.

A ReLU follows each convolution. The four outputs are concatenated and, in the original design (PINC), batch-normalized and passed through a further ReLU; the final variant (PINC-noBN) omits batch normalization.

$$\mathbf{Z}_{concat}^{(c)} = \sigma\Big(\mathcal{BN}\big([\mathbf{z}_1^{(c)}, \mathbf{z}_2^{(c)}, \mathbf{z}_3^{(c)}, \mathbf{z}_4^{(c)}]\big)\Big) \quad \text{(PINC)}, \qquad \mathbf{Z}_{concat}^{(c)} = [\mathbf{z}_1^{(c)}, \mathbf{z}_2^{(c)}, \mathbf{z}_3^{(c)}, \mathbf{z}_4^{(c)}] \quad \text{(PINC-noBN)} \tag{5}$$

where $\mathbf{z}_p^{(c)} = \sigma(\mathbf{x}^{(c)} * \mathbf{W}_p)$. Max-pooling uses a 3-point window with stride 1 and same padding. Stage 1 has no bottleneck layer; Stage 2 has one.

2.3.2. Stage 2: Cross-Channel Inception and Fusion

To fuse independent kinematic features into a cohesive understanding of the robot's physical state, we pass $\mathbf{H}_{stage1}$ through a second master inception block. To reduce the dimensionality before fusion, a bottleneck mechanism is first applied using a 1×1 convolution, yielding $\mathbf{H}_{bottleneck}$. Concurrently, a secondary max-pooling pass $\mathbf{H}_{pool}$ is extracted directly from $\mathbf{H}_{stage1}$.

Following the exact parallel structure of Stage 1, three new 1D convolutions ($k \in \{10, 20, 40\}$) operate on the bottleneck tensor, and a 1×1 convolution operates on the pooled tensor. These four paths are concatenated, batch-normalized (in PINC, not in PINC-noBN), and passed through ReLU to produce the final spatial feature map $\mathbf{Y}_{fusion}$, representing the complex cross-axis kinematics.

2.3.3. Stage 3: GAP and Binary Classification

To aggressively reduce dimensionality before classification and minimize the fully connected parameter footprint, we apply Global Average Pooling 1D (GAP) across the temporal axis, yielding a dense vector $\mathbf{v}$. Finally, $\mathbf{v}$ is passed through a single fully connected dense layer to output a probability distribution over the $K = 2$ predefined operational states (Normal vs. Anomaly) utilizing the Softmax activation function:

$$\hat{\mathbf{y}} = \text{Softmax}(\mathbf{v} \cdot \mathbf{W}_{dense} + \mathbf{b}_{dense}) \tag{6}$$

2.4. Training Protocol and Optimization

We implemented all models in Keras (version 3.13.2) and trained them with Adam and sparse categorical cross-entropy (batch size 32, 50 epochs). The learning rate decays exponentially, $\eta(t) = \eta_0 \gamma^{t/\tau}$ with $\gamma = 0.1$ and $\tau = 20\lceil N_{windows}/32 \rceil$ (continuous decay, one order of magnitude per 20 epochs). The initial rate was set per model after pilot runs in

which some models collapsed to the majority class: 0.01 for PINC; 10^{-3} for PINC-noBN, the CASPER-style CNN with kernel 5, the single-stage variant, and InceptionTime; 10^{-4} for the CASPER-style CNN with kernel 50. Baselines were not otherwise tuned. We evaluate the model on the test block after the final epoch; we save a checkpoint at the best validation accuracy but do not use it for test results. We trained each configuration with three seeds (1, 2, 3); runs are seeded but not bit-reproducible on GPU. We accidentally ran PINC-noBN with seed 2 twice; the two runs reached test accuracies of 0.87 and 0.84, and we use the run whose files were saved last (0.84).

The baselines are defined as follows. The CASPER-style CNN consists of four Conv1D layers with 32 filters and kernel size k (5 or 50, ReLU), a max-pooling layer of size 2 after every second convolution, flattening, and a dense softmax layer (27,266 and 169,794 parameters for $k = 5$ and $k = 50$). This network is a supervised classifier built on the stacked-convolution design of the 1D-CNN in the CASPER paper [21], and its kernel sizes (5 and 50), filter count (32) and pooling size (2) lie within the grid-search values reported there. This is our implementation, not the original model: the original uses two convolution layers on windows of 755 time-steps, predicts the next time-step, and flags anomalies by thresholding the prediction error, whereas ours has four convolution layers and classifies each window directly. The InceptionTime-type network has six inception modules with 32 filters, kernel sizes 10, 20, and 40, a bottleneck of 32, and residual connections every three modules, followed by global average pooling and a dense softmax layer (492,930 parameters). The variants labeled gyroscope ω_z use the same architectures but receive only the gyroscope ω_z channel (channel 5): in PINC, every branch gets that channel, and in the CASPER-style CNNs it is the only input channel. In the single-stage ablation, we remove the Stage 2 block, and the 384-channel Stage 1 output feeds directly into the pooling and dense layers (7586 parameters). In the tables, the models are identified as follows: PINC is the six-channel PINC with batch normalization; PINC_noBN is the final variant without batch normalization (written PINC-noBN in the text); PINC_noBN_1stage is PINC_noBN without Stage 2; PINC_flat is PINC with a flattened instead of globally pooled classification head (one seed); PINC_GyroZ is PINC with all six branches receiving gyroscope ω_z, which corresponds to the original code; CASPER_K5 and CASPER_K50 are the CASPER-style CNNs with kernel size 5 and 50; CASPER_K5_GyroZ and CASPER_K50_GyroZ are the same networks using only gyroscope ω_z; InceptionTime is the InceptionTime-type network.

2.5. Model Selection

The chronological split into training, validation, and test blocks was fixed before any training, and no window is shared between blocks. The design choices, including the removal of batch normalization, the learning rates, and the choice of PINC-noBN as the final variant, were made iteratively from preliminary runs, and these runs included test-block results. All examined variants are reported in Table 2. The validation block was used to monitor training and not to choose epochs or checkpoints for the reported results. The test results of PINC-noBN are therefore optimistic estimates of performance on new data. The validation accuracy of the same models (Section 4.2) provides a second view from a different block, though with different label offsets. The CASPER dataset consists of a single recording from one robot, and all three blocks come from it, so the blocks share the robot, the recording session, and the labeling procedure. A fully independent evaluation therefore requires additional recordings, ideally from other robots and operating conditions, which are not available in this dataset. This is left for future work.

Table 2. Test performance of the evaluated models as labeled (Condition A; mean ± SD over three seeds; $N = 10{,}851$ windows from about two anomaly episodes). Precision, recall, and F_1 refer to the anomaly class. Model identifiers are defined in Section 2.4; PINC_flat was run with one seed, so no SD is given.

Model	Parameters	Test Acc.	Anomaly P	Anomaly R	Anomaly F_1	ROC-AUC
PINC	38,946	0.665 ± 0.044	0.976 ± 0.023	0.384 ± 0.089	0.547 ± 0.094	0.865 ± 0.014
PINC_noBN	37,154	0.858 ± 0.015	0.927 ± 0.002	0.799 ± 0.033	0.858 ± 0.019	0.897 ± 0.003
PINC_noBN_1stage	7586	0.562 ± 0.006	0.895 ± 0.017	0.205 ± 0.014	0.334 ± 0.018	0.742 ± 0.035
CASPER_K5	27,266	0.816 ± 0.091	0.925 ± 0.003	0.714 ± 0.182	0.797 ± 0.126	0.887 ± 0.017
CASPER_K50	169,794	0.881 ± 0.009	0.926 ± 0.001	0.845 ± 0.018	0.884 ± 0.010	0.898 ± 0.007
InceptionTime	492,930	0.632 ± 0.062	0.951 ± 0.023	0.330 ± 0.119	0.482 ± 0.133	0.846 ± 0.053
PINC_GyroZ	38,946	0.587 ± 0.020	0.933 ± 0.070	0.246 ± 0.023	0.389 ± 0.033	0.487 ± 0.232
PINC_flat	47,778	0.662	0.961	0.385	0.549	0.873
CASPER_K5_GyroZ	26,466	0.746 ± 0.112	0.927 ± 0.005	0.572 ± 0.227	0.690 ± 0.166	0.882 ± 0.014
CASPER_K50_GyroZ	161,794	0.877 ± 0.012	0.925 ± 0.002	0.839 ± 0.026	0.880 ± 0.014	0.895 ± 0.007

2.6. Correction of the Original Submission

During this revision, we found an implementation error in the code used for the original submission. We implemented the channel selection for the six inception branches using a Lambda layer whose channel index was captured by reference inside a loop. As a result, all six branches received the same input channel (gyroscope ω_z) instead of one channel each, and the model trained in the original run did not match the architecture described in this section. We fixed the error, verified that each branch receives its own channel, and re-evaluated all models under the single protocol described above. The corrected six-channel PINC with batch normalization reaches a test accuracy of 0.66 ± 0.04, and a configuration in which all branches receive gyroscope ω_z, which corresponds to the original code, reaches 0.59 ± 0.02. We could not reproduce the originally reported accuracy of 86.19% with either configuration under the 50-epoch protocol used here (the original run used 100 epochs and no fixed seed). The results in this revision replace those of the original submission.

2.7. Network Architecture and Protocol Overhead

In conventional DT architectures, streaming continuous raw IMU data via protocols like RTDE at 1000 Hz incurs massive protocol overhead (O_{net}). A single 6-axis `float32` reading yields a raw payload (P_{raw}) of 24 bytes. Standard industrial IPv4 requires a 20-byte IP header, a 20-byte TCP header, and an 18-byte Ethernet frame:

$$S_{packet} = P_{raw} + O_{net} = 24 \text{ bytes} + 58 \text{ bytes} = 82 \text{ bytes} \quad (7)$$

At 1000 Hz, roughly $> 70.7\%$ of the consumed bandwidth is entirely wasted on non-data routing headers. The E-DTNCS operates strictly as an event-driven semantic filter. In normal operation, the node sends a low-rate heartbeat (1 Hz); when it detects an anomaly, it switches to a raw-data burst. The trigger rule that suppresses isolated false detections is specified in Section 4.6. The 1000 Hz figures in this section are protocol arithmetic for a projected deployment scenario, not measurements.

3. The E-DTNCS Architecture

To address the protocol-overhead bottleneck systematically, the proposed Edge-Enabled digital twin Networked Control System (E-DTNCS) aligns with the standardized multi-layer digital twin reference architecture [25]. Traditional IoT deployments often suffer from network fragmentation because they fail to properly delegate computational responsibilities across the cloud-to-edge continuum [26]. By adopting a formal hierarchical model spanning the Physical (L0), Data (L1), and Integration (L2) layers [27], the

framework decouples the high-frequency physical data generation rate from the network communication rate. The complete data flow of this architecture is illustrated in Figure 3.

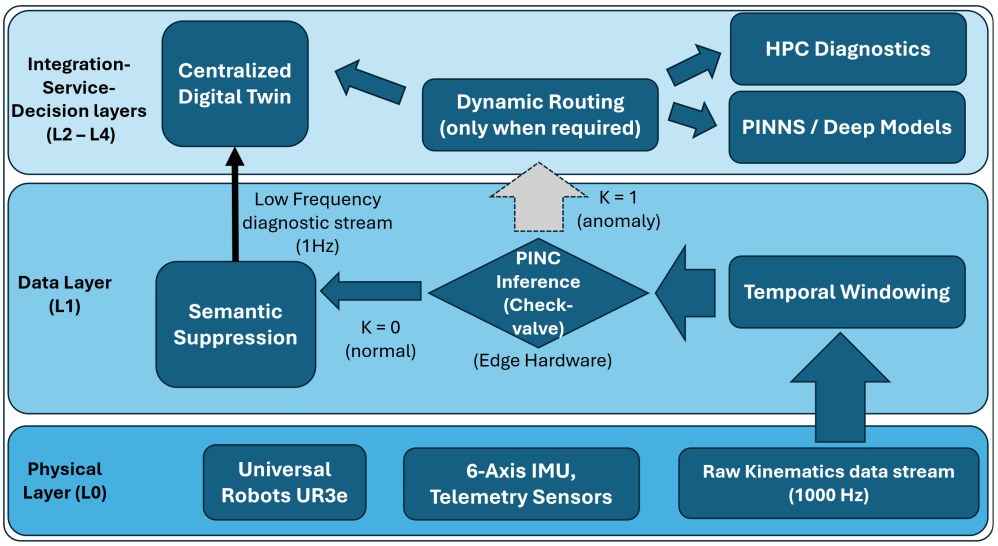


Figure 3. The proposed Composable E-DTNCS Architecture. Raw kinematics (1000 Hz in the projected scenario) are generated at the Physical Layer (L0) and processed by the PINC model at the Semantic Edge Layer (L1). The localized check valve suppresses nominal data, allowing only event-driven semantic state updates to ascend (L2–L4).

3.1. L0: The Physical Layer

The Physical Layer (L0) encompasses the cyber-physical assets, sensors, and actuators residing on the shop floor. In the E-DTNCS, the Universal Robots UR3e manipulator and its rigidly coupled 6-axis Inertial Measurement Unit (IMU) instantiate L0. This layer captures raw measurements of physical reality, generating a continuous, uncompressed stream of multivariate kinematic arrays (acceleration and angular velocity) at 1000 Hz in the projected deployment scenario (the CASPER data used for evaluation were sampled at 20 Hz). In a legacy architecture, L0 would push this massive telemetry volume upward, possibly saturating the L2 network switch buffers.

3.2. L1: The Data Layer (Semantic Edge Inference)

The critical departure from legacy architectures occurs at the Data Layer (L1). Rather than acting as a passive data lake or simple protocol gateway, the E-DTNCS instantiates L1 as an active, intelligent edge computing node (targeted for edge devices; the current implementation requires more memory than the Cortex-M4 sensor node offers; see Section 5.3). This layer ingests the sensor stream from L0 (1000 Hz in the projected scenario) and immediately structures it into temporal windows ($N = 700$) for live execution through the two-stage PINC model.

By deploying TinyML inference directly at L1, the system creates a semantic check valve. In nominal operation ($K = 0$) the node withholds the raw stream. It sends only a low-rate heartbeat, following the principle that raw physical telemetry should be semantically translated before it consumes factory network bandwidth.

Data ingested from L0 enters a localized pre-processing pipeline, where it undergoes real-time Min-Max normalization and is cast to 32-bit floating-point precision to align with the hardware's Floating-Point Unit (FPU). The continuous stream is segmented into temporal windows of 700 time-steps ($N = 700$).

Crucially, migrating deep learning to the edge requires strict memory management. The compiled PINC-noBN model has 37,154 parameters (about 0.15 MB in 32-bit floating-

point precision; PINC with batch normalization has 38,946 parameters, including batch-normalization statistics). These weights fit within the 0.5 MB Flash memory of a Cortex-M4 class device. However, executing the forward pass in float32 requires at least 1.05 MB for the Stage 1 activation map alone (700 × 384 values); peak activation memory ranges from 2.05 to 8.1 MB . Therefore, although we acquired the data on an ultra-constrained 0.06 MB node (the Nicla Sense ME), the current unoptimized float32 inference requires an edge device with substantially more memory. To separate what we did from what we propose: the Nicla Sense ME can currently be used to acquire the data, and we obtained all classification results offline in float32. The intended inference platform is an edge device with several megabytes of RAM. Running the model on the sensor node itself requires the changes listed in Section 5.3.

The PINC executes this forward pass locally, classifying the 700-time-step window into discrete operational states (Normal vs. Anomaly). If the operational state remains stable, the edge node withholds the raw telemetry and sends only the heartbeat.

3.3. L2 and Beyond: Integration and Orchestration

The Integration Layer (L2) manages middleware and communication protocols that bridge the factory floor with centralized IT infrastructure. In the E-DTNCS, L2 is completely insulated from L0's high-frequency data generation rate. L1 communicates with L2 only via event-driven TCP/IP state updates.

When L1 classifies a kinematic anomaly ($K = 1$), it pushes a lightweight semantic payload to L2, which then orchestrates a dynamic telemetry fallback to stream the burst data. This architectural isolation lets the higher Service (L3) and Decision (L4) layers, which reside in the centralized digital twin, receive event-driven insights, with the continuous protocol-overhead queuing delay expected to be reduced (not measured).

When the Semantic Edge Layer detects a transition in the physical asset's state (e.g., a shift from standard operation to a kinematic anomaly such as the speed changes in the CASPER benchmark), it initiates a lightweight TCP/IP transmission to L2. This payload contains only the semantic identifier of the new state and a synchronized timestamp, rather than the bulky 24-byte floating-point array of the raw IMU reading. Consequently, L2 is expected to receive event-driven updates about cyber-physical disturbances with less queuing delay from continuous RTDE protocol overhead, but we did not measure this expectation.

Unlike traditional gateway-centric frameworks that simply batch or compress data before cloud transmission, the E-DTNCS deploys intelligent semantic translation directly at the point of data ingestion—the extreme edge.

4. Results

We evaluate algorithm performance, edge hardware capabilities, and training stability using the metrics below, followed by an empirical analysis of condition classification and an analytical scalability projection for multi-agent environments. We report all results as the mean ± standard deviation over three training runs with seeds 1, 2, and 3. We evaluate the model after the final epoch on the test block. We seed the runs but cannot achieve bit-level reproducibility on GPU. Because the labels and the signal are not aligned near the class transitions, results are reported under two conditions defined in Section 4.3.

4.1. Evaluation Metrics

To evaluate binary classification performance ($K = 2$) for anomaly detection, we used standard statistical metrics, including accuracy (Acc), Precision (P), Recall (R), and the F1-Score (F_1). In the context of industrial condition monitoring, these are formally defined as:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}, \quad P = \frac{TP}{TP + FP} \tag{8}$$

$$R = \frac{TP}{TP + FN}, \quad F_1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{9}$$

where TP, TN, FP, and FN represent True Positives, True Negatives, False Positives, and False Negatives, respectively. Because the test block contains only about two independent anomaly episodes, seed-based standard deviations reflect only training variability and understate uncertainty about new episodes. In addition, the false-positive rate $FPR = FP/(FP + TN)$ and the area under the ROC curve (ROC-AUC) are reported.

4.2. Training Convergence Dynamics

We evaluated the learning trajectory of the two-stage PINC models over 50 epochs with a batch size of 32. The custom exponential decay schedule used a decay factor $\gamma = 0.1$, applied dynamically based on dataset volume, and an initial learning rate that was set per model (0.01 for PINC and 0.001 for PINC-noBN; see Section 2.4).

Figure 4 shows the training curves of PINC-noBN. Training accuracy reaches about 0.96 with a training loss of about 0.12–0.13, while final-epoch validation accuracy is 0.81–0.86 and validation loss is 0.48–0.58. There is therefore a clear generalization gap, and the model is not claimed to be free of overfitting. Part of the gap is due to the offset between labels and signal described in Section 4.3, which also affects the validation block. The original PINC with batch normalization fits the training data almost perfectly (training accuracy 1.000, training loss below 0.001), and its validation loss is lowest within the first four epochs and ends at 4.7 to 7.6 across the three seeds, so it overfits strongly. Removing batch normalization was the largest single change in test accuracy among the variants examined (0.66 ± 0.04 vs. 0.86 ± 0.02), and the gap remains when the transition zones are excluded (Table 3). We did not investigate why batch normalization hurts here. A plausible, untested explanation is that its statistics differ between the training block and later blocks of the recording. Calibration of the predicted probabilities was not assessed.

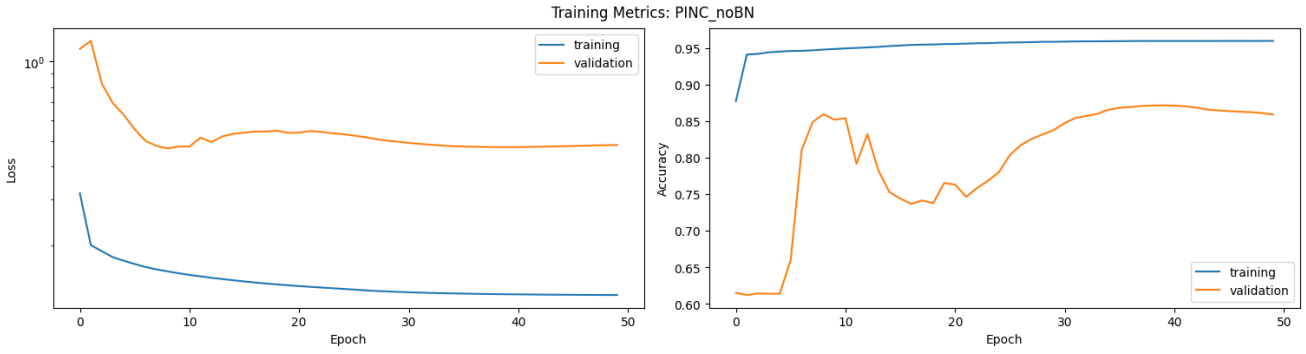


Figure 4. Training and validation convergence curves over 50 epochs for Seed 1. (**Left**): Logarithmic categorical cross-entropy loss demonstrating the effect of the exponential decay schedule. (**Right**): Classification accuracy.

Table 3. Test performance with the transition zones excluded (Condition B; mean ± SD over three seeds; 9138 windows). The zone width was fixed before the metrics were computed and the zones were derived from the test signal. Precision, recall, and F_1 refer to the anomaly class. Model identifiers are defined in Section 2.4; PINC_flat was run with one seed, so no SD is given.

Model	Test Acc.	Anomaly P	Anomaly R	Anomaly F_1	FPR	ROC-AUC
PINC	0.706 ± 0.053	0.988 ± 0.008	0.438 ± 0.107	0.602 ± 0.106	0.006 ± 0.005	0.950 ± 0.019
PINC_noBN	0.965 ± 0.020	0.999 ± 0.001	0.933 ± 0.040	0.965 ± 0.021	0.001 ± 0.001	0.998 ± 0.001
PINC_noBN_1stage	0.585 ± 0.008	0.902 ± 0.015	0.223 ± 0.016	0.358 ± 0.020	0.026 ± 0.005	0.796 ± 0.041
CASPER_K5	0.910 ± 0.117	0.993 ± 0.011	0.831 ± 0.220	0.894 ± 0.145	0.005 ± 0.007	0.985 ± 0.025
CASPER_K50	0.992 ± 0.010	1.000 ± 0.000	0.985 ± 0.019	0.992 ± 0.010	0.000 ± 0.000	1.000 ± 0.000
InceptionTime	0.675 ± 0.074	0.993 ± 0.005	0.375 ± 0.143	0.534 ± 0.152	0.003 ± 0.001	0.935 ± 0.074
PINC_GyroZ	0.617 ± 0.024	0.950 ± 0.080	0.276 ± 0.029	0.427 ± 0.040	0.016 ± 0.026	0.472 ± 0.302
PINC_flat	0.709	0.997	0.441	0.612	0.001	0.957
CASPER_K5_GyroZ	0.820 ± 0.144	0.986 ± 0.012	0.661 ± 0.272	0.772 ± 0.191	0.008 ± 0.007	0.978 ± 0.021
CASPER_K50_GyroZ	0.989 ± 0.015	0.999 ± 0.001	0.980 ± 0.030	0.989 ± 0.015	0.001 ± 0.001	1.000 ± 0.000

4.3. Alignment of Labels and Signal

Inspection of the prediction errors showed they cluster around the class transitions in the test block. The block contains three labeled transitions (windows 1490, 5810 and 10,130, shown in Figure 5). At each of them, the variability of the gyroscope signal changes several minutes before the labeled transition, and the predictions of the classifiers change at the same point as the signal, not at the label. The offset was estimated from the signal alone, without model outputs: for each labeled transition, the mean gyroscope variability (standard deviation over the 700 samples of a window, averaged over the three gyroscope channels) over the 300 windows before and after a candidate position was compared for candidates between 800 windows before and 100 windows after the transition, and the position with the largest difference was taken. The estimated offsets are 390, 410, and 460 windows (3.3 to 3.8 min) in the test block and 280 and 370 windows in the validation block. The first validation estimate has weak contrast, and the resolution is 10 windows. The offset thus appears to grow over the recording. A likely cause is the drift between the two recordings: the CASPER authors report that the sensor board does not run at exactly 20 Hz [21], and a fixed time offset cannot compensate for a drift (Section 2.1). We have not verified this in our data. We could not check the training block in the same way because the stored training windows are shuffled.

All models were trained and evaluated with the same labels, so comparisons between models use identical labeling, but absolute accuracy and false-positive rates near the transitions are affected. Two evaluation conditions are therefore reported. Condition A uses all test windows with the original labels and is the primary, pessimistic result. Condition B removes the windows from 500 before to 70 after each labeled transition (1713 of 10,851 windows). The zone covers the largest estimated offset and the length of one classification window (70 strides). We fixed the zone width before computing the metrics. Condition B uses the test signal but no model output. It is exploratory and an optimistic bound. A third variant, which moved the test labels to the estimated signal transitions, gave accuracies within 0.02 of Condition B for all models and is not tabulated. Condition B should therefore be read as an upper bound on how much error stems from label misalignment, not as the expected performance on new data. A more reliable estimate requires independently corrected labels, which we consider the most important next step.

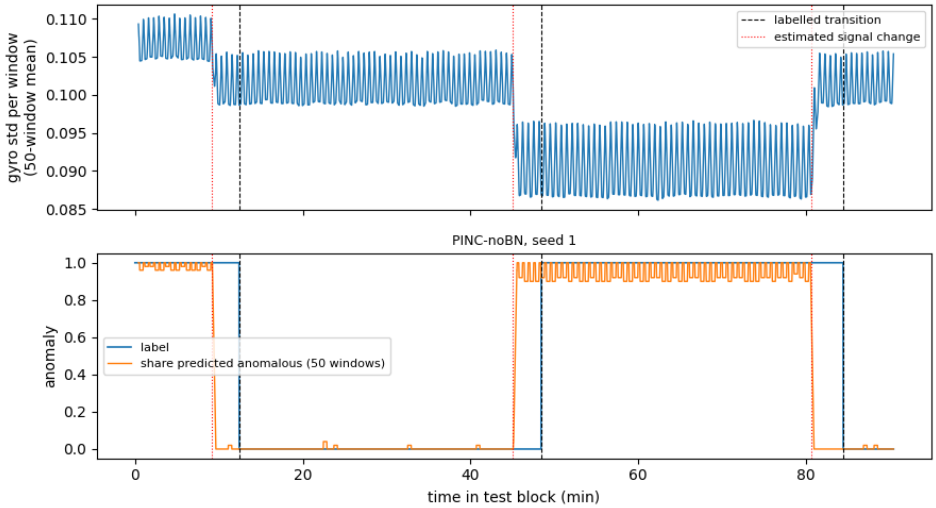


Figure 5. Test block: gyroscope variability per window (**top**) and labels with the share of windows predicted anomalous by PINC-noBN, seed 1 (**bottom**), both as 50-window moving means. The signal level changes about 3.3 to 3.8 min before each labeled class transition (red dotted: estimated from the signal alone; black dashed: labeled), and the predictions change at the same point as the signal.

4.4. Classification Performance

We split the recording chronologically into 70/15/15 training, validation, and test blocks before windowing (Section 2.2). The test block contains $N = 10{,}851$ windows (stride 10 samples, 98.6% overlap) of which 5041 are normal and 5810 anomalous. These windows come from about two independent anomaly episodes. We compared PINC-noBN with a CASPER-style 1D-CNN (kernel sizes 5 and 50), an InceptionTime-type network, PINC with batch normalization, a single-stage ablation, a variant with a flattened instead of globally pooled classification head (one seed), and versions of PINC and the CASPER-style CNNs that use only gyroscope ω_z. All models used the same data, optimizer, batch size, epochs, and final-epoch evaluation. We set initial learning rates per model (Section 2.4) and tuned none of the baselines further. We did not evaluate temporal convolutional networks, residual CNNs, or depthwise-separable CNNs and leave them for future work.

Table 2 summarizes test performance as labeled (Condition A), and Table 3 summarizes performance with the transition zones excluded (Condition B). Under Condition A, PINC-noBN reaches a test accuracy of 0.86 ± 0.02 (anomaly recall 0.80 ± 0.03, FPR 0.073 ± 0.005, ROC-AUC 0.897 ± 0.003) with 37,154 parameters. The CASPER-style CNN with kernel size 50 reaches 0.88 ± 0.01 with 169,794 parameters, the kernel-5 variant 0.82 ± 0.09 with 27,266 parameters (one of three seeds converged to a worse solution), InceptionTime 0.63 ± 0.06 with 492,930 parameters and PINC with batch normalization 0.66 ± 0.04. Under Condition B, PINC-noBN reaches 0.965 ± 0.020 (recall 0.93, FPR 0.001), the kernel-50 CNN 0.992 ± 0.010, and the kernel-5 CNN 0.91 ± 0.12. The kernel-50 CNN therefore has the highest accuracy in both conditions, with a margin of 0.02 to 0.03 over PINC-noBN, about one to two standard deviations. PINC-noBN uses 22% of its parameters and 43% of its multiply-accumulate operations (MACs), but needs far more activation memory (Section 4.5). Compared with a similar-size kernel-5 CNN, PINC-noBN varied less across seeds (SD 0.02 vs. 0.09 to 0.12), but with three seeds and two test episodes, the data do not support a claim that PINC-noBN is superior to either CNN.

Removing the second stage lowers test accuracy to 0.56 ± 0.01 (Condition A) and 0.59 ± 0.01 (Condition B), with anomaly recall of about 0.2 in both. This variant also has 81% fewer parameters (7586), so the result shows that the second stage supplies necessary capacity in this design, not that the specific bottleneck-inception layout is optimal. PINC

with batch normalization overfits (Section 4.2) and remains low when the transition zones are excluded (0.71 ± 0.05, ROC-AUC 0.950 against 0.998 for PINC-noBN), so its deficit is not only a decision-threshold effect. Using gyroscope ω_z alone, the kernel-50 CNN reaches 0.877 ± 0.012 and 0.989 ± 0.015, comparable to six channels. In contrast, PINC with all six branches fed ω_z (the configuration of the original code) did not learn a usable decision rule in these runs (ROC-AUC 0.49 ± 0.23 under Condition A, below 0.5 in two of three seeds). We replace the result originally reported in the first submission (Section 2.6).

In industrial operation, the cost of a false alarm depends on how often anomalies actually occur. On the test block, roughly 54% of windows are anomalous, which is far above plausible plant prevalence, so precision measured there overstates what operators would see. With the rates observed under Condition A (TPR 0.80, FPR 0.073), the precision of an alert would be about 0.01 at 0.1% prevalence, 0.10 at 1%, 0.37 at 5% and 0.55 at 10%; under Condition B (TPR 0.93, FPR rounded to 0.001) it would be about 0.48, 0.90, 0.98 and 0.99 (illustrative, from Bayes' rule).

To ground the classification metrics in operational reality, the confusion matrix of PINC-noBN (seed 1, chosen before the results were known) was evaluated on the test block ($N = 10{,}851$) under both conditions, as illustrated in Figure 6.

As shown in Figure 6, PINC-noBN (seed 1) classified 4661 of the 5041 normal windows correctly and flagged 380 of them (7.5%) as anomalous under Condition A. Outside the transition zones, 7 of 4399 normal windows (0.16%) were flagged (Condition B). The false-alarm rate under Condition A is too high for unsupervised deployment, and the node is best used as a first-stage filter whose alerts are confirmed downstream.

Of the 5810 anomalous windows, 4721 were detected (recall 0.81), and 1089 were missed under Condition A; outside the transition zones, 4501 of 4739 were detected (recall 0.95). The errors are concentrated near the class transitions. In seed 1, 281 of the 380 false positives and 611 of the 1089 false negatives lie within 300 windows (2.5 min) of a labeled transition, and the false positives lie almost entirely before the labeled onset of the second episode (Figure 5). Alert-level results with the trigger rule are given in Section 4.6.

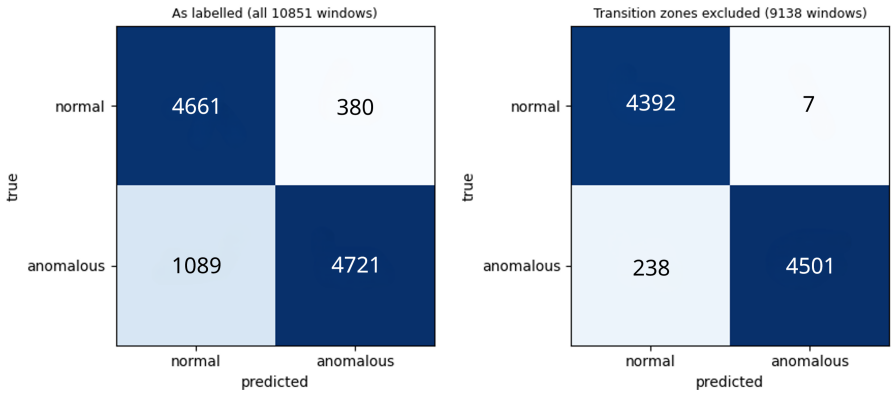


Figure 6. Confusion matrices of PINC-noBN (seed 1, chosen before the results were known). (**Left**): all $N = 10{,}851$ test windows with the original labels (Condition A). (**Right**): the 9138 windows outside the transition zones (Condition B). Cells give raw window counts.

4.5. Edge Footprint

Table 4 lists the computational footprint of the evaluated networks. We counted MACs only for convolutional and dense layers; we did not count batch normalization, pooling, or concatenation. The largest tensor is the largest single activation tensor in 32-bit floating point. The lower bound on peak activation memory is the maximum, over layers, of a layer's input and output sizes; the upper bound is the sum of all activation tensors. The actual peak depends on the inference runtime's memory planner and lies between the two.

Table 4. Footprint of the evaluated networks for one input window of 700 × 6 samples (1 KB = 1024 bytes; activations in 32-bit floating point; weights in 8-bit integers if quantized). Model identifiers are defined in Section 2.4.

Model	Parameters	MACs (M)	Weights, int8 (KB)	Largest Tensor	Peak Activations, Lower-Upper
PINC_noBN	37,154	25.9	36	1.05 MB	2.05–8.1 MB
PINC_noBN_1stage	7586	4.8	7	1.05 MB	2.05–6.5 MB
CASPER_K5	27,266	7.7	27	87 KB	0.17–0.35 MB
CASPER_K50	169,794	60.4	166	81 KB	0.15–0.29 MB
InceptionTime	492,930	342.0	481	350 KB	1.03–13.2 MB

PINC-noBN has 22% of the parameters and 43% of the MACs of the kernel-50 CNN, but 3.4 times the MACs of the kernel-5 CNN. Its Stage 1 output (700 × 384 values, 1.05 MB) makes its activation memory more than ten times larger than that of either CASPER-style CNN. Hence, the model is parameter-efficient but not memory-efficient. In 32-bit floating point, none of the networks fits the 0.06 MB sensor node used for data acquisition. This work does not include 8-bit quantization, a redesign that reduces the Stage 1 tensor, or measurements on target hardware.

4.6. Projected Bandwidth Under Event-Driven Reporting

To estimate the effect of event-driven reporting, we analytically calculated network consumption for a projected 1000 Hz scenario across several factory scales and anomaly prevalences. We made no network measurements, and we evaluated the classifier only at 20 Hz.

We derive the reporting state from window predictions using a fixed trigger rule defined before any evaluation and not optimized: an alert starts when at least 3 of the last 5 predictions are anomalous and ends after 11 consecutive normal predictions (5.5 s); windows are classified every 0.5 s. Alternative settings (2 of 3, 3 of 4, 5 of 6, and 8 of 10, each with release after 11) changed the fraction of time in the burst state by at most one percentage point. For PINC-noBN, the rule reduced the number of alert events in the test block from 100, 69 and 100 (raw predictions, seeds 1 to 3) to 2, 37 and 2. Counting events that contain no anomalous window under the original labels, seeds 1 and 3 produced none, and seed 2 produced four false-alarm events in 0.70 h of normal operation. With the transition zones excluded, no false-alarm event occurred in 0.61 h for any seed. Because only 0.6 to 0.7 h of normal operation is available and one event corresponds to about 1.4 to 1.6 events per hour, these counts do not support a false-alarm rate estimate; for zero observed events, the 95% upper bound is about 5 events per hour (rule of three, seeds treated as one sample). Relative to the signal change estimated in Section 4.3, the alert followed with a delay of 8.5 to 10.5 s at the onset of the anomalous episode, 8.5 s at the end of the first and −0.5 to 2.0 s at the end of the second; these estimates have a resolution of 5 s, and with a 35 s classification window delays of this order are expected. No latency was measured on hardware.

In a conventional architecture, a single robot,represented as 1 node in Table 5, streaming 6-axis kinematics at 1000 Hz via TCP/IPv4 generates 82.0 KB/s of continuous traffic. In the event-driven framework, the edge node sends a 1 Hz heartbeat of $B_{hb} = 0.06$ KB/s (assumed) during normal operation and switches to the full $B_{burst} = 82.0$ KB/s raw burst while it reports an anomaly. The fraction of time in the burst state is

$$q = p\, q_a + (1 - p)\, q_n, \tag{10}$$

where p is the anomaly prevalence, q_a is the fraction of anomalous operation spent in the burst state, and q_n is the corresponding fraction of normal operation. Both were measured on the alert state of PINC-noBN, including the hold time of the trigger rule, so false-positive bursts are included. The time-averaged traffic per robot is $B = (1 - q)B_{hb} + qB_{burst}$. Table 6 gives B for prevalences between 0.1% and 10% under the two evaluation conditions of Section 4.3: Condition A uses $q_a = 0.851$ and $q_n = 0.077$ (mean over three seeds, all test windows, original labels), Condition B uses $q_a = 0.993$ and $q_n = 0.000$ (windows outside the transition zones). The reduction relative to 82.0 KB/s is 84.5–92.2% under Condition A and 90.0–99.8% under Condition B. The earlier estimate of about 98.9% ignored false-positive bursts; it coincides with the optimistic value at 1% prevalence but is no longer the headline figure. These figures are analytical projections for a 1000 Hz deployment scenario that assume the 20 Hz classifier can be applied to 1000 Hz data; neither network behavior nor switch-buffer occupancy was measured.

Table 5. Projected bandwidth by factory scale (analytical, 1000 Hz scenario, not measured).

Scenario	Nodes *	Continuous Raw Bandwidth	Event-Driven, 1% Prevalence (A/B)	Event-Driven, 10% Prevalence (A/B)
Single Robot	1	82.0 KB/s	6.99/0.87 KB/s	12.70/8.20 KB/s
Assembly Line	50	4.10 MB/s	350/44 KB/s	635/410 KB/s
Factory Floor	200	16.40 MB/s	1.40/0.17 MB/s	2.54/1.64 MB/s

* Numbers of nodes chosen arbitrarily to represent scale. A: all test windows with original labels; B: transition zones excluded (optimistic); burst-state fractions as in the text. Heartbeat of 0.06 KB/s assumed.

Table 6. Projected per-robot bandwidth versus anomaly prevalence p (analytical, 1000 Hz scenario). Condition A: all test windows, original labels; Condition B: transition zones excluded (optimistic).

Prevalence p	q	Condition A Bandwidth (KB/s)	Reduction	q	Condition B Bandwidth (KB/s)	Reduction
0.1%	7.8%	6.42	92.2%	0.1%	0.14	99.8%
1%	8.5%	6.99	91.5%	1.0%	0.87	98.9%
5%	11.6%	9.53	88.4%	5.0%	4.13	95.0%
10%	15.4%	12.70	84.5%	9.9%	8.20	90.0%

5. Discussion

This study examined whether a compact edge classifier can support event-driven reporting that reduces network load. The classification results show that a model with 37,154 parameters can reach a test accuracy of 0.86 on a chronologically held-out block of the CASPER recording with the given labels, and 0.97 outside the zones where labels and signal are offset (Section 4.3). We derived the bandwidth benefit analytically but did not measure it, and we did not test the premise that communication, rather than computation, is the primary bottleneck.

5.1. Architecture and Design Choices

PINC-noBN reaches slightly lower accuracy than the CASPER-style CNN with kernel size 50 in both conditions (0.86 ± 0.02 against 0.88 ± 0.01 as labeled; 0.965 ± 0.020 against 0.992 ± 0.010 outside the transition zones), with 22% of its parameters and 43% of its MACs. It varied less across seeds than a similarly sized kernel-5 CNN. Its activation memory, however, is more than ten times larger than that of either CASPER-style CNN (Table 4). On this benchmark, the architecture therefore offers parameter efficiency and stable training,

not higher accuracy or lower memory use. The single-stage ablation (0.56 ± 0.01 as labeled) shows that Stage 2 provides capacity that Stage 1 alone lacks. Still, the ablation also removes 81% of the parameters, so it does not isolate the effect of the bottleneck-inception layout. We did not directly test whether processing the channels separately in Stage 1 helps; the kernel-50 CNN reaches 0.989 outside the transition zones with gyroscope ω_z alone, suggesting this benchmark can be solved from a single channel and does not require cross-channel fusion. Batch normalization caused strong overfitting (training accuracy 1.000, test accuracy 0.66 ± 0.04 as labeled and 0.71 ± 0.05 outside the transition zones). In contrast, the variant without it generalizes better, with a clear but moderate train-validation gap. Validation and test accuracy differ across models and seeds because the blocks contain different anomaly episodes and different label offsets. These observations are specific to this dataset and split, which contains about two test episodes; they motivate the design but do not establish it.

5.2. Operational Significance of False Alarms

With the given labels, PINC-noBN flags about 7% of normal windows; most of these occur in the minutes before a labeled transition, when the signal has already changed (Section 4.3). Outside the transition zones, the false-positive rate is about 0.1%, and with the trigger rule no false-alarm event occurred in 0.61 h of normal operation for any seed; this duration does not support a rate estimate (upper bound about 5 events per hour). The anomaly precision of 0.93, with the labels as given, reflects the test block's roughly 54% anomaly share. At plausible plant prevalences, alert precision would be much lower (about 0.10 at 1% and 0.55 at 10% with the labels as given, and 0.90 and 0.99 outside the transition zones, Section 4.4). The edge node is therefore best understood as a first-stage filter whose alerts need confirmation, for example, by central analysis of the raw burst, rather than a basis for automatic stops.

5.3. Limitations and Future Directions

Despite these results, several limitations define the boundaries of the current study and present avenues for future research.

First, the classification results rely on a single recording from one robot (CASPER), which contains eight speed-change episodes but no mechanical faults or collisions. The chronological test block (about 91 min) contains only about two anomaly episodes; we label them by aligning two non-hardware-synchronized recordings with a constant time offset and assigning each window to the last time-step. The labeled class transitions lag the change in the signal by several minutes, and the offset appears to grow over the recording (Section 4.3); the CASPER authors report that the sensor board does not run at exactly 20 Hz [21], and a constant offset cannot compensate for the resulting drift, which probably explains this; we have not verified it in our data, and the training block could not be checked. Absolute accuracy and false-positive rates near the transitions are therefore affected; Condition B is exploratory and was derived from the test signal. Correcting the alignment and retraining the models on corrected labels is left for future work. Calibration of the predicted probabilities was not assessed. Published CASPER results use a different protocol and are not directly comparable. We also used the test block during exploration and chose PINC-noBN as the final variant based on test results; we list all examined variants in Table 2 (see also Section 2.5). The test results of the selected variant are therefore optimistic. With one robot and about two test anomaly episodes, the results do not show how the models behave under other operating conditions, on other robots, or for other fault types; anomalies such as gradual wear may differ greatly from speed changes in the signal. The results should therefore be read as a feasibility study on one benchmark. We seed the runs, but they are not bit-reproducible on GPU. The anomaly share is 43%, 60%, and 54%

in the training, validation, and test blocks, far above the below-1% prevalence plausible in a live deployment, so the reported precision values overstate what operators would see ("base-rate fallacy"). Organic, progressive mechanical degradation, such as gradual bearing race deterioration or gearbox backlash developing over several months, introduces highly nonlinear, creeping noise profiles that this benchmark does not contain. The binary framework ($K = 2$) must be expanded into a multi-class, continuous degradation model to capture these long-term phenomena, and future iterations must be evaluated on heavily skewed datasets (e.g., 99:1 ratios) to validate real-world false-positive tolerance.

Second, executing continuous inference loops on 700-time-step windows at the extreme edge introduces inherent thermal and energy constraints. For battery-operated or thermally encapsulated sensor nodes (such as the Arduino Nicla Sense ME), running the forward pass continuously at high frequencies will rapidly deplete the battery. Future iterations must investigate intelligent inference duty cycling—for example, using a cascaded threshold where a micro-power wake-up filter triggers full PINC inference only when baseline vibration exceeds a threshold.

Building on this tiered approach, identifying the root cause of a mechanical failure requires higher diagnostic resolution. A critical implication of the proposed E-DTNCS architecture is that it could support a tiered inference pipeline without adding continuous load to the industrial network, but we did not test this.

During nominal operation, bandwidth utilization is limited to the heartbeat; false-positive bursts add traffic (Section 4.6). When the edge node classifies a kinematic window as anomalous, it can temporarily suspend semantic suppression and dynamically open a high-throughput channel to stream raw 1000 Hz IMU telemetry to the centralized digital twin, supported by state-of-the-art industrial network solutions [28]. This tiered mode is a design proposal and was not tested on a network.

This transient, event-triggered transmission allows the cloud infrastructure to leverage unconstrained High-Performance Computing (HPC) environments. The centralized digital twin can ingest the high-fidelity burst and process it through much heavier, multi-class diagnostic architectures - such as deep Physics-Informed Neural Networks (PINNs) or large-scale sequence models- that far exceed the microcontroller's 0.5 MB Flash and 0.06 MB SRAM limits. While the parameters of PINC-noBN occupy about 0.15 MB in float32 (37,154 parameters), the baseline 32-bit Keras implementation evaluated in this study produces an intermediate concatenated tensor ($\mathbf{H}_{stage1}$) with a depth of 384. Because Stage 1 preserves the temporal resolution of the 700-time-step window, this float32 activation map alone requires approximately 1.05 MB of working memory; the peak activation memory lies between 2.05 and 8.1 MB (Table 4). The model therefore requires an edge device with substantially more memory than the Nicla and exceeds the 0.06 MB internal SRAM limit of ultra-low-power sensor nodes (such as the Nicla Sense ME used for data acquisition). Therefore, to run inference directly on the sensor chip, future work must integrate INT8 post-training quantization and stronger early temporal down-sampling to keep the Tensor Arena within 0.06 MB. By combining ultra-low-power edge filtering with on-demand cloud analytics, the E-DTNCS is expected to reduce network load for the broader multi-agent ecosystem while keeping detailed diagnostics available during disturbances; this expectation still needs experimental validation.

Third, while the protocol overhead reduction is expected to lower network saturation, physical Hardware-in-the-Loop (HiL) response times remain to be validated. The latency of a physical UR3e reacting autonomously to a digital twin's command over a live factory network, subject to unpredictable electromagnetic interference (EMI) and variable routing paths, requires comprehensive empirical testing.

Finally, this study evaluated PINC-noBN (0.86 ± 0.02) against CASPER-style CNNs and an InceptionTime-type network on the 20 Hz CASPER benchmark. Temporal convolutional networks, residual CNNs, and depthwise-separable CNNs were not evaluated. Evaluating the architecture on higher-frequency (1000 Hz) multi-robot datasets and dynamic vibration regimes, and measuring inference latency on embedded hardware, remain important directions for future research.

6. Conclusions

Continuously synchronizing digital twins with high-frequency physical dynamics create a fundamental network bottleneck in modern smart manufacturing. This study proposes the E-DTNCS architecture, in which an edge node classifies IMU windows from a Universal Robots UR3e and reports events rather than streaming raw telemetry, and evaluates a compact two-stage Parallel Inception 1D-CNN (PINC-noBN) as the classifier.

Mathematical evaluation of industrial IPv4 routing shows that streaming 1000 Hz raw IMU telemetry incurs a protocol overhead exceeding 70% in the projected scenario. The proposed edge node is designed to decouple this data generation from the transmission bus. The classifier processes 700-time-step windows ($N = 700$) in 32-bit floating-point precision, handling each axis separately before cross-channel fusion.

On the CASPER benchmark (20 Hz, speed-change anomalies, chronological split, about two anomaly episodes in the test block), PINC-noBN reached a test accuracy of 0.86 ± 0.02 over three seeds with 37,154 parameters when evaluated with the labels as given, slightly below a CASPER-style CNN with kernel size 50 (0.88 ± 0.01, 169,794 parameters) and above an InceptionTime-type network (0.63 ± 0.06). The labeled class transitions lag the signal change by several minutes; outside the transition zones, PINC-noBN reached 0.97 and the kernel-50 CNN 0.99. PINC-noBN is parameter-efficient and trains stably across seeds, but requires more than ten times the activation memory of CASPER-style CNNs (at least 2 MB) and does not fit on the 0.06 MB sensor node used for acquisition. An analytical projection that includes false-positive bursts indicates bandwidth reductions of roughly 85–92% with the labels as given and 90–99.8% outside the transition zones, for anomaly prevalences between 10% and 0.1% in a 1000 Hz scenario. This is a projection, not a network measurement; Hardware-in-the-Loop response times, measured latency, higher-frequency data, the effect on switch buffers, and a corrected label alignment remain to be validated.

Author Contributions: Conceptualization, D.B. and K.J.; methodology, D.B.; software, D.B.; validation, D.B.; formal analysis, D.B.; investigation, D.B.; data curation, D.B.; writing—original draft preparation, D.B.; writing—review and editing, D.B. and K.J.; visualization, D.B.; supervision, K.J.; project administration, K.J.; funding acquisition, D.B. and K.J. All authors have read and agreed to the published version of the manuscript.

Funding: Project no. C2307874 has been implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the KDP-2023 funding scheme.

Data Availability Statement: The CASPER dataset used for the empirical validation in this study is publicly available. The dataset can be accessed at: https://www.kaggle.com/datasets/hkayan/industrial-robotic-arm-imu-data-casper-1-and-2 (accessed on 5 May 2025).

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

1D-CNN	One-Dimensional Convolutional Neural Network
BN	Batch Normalization
CASPER	Context-Aware IoT Anomaly Detection System for Industrial Robotic Arms
CNN	Convolutional Neural Network
CPS	Cyber–Physical Systems
DoF	Degree-of-Freedom
DT	Digital Twin
E-DTNCS	Edge-Enabled Digital Twin Networked Control System
EMI	Electromagnetic Interference
FPR	False-Positive Rate
FPU	Floating-Point Unit
GAP	Global Average Pooling
GPU	Graphics Processing Unit
HiL	Hardware-in-the-Loop
HPC	High-Performance Computing
IMU	Inertial Measurement Unit
INT8	8-bit Integer
IoT	Internet of Things
IPv4	Internet Protocol version 4
MACs	Multiply-Accumulate Operations
OPC UA	Open Platform Communications Unified Architecture
OT	Operational Technology
PINC	Parallel Inception 1D-Convolutional Neural Network
PINN	Physics-Informed Neural Network
RAM	Random-Access Memory
ReLU	Rectified Linear Unit
ROC-AUC	Area Under the Receiver Operating Characteristic Curve
RTDE	Real-Time Data Exchange
SD	Standard Deviation
SIMs	Subspace Identification Methods
SRAM	Static Random-Access Memory
TCP	Transmission Control Protocol

References

1. Sisinni, E.; Saifullah, A.; Han, S.; Jafari, U.; Passerini, M. Industrial Internet of Things: Challenges, Opportunities, and Directions. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4724–4734. [CrossRef]
2. Tao, F.; Cheng, J.; Qi, Q.; Zhang, M.; Zhang, H.; Sui, F. Digital twin-driven product design, manufacturing and service with big data. *Int. J. Adv. Manuf. Technol.* **2018**, *94*, 3563–3576. [CrossRef]
3. Negri, E.; Fumagalli, L.; Macchi, M. A review of the roles of Digital Twin in manufacturing. *Procedia CIRP* **2017**, *60*, 560–565.
4. Gopal, L.; Singh, H.; Mounica, P.; Mohankumar, N.; Challa, N.P.; Jayaraman, P. Digital twin and IOT technology for secure manufacturing systems. *Meas. Sens.* **2023**, *25*, 100661. [CrossRef]
5. Abudoureheman, T.; Otsubo, H.; Wang, F.; Shimasaki, K.; Ishii, I. High-Frame-Rate Camera-Based Vibration Analysis for Health Monitoring of Industrial Robots Across Multiple Postures. *Appl. Sci.* **2025**, *15*, 12771. [CrossRef]
6. Minerva, R.; Biru, A.; Rotondi, D. Towards a definition of the Internet of Things. *IEEE Internet Things Initiat.* **2015**, *1*, 1–74.
7. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376. [CrossRef]
8. Lu, Y.; Liu, C.; Wang, K.I.-K.; Huang, H.; Xu, X. Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues. *Robot. Comput. Integr. Manuf.* **2020**, *61*, 101837. [CrossRef]
9. Friederich, J.; Francis, D.P.; Lazarova-Molnar, S.; Mohamed, N. A framework for data-driven digital twins of smart manufacturing systems. *Comput. Ind.* **2022**, *136*, 103586. [CrossRef]
10. He, R.; Chen, G.; Dong, C.; Sun, S.; Shen, X. Data-driven digital twin technology for optimized control in process systems. *ISA Trans.* **2019**, *95*, 221–234. [CrossRef]
11. Preuss, K.; Schulte, S.N.; Rzazonka, L.; Befort, L.; Fresemann, C.; Stark, R.; Russwinkel, N. Towards A Human-Centered Digital Twin. *Procedia CIRP* **2023**, *118*, 324–329. [CrossRef]

12. Pratt, M.J. Introduction to ISO 10303—The STEP Standard for Product Data Exchange. *J. Comput. Inf. Sci. Eng.* **2001**, *1*, 102–103. [CrossRef]
13. Kitsios, F.; Chatzidimitriou, E.; Kamariotou, M. The ISO/IEC 27001 Information Security Management Standard: How to Extract Value from Data in the IT Sector. *Sustainability* **2023**, *15*, 5828. [CrossRef]
14. Shaaban, A.M.; Chlup, S.; El-Araby, N.; Schmittner, C. Towards Optimized Security Attributes for IoT Devices in Smart Agriculture Based on the IEC 62443 Security Standard. *Appl. Sci.* **2022**, *12*, 5653. [CrossRef]
15. Capuano, G.; Rimoli, J.J. Smart finite elements: A novel machine learning application. *Comput. Methods Appl. Mech. Eng.* **2019**, *345*, 363–381. [CrossRef]
16. Udrescu, S.-M.; Tegmark, M. AI Feynman: A physics-inspired method for symbolic regression. *Sci. Adv.* **2020**, *6*, eaay2631. [CrossRef] [PubMed]
17. Lawal, Z.K.; Yassin, H.; Lai, D.T.C.; Che Idris, A. Physics-Informed Neural Network (PINN) Evolution and Beyond: A Systematic Literature Review and Bibliometric Analysis. *Big Data Cogn. Comput.* **2022**, *6*, 140. [CrossRef]
18. Ye, C.; Wang, J.; Peng, C.; Ju, Z.; Geng, X.; Zhang, L.; Sui, Q.; Jiang, M.; Zhang, F. Novel cross-domain fault diagnosis method based on model-agnostic meta-learning embedded in adaptive threshold network. *Measurement* **2023**, *222*, 113677. [CrossRef]
19. Tao, F.; Zhang, M.; Liu, Y.; Nee, A.Y.C. Digital Twin in Industry: State-of-the-Art. *IEEE Trans. Ind. Inform.* **2019**, *15*, 2405–2415. [CrossRef]
20. Leclerc, S.; Hansen, E.; Bucaioni, A.; Ashjaei, M. *Reducing IoT Data at the Edge: A Comparative Evaluation*; Mälardalen University: Västerås, Sweden, 2023.
21. Kayan, H.; Heartfield, R.; Rana, O.; Burnap, P.; Perera, C. CASPER: Context-Aware IoT Anomaly Detection System for Industrial Robotic Arms. *ACM Trans. Internet Things* **2024**, *5*, 18. [CrossRef]
22. Niu, Q.; Sui, Z.; Han, J.; Zhao, Y. An Industrial Robot Gearbox Fault Diagnosis Approach Using Multiscale Empirical Mode Decomposition and a One-Dimensional Convolutional Neural Network-Bidirectional Gated Recurrent Unit Method. *Processes* **2025**, *13*, 1722. [CrossRef]
23. Warden, P.; Situnayake, D. *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*, 1st ed.; O'Reilly Media: Sebastopol, CA, USA, 2019; pp. 15–45.
24. Abdallah, M.; Joung, B.G.; Lee, W.J.; Mousoulis, C.; Raghunathan, N.; Sutherland, J.W.; Bagchi, S. Anomaly Detection and Inter-Sensor Transfer Learning on Smart Manufacturing Datasets. *Sensors* **2023**, *23*, 486. [CrossRef]
25. da Silva Mendonça, R.; de Oliveira Lins, S.; Valente de Bessa, I.; de Carvalho Ayres, F.A., Jr.; Paiva de Medeiros, R.L.; Ferreira de Lucena, V., Jr. Digital Twin Applications: A Survey of Recent Advances and Challenges. *Processes* **2022**, *10*, 744. [CrossRef]
26. Petrakis, K.; Agorogiannis, E.; Antonopoulos, G.; Anagnostopoulos, T.; Grigoropoulos, N.; Veroni, E.; Berne, A.; Azaiez, S.; Benomar, Z.; Kakoulidis, H.; et al. Enhancing DevOps Practices in the IoT–Edge–Cloud Continuum: Architecture, Integration, and Software Orchestration. *Software* **2024**, *4*, 10. [CrossRef]
27. Ieva, S.; Loconte, D.; Pazienza, A.; Colombo, M.; Marzo, F.; Loseto, G.; Scioscia, F.; Ruta, M. A Composable Architectural Model for Digital Twin Computing Applications. *Appl. Sci.* **2026**, *16*, 4541. [CrossRef]
28. Cheng, J.; Chen, W.; Tao, F.; Lin, C.L. Industrial IoT in 5G environment towards smart manufacturing. *J. Ind. Inf. Integr.* **2018**, *10*, 10–19. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.