

# Real-time Anomaly Detection for Industrial Robotic Arms using Edge Computing

Hakan Kayan, Ryan Heartfield, Omer Rana, Pete Burnap, and Charith Perera

**Abstract**—The integration of Internet of Things (IoT) devices in industrial applications has become viable due to advancements in ubiquitous computing that enable complex machine learning (ML) tasks on resource-constrained devices. Unlike prior approaches that rely on built-in sensors, our system utilizes externally gathered Inertial Measurement Units (IMU) data for anomaly detection. In this paper, we show that simple 1D-CNN and LSTM models on an ultra-low-power device (Nucleo Sense ME) optimized for edge-based industrial anomaly detection can achieve approximately 98% accuracy and F1 score in detecting movement-based anomalies (e.g., collisions and joint velocity deviations) in industrial robotic arms. We analyzed an advanced manufacturing scenario where the robotic arm performs three consecutive, distinct tasks (pick-and-place, painting, and screw-driving) and demonstrated that the proposed anomaly detection system is task-independent. We implemented these models on-device by designing a minimal model architecture and modifying source code to minimize RAM usage and Bluetooth Low Energy (BLE) overhead. Additionally, we examined the challenges of deploying ML models in resource-constrained environments by analyzing various quantization methods and the impact of hyperparameter choices on inference time, accuracy, and memory consumption. Our approach focuses on detecting anomalies directly at the data source which enables true real-time detection with a complete edge computing framework that achieves a 10Hz data frequency and a 250ms inference time when BLE is active. Furthermore, we generated a comprehensive dataset capturing quaternion and IMU data from an industrial robotic arm over 26 hours, including various anomaly scenarios, and made the source code available on GitHub for replicability.

**Index Terms**—anomaly detection, industrial robotic arms, edge computing, cyber-physical systems, real-time monitoring

## I. INTRODUCTION

UBIQUITOUS computing enhances our lives by integrating seamless technologies into our daily activities while remaining mostly invisible. A heterogeneous set of devices is utilized to create context-aware applications that provide a more efficient, convenient, and personalized experience for end-users. Thanks to their high mobility, interconnectivity, and adaptability, ubiquitous applications are present in various domains, including smart homes and cities, wearables, entertainment, and manufacturing systems [1]. In industrial domains, ubiquitous computing is primarily researched under the paradigms of the industrial internet of things (IIoT), industrial wireless sensor networks (IWSN), and industrial cyber-physical systems (ICPS),

With the arrival of Industry 4.0 [2], industrial systems have become increasingly interconnected. Early systems were air-gapped due to security concerns, but current systems have adopted internet integration, leading to the widespread adoption of IIoT devices. This transformation has optimized resource use through intelligent automation, high-resolution production, smart human-machine interaction, and predictive maintenance [3]. However, this increased connectivity has expanded the attack surface, making systems more vulnerable to adversarial attacks. Recent cyber-physical incidents demonstrate that these attacks often target data integrity to cause significant physical disruptions [4]. Therefore, anomaly detection methods that consider physical parameters are desired. Current information security systems, focused on protecting enterprise IT systems, often fail when operational technologies (OT) are involved [5].

Physical anomalies in manufacturing and other industrial systems, such as electrical grids, water, and gas plants, can lead to disasters in critical infrastructures (CIs) with time-sensitive applications. Therefore, real-time anomaly detection, which minimizes the time between an anomalous physical event and its detection, is important but often overlooked in existing physics-based cyber-physical detection methods [6]. These methods generally fall into two categories based on their detection methodology: statistical and data-driven [7]. They can also be classified by their localization approach, distinguishing them as either centralized or decentralized. Statistical solutions face scalability challenges as they rely on predefined assumptions about data distribution, making them less flexible and harder to scale with increasing data complexity and volume while centralized ones suffer from detection delays due to the time required to transmit data to a central server and back. Consequently, there is a growing interest in decentralized data-driven anomaly detection approaches [8], which offer improved scalability and lower detection latency.

Edge computing improves the effectiveness of decentralized data-driven approaches by processing data closest to the source of where it is generated. Naturally, this reduces latency, improves real-time response, and mitigates the burden on centralized systems [9]. By utilizing ultra-low-power edge devices directly at the data source, which are often power and resource-constrained, we can implement efficient ML models that perform anomaly detection in near real-time. Direct integration at the edge reduces communication latency, enhances privacy and security, and provides a practical solution for monitoring industrial systems [10]. Embedded ML frameworks such as TensorFlow Lite Micro (TFLite Micro) [11] facilitate the deployment of neural networks, including

H. Kayan, O. Rana, P. Burnap, and C. Perera are with Cardiff University, Senghennydd Rd, Cardiff, United Kingdom, CF24 4AX.

R. Heartfield is with the Kingston University, Penrhyn Road, Kingston upon Thames, Surrey KT1 2EE, United Kingdom.

(Corresponding author: Hakan Kayan.)

1D and 2D convolutional neural networks (CNNs) and Long Short-Term Memory (LSTM) networks, on these edge devices, though they have constraints on the available layers due to the low-computing power of edge devices.

In our previous work [12], we developed a system that used externally gathered IMU data from an industrial robotic arm to detect movement-based anomalies. In this paper, we extend that work by presenting an IoT-based anomaly detection system that identifies contextual anomalies in time series data within a manufacturing testbed. Unlike traditional methods that rely on built-in sensors, our system uses external IMU data to detect anomalies. We evaluate the system in an advanced manufacturing scenario where the robotic arm consecutively performs three distinct tasks, screwdriving, painting, and pick-and-place, to demonstrate task-independent anomaly detection. Furthermore, we deploy our ML models as TinyML solutions on a resource-constrained, ultra-low-power edge device, and we analyze different quantization methods and hyperparameter settings to assess their impact on model accuracy, inference time, and memory usage. Finally, this work provides a fully automated pipeline using open-source tools for ML-based anomaly detection on industrial robotic arms. This enables non-intrusive, real-time monitoring. We also address the challenges of running ML algorithms directly on ultra-low-power devices and present a real-world dataset with corresponding experimental results.

- We introduce a novel end-to-end edge anomaly detection framework that can operate independently of cloud infrastructure which enables real-time detection with minimal latency where cloud dependency is only required to update the model running on the edge. This system is optimized for deployment on an ultra-low-power industrial edge device (Nicla Sense ME) and contributes insights into practical trade-offs in accuracy, energy efficiency, and inference speed. The source code is available on GitHub<sup>1</sup>.
- We generate a comprehensive dataset capturing quaternion and IMU data from an industrial robotic arm over 26 hours, including anomalies such as: (I) hitting the arm, (II) hitting the platform, (III) attaching an extra weight, (IV) generating a magnetic field, and (V) earthquake simulation.
- We conduct an extensive performance analysis of ML models (LSTM and 1D-CNN) on edge and cloud platforms, utilizing techniques like root mean square error (RMSE) for loss metrics and various quantization methods to assess model size, accuracy, and latency, hence focusing on the feasibility of deploying these models in edge environments.
- We analyze the operational boundaries and performance capabilities of an industry leading edge development board, evaluating the performance trade-offs of neural network model architectures and hyperparameters for state-of-the-art algorithms (1D-CNN, LSTM) on key edge device resource metrics such as RAM, flash memory and

energy consumption to optimize model efficiency and performance that also balances practical deployment in resource-constrained environments.

The proposed system operates under specific conditions that support reliable anomaly detection in an industrial setting. To keep the evaluation focused on the effectiveness of the method rather than external disruptions, we define key operational constraints. These constraints cover hardware stability, environmental factors, task consistency, and reliable communication between system components. Based on these points, we make the following assumptions:

- **System and Environmental Stability:** The integrity of the edge board and the entire system is protected, with a stable power supply and controlled environmental conditions which does not affect the integrity of the generated IMU data.
- **Predictable Task Execution and Data Quality:** The robotic arm performs repetitive tasks with inherent patterns, and the externally gathered sensor data is assumed to be calibrated and reliable.
- **Robust and Secure Communication:** BLE connectivity is maintained without drops due to close proximity to the Fog Node (PiHMI) and the data sampling rate being fixed to 10Hz, and data transmission is secure and reliable. The cloud node is always active with stable connection as it is required for the initial development and updating the model when necessary.

## II. RELATED WORK

### A. Industrial Robotic Arms

In recent years, a significant amount of research has focused on anomaly detection in industrial robotic arms using various modeling and machine learning approaches. One study [13] employed autoencoders (AE) to analyze sound data from internal sensors, effectively identifying abnormal statuses despite limitations in sound frequency response and reliance on normal status data for training. Another approach used Support Vector Machines (SVMs) to detect trajectory deviations in repetitive robotic tasks, demonstrating high accuracy but limited to predefined trajectories and requiring precise boundary settings [14]. Additionally, backscatter signals have been utilized to ensure movement accuracy by monitoring signal propagation signatures, though this method requires precise placement and calibration of tags and is susceptible to environmental interference [15]. One-class SVMs have also been used for cognitive analytics-based machine health monitoring and predictive maintenance, providing real-time insights but struggling with complex anomaly detection due to the lack of labeled data [16]. Collision detection has been achieved through analyzing joint torque data with feed-forward neural networks, showing high accuracy but limited to collision-specific anomalies and reliant on sensor data accuracy [17]. Gradient boosting techniques have been applied for failure detection, requiring extensive training data and being computationally intensive [18]. Our approach also utilizes ML but differs by focusing on externally gathered IMU data which enables real-time on-device anomaly detection using

<sup>1</sup><https://github.com/hkayann/Real-time-Anomaly-Detection-in-Industrial-Robotic-Arms-via-TinyML>

neural networks such as 1D-CNN and LSTM, while improving adaptability across different operational conditions.

At the time of writing, the aforementioned work of [14] may be considered the closest to our research. The authors explore a scenario where a Yaskawa Motoman MH5 industrial robotic arm performs a repetitive task, following a predetermined trajectory. They introduce slight deviations to the path to generate anomalies and utilize SVMs and One-class SVMs for detection. Similarly, our work focuses on detecting movement-based anomalies where the arm is subject to a range of different physical disturbances. However, unlike prior studies that focus on a single-task robotic operation with predefined motion patterns, our work extends to task-independent anomaly detection, evaluating the system across multiple distinct manufacturing tasks. Furthermore, we provide a fully integrated edge-based ML pipeline designed for real-time anomaly detection, focusing on deployment feasibility on ultra-low-power embedded devices. To the best of our knowledge, we are one of the first to thoroughly investigate the feasibility of a real-time edge anomaly detection system utilizing externally gathered IMU data for predictive maintenance in industrial robotic arms, supporting early identification of contextual anomalies across diverse operational tasks. The main motivation behind these kind of works where external sensors are utilized is the questionable integrity of the built-in data as we previously discussed in the introduction.

### B. Industrial Anomaly Detection on Edge

Industrial settings require continuous, real-time detection, requiring decentralized, low-latency, low-power systems due to their 24/7 operation and the growing heterogeneity of environments. While raw inference speed is important, the choice of edge computing over cloud-based inference should also consider factors like network reliability, which can be inconsistent in industrial environments, leading to potential interruptions in cloud-based processing [19]. Edge computing also enhances privacy and security by processing sensitive data locally, reducing exposure to potential breaches during transmission [20]. Additionally, the ability to operate efficiently in low-power conditions is vital for systems that must function autonomously for extended periods, making edge solutions not only practical but essential in many industrial applications [21]. While cloud offloading is well-studied [22], applying it to industrial anomaly detection with ultra-low-power edge/IoT devices is relatively new. Recent advancements now enable ML algorithms, such as neural networks, to run efficiently on ultra-low-power edge/IoT devices. However, existing studies on edge-based industrial anomaly detection often rely on single-task operations [23], [14] or cloud-dependent anomaly detection [24], [25], limiting real-time feasibility. Our work addresses this gap by implementing a fully edge-deployed TinyML system which enables real-time anomaly detection directly on an ultra-low-power device across multiple robotic tasks where cloud dependency is only required if the model to be updated over-the-air (OTA). Some literature defines edge anomaly detection as using nodes that gather data from multiple sensors [26], often with Single Board Computers (SBCs) like Raspberry Pi as edge nodes [27].

TABLE I: Example Embedded ML Applications

Reference	Model	Open-source	Application
[41]	kNN	✓	Cane Gesture Recognition
[37]	NN	✓	Speech Enhancement
[35]	NN	X	Vital Sign Monitoring
[34]	TEDA	X	Road Hole Detection
[33]	NN	X	USB Fan Tilting Detection
[38]	NN	X	Hand Gesture Recognition
[39]	RF	X	Vehicle Identification
[36]	NN	X	Face Mask Detection
[40]	NN	X	Drone Navigation
[32]	NN	✓	Anomalous Sound Detection
[42]	NN	X	Object Detection
[43]	TEDA	X	Vehicle Emission Monitoring
[31]	IF	X	Submersible pump
This Work	NN	✓	Anomalous Movement Detection

**TEDA**: Typicality and Eccentricity Data Analytics, **RF**: Random Forest, **IF**: Isolation Forest, **NN**: Neural Networks, **kNN**: k-Nearest Neighbors.

Embedded ML applications vary widely due to the availability of numerous sensor types and actuators. Their attributes of low latency, enhanced security, and privacy, coupled with minimal bandwidth requirements, make them ideal for scalable and real-time applications [28]. Edge development boards with Arm Cortex processors are particularly favored in industrial settings [29] as they also contain various types of built-in sensors. These devices support a range of applications, leading to the development of Embedded ML-as-a-service [30]. Examples of Embedded ML applications include anomaly detection in submersible pumps [31], recognition of anomalous sounds [10], [32], detection of fan tilting [33], identification of road anomalies [34], monitoring of vital signs for Covid-19 treatment [35], verification of face mask use [36], speech enhancement for hearing aid users [37], hand gesture recognition [38], vehicle identification [39], drone navigation assistance [40], cane gesture recognition [41], fruit detection [42], and vehicle emission monitoring [43]. Despite the widespread adoption of TinyML in various fields, its application in industrial robotic arms for real-time movement-based anomaly detection remains largely unexplored. Our work contributes by deploying optimized 1D-CNN and LSTM models directly on an ultra-low-power edge device demonstrating the feasibility of task-independent anomaly detection in an industrial setting. Neural networks are popular in Embedded ML applications due to their adaptability, flexibility, and efficient optimization techniques. Table I compares recent Embedded ML applications. TFLite Micro, developed by Google and now open-source, is the most common framework used in these applications [44] which is also utilized in this work.

### C. Neural Network Models for Edge Anomaly Detection

Neural network-based models have proven highly effective for time-series anomaly detection [12]. For regression-based anomaly detection, 1D-CNNs and LSTMs are preferred [45], while 2D-CNNs have proven superior when classifying image data due to their ability to extract spatial features [46]. 1D-CNNs effectively recognize temporal patterns in sequence data, such as time series or speech [47]. LSTMs, though

computationally more complex, consistently outperform other models in these contexts [48]. This complexity makes 1D-CNNs more favored for Embedded ML applications due to their lower computational requirements. Recent research on 1D-CNNs includes intelligent fault diagnosis of bearings [49], human activity recognition [50], emotion recognition via speech analysis [51], and anomaly detection in industrial water treatment systems [52].

Contrasting with CNNs, Recurrent Neural Networks (RNNs) are distinguished by their inherent memory, enabling them to process sequential data with a temporal context. Among RNN architectures, LSTM units and Gated Recurrent Units (GRU) are more popular. LSTMs, along with their variants, have been shown to consistently outperform other models in tasks involving sequential data [53], [48], thanks to their ability to capture long-term dependencies. However, the computational complexity of LSTM-based models is generally higher than that of CNNs, including 1D-CNNs [54], which may affect their suitability for certain real-time or resource-constrained applications. In this work, we design lightweight 1D-CNN and LSTM models, tuning hyperparameters to minimize model size for deployment on an ultra-low-power edge device (Nicla Sense ME) with only 64KB RAM, much of which is used for BLE connectivity. We apply these models to detect movement-based anomalies across multiple robotic tasks and evaluate their performance in an edge computing environment.

#### D. Edge-to-Cloud ML Systems for Industrial Applications

ML pipelines automate the process from data collection to inference, essential for maintaining model performance over time. Data is typically collected at the edge, with model training and deployment occurring on centralized or cloud platforms due to high computational requirements. However, popular ML pipelines like TensorFlow Extended [55] and Kubeflow [56] often fall short for Embedded ML models. Existing commercial solutions, such as Edge Impulse [57] and NanoEdge AI Studio [58], are limited by reduced customization and specific hardware dependencies. In contrast, our work introduces a novel, fully open-source edge-to-cloud ML pipeline that bridges these gaps. Training is conducted on a cloud-based workstation, and inference is performed on ultra-low-power edge devices which enables real-time anomaly detection in industrial settings while emphasizing low latency, power efficiency, and adaptability in resource-constrained environments, supporting continuous model updates and real-time monitoring.

### III. SYSTEM OVERVIEW

#### A. The Use Case & Industrial Robotic Arm

TABLE II: Joint Parameters

Task	Joint Velocity ( $^{\circ}/s$ )	Joint Acceleration ( $^{\circ}/s^2$ )
Screwdriving	80	100
Painting	120	100
Pick-and-place	80	80

We implement a scenario using a cobot, UR3e by Universal Robots<sup>2</sup>. This cobot emulates three distinct tasks performed non-anomalously for 24 hours consecutively: screwdriving, painting, and pick-and-place. In total, there are 23 poses; 21 of these poses are unique, with the arm repeating one pose twice during the pick-and-place task. The arm initiates from a home position, returning there after completing each task to remain idle for 5 seconds. It is attached with a 2FG7 OnRobot Parallel Gripper<sup>3</sup> and mounted on a custom-made steel platform. Figure 2 showcases a sample pose from each task along with an image of the teach pendant, the built-in human-machine interface (HMI), used to configure the movements. UR3e provides three movement types: *moveJ*, *moveL*, and *moveP*. During the experiment, *moveJ* is employed, determining the movement per pose, joint velocity, and joint acceleration as shown in Table II. This ensures all joints start and finish their motion at the same time. The *moveL* and *moveP* movement types, which focus on straight path movements and circular blends, are not employed in our use case scenario. These movement types are more suited to traditional applications and do not fully capture the complex multitasking capabilities required in smart factory scenarios, where robotic arms need to perform multiple tasks with varied movements. Figure 1 demonstrates the movement types.

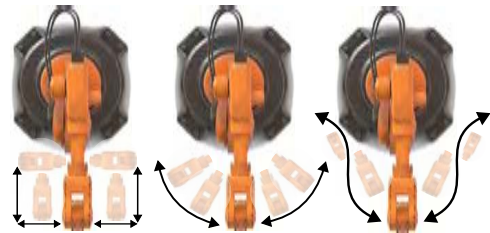


Fig. 1: The movement types (from left to right, *moveL*, *moveP*, and *moveJ*) offered by the UR3e software. Any movement that does not strictly follow a linear or circular path requires the use of the *moveJ* command. In this use case scenario, the tasks are designed according to advanced scenarios, thus we implement the *moveJ* option.

#### B. Quaternions and IMU Data

Quaternions represent rotations in three-dimensional space using a four-dimensional vector, offering a robust solution for orientation representation compared to Euler angles, which can suffer from gimbal lock [59]. IMU data can include up to nine features, such as accelerometer, gyroscope, and magnetometer axes, while quaternions use just four features ( $q = w + xi + yj + zk$ ), enhancing computational efficiency. This is particularly beneficial in edge-based neural networks, where reduced dimensions lower computational demands and improve real-time performance. Quaternions effectively capture anomalies related to orientational deviations, while IMU data detects linear motion anomalies through the accelerometer, angular velocity through the gyroscope, and magnetic field disruptions with the magnetometer. Our previous work [60]

<sup>2</sup><https://www.universal-robots.com/products/ur3-robot/>

<sup>3</sup><https://onrobot.com/en/products/2fg7>

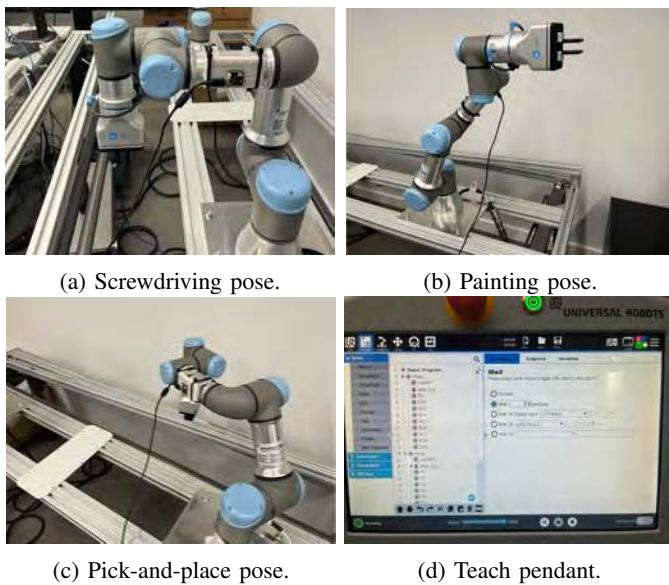


Fig. 2: There are 23 example poses associated with tasks such as screwdriving (10 poses), painting (7 poses), and pick-and-place (6 poses). Each pose is manually configured using the teach pendant, a standard built-in HMI accompanying Universal Robot industrial robotic arms.

demonstrated that the Nicla Sense ME <sup>(4)</sup> provides the most consistent quaternion and IMU data among tested devices.

### C. Edge Anomaly Detection Framework

To achieve true real-time anomaly detection, detecting anomalies directly at the data source is required. There are two key parameters to consider: (I) The delay in detection increases with the distance from the data source due to communication latency, and (II) the inference time depends on the available computing power. Low computing power results in higher inference time. While inference time in the cloud can be measured in nanoseconds, on the edge (data source), it can be seconds. This is a critical factor in determining the highest available frequency for data generation, as inference time slower than data generation leads to incremental delay, making real-time detection impossible.

We developed an anomaly detection framework that integrates edge, fog, and cloud nodes to ensure real-time anomaly detection. The cloud node, a secured data science workstation equipped with an NVIDIA RTX A6000 GPU, offers high computational power with a power consumption of 300W and a cost of approximately £8000. This node handles model training, generation, and updates. Once trained, models are transmitted to the fog node, a cost-effective Raspberry Pi 4B (6.4W, £80), located on-site to maintain reliable BLE communication with the edge node. The fog node supervises the edge node, displays anomaly statuses, and performs OTA updates. The edge node is mounted directly on an industrial robotic arm and offers ultra-low power consumption (0.55mW) and low cost (£69), making it ideal for real-time processing of IMU data (accelerometer, gyroscope, and magnetometer readings).

<sup>4</sup>We will refer to this as "edge" or "Nicla" from now on.

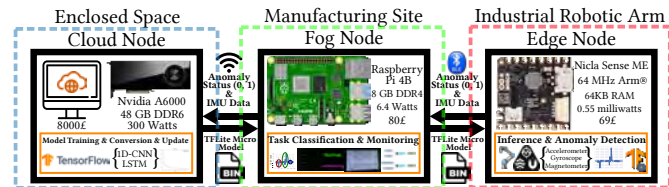


Fig. 3: The anomaly detection framework.

Anomalies are identified by comparing the prediction residuals against a pre-determined threshold; if the residuals exceed this threshold, the corresponding sample window is classified as anomalous. This setup ensures low-latency detection and immediate response, with the fog node providing detailed task and anomaly status indicators as the edge node is limited with possible LED use as an indicator. Figure 3 demonstrates the complete anomaly detection framework.

### D. Anomaly Detection Model & Methodology

We conduct regression on nine input features from the accelerometer, gyroscope, and magnetometer, each providing three-axis information. Anomalies are detected by comparing the residuals of predictions against a preset threshold determined through grid search. Using the root mean square error (RMSE) as our loss metric helps mitigate the effects of point anomalies or noise. This approach enhances real-time detection capabilities by leveraging machine learning models on edge IoT devices, ensuring low-latency responses essential for industrial applications. This work extends our prior research [61] by introducing new anomaly types and assessing performance directly on the edge device, addressing practical limitations and emphasizing the benefits of real-time detection for industrial robotic arms.

We chose TensorFlow for its open-source nature and its TFLite Micro sub-framework, designed by Google for resource-limited edge devices, particularly ARM-based CPUs. This framework supports key TensorFlow operations like convolutional, dense, pooling layers, and activation functions but lacks native support for 1D layers. To address this, we simulate 1D layers using 2D counterparts by adding an extra dimension to the 1D input data, applying 2D convolutional and max pooling layers with adjusted kernel and pooling sizes to mimic 1D operations. After processing, data is reshaped back to 1D for further analysis. Our LSTM model implementation directly stacks two LSTM layers followed by a dense layer, optimizing network architectures through grid search. This setup allows us to implement efficient ML models on edge devices, ensuring real-time anomaly detection essential for industrial applications.

The conversion from TensorFlow to TensorFlow Lite Micro (TinyML) is performed using the TensorFlow Lite converter. This process applies graph optimizations such as constant folding, pruning of unused operations, and removal of training-specific layers (e.g., dropout layer). The model is then converted into a compact FlatBuffer format optimized for embedded systems. These optimizations reduce model size while maintaining full precision and accuracy, as confirmed by our findings when no additional compression methods, such as

pruning or quantization, were applied. Since TinyML operates on resource-constrained edge devices, this conversion provides efficient inference without requiring specialized hardware acceleration. As depicted in Fig. 5 and explained in this section, the converted model runs directly on an ultra-low-power edge device (Nicla Sense ME) which enables real-time anomaly detection with minimal computational overhead.

The 1D-CNN model, shown in Figure 4, utilizes nine features ( $x$ ,  $y$ ,  $z$  axes of IMU data). To prevent overfitting due to input periodicity, we implemented various mitigation measures, stopping training if there is no loss improvement over five epochs. Data is split into 60% training, 20% validation, and 20% test sets, with hyperparameters optimized via grid search. RMSE is the selected loss metric. The model includes two 1D convolutional layers followed by a max pooling layer. During both training and anomaly detection, a sliding window approach is applied. If the overall RMSE of a window exceeds a pre-determined threshold, the window is classified as anomalous. Anomaly detection data is excluded from the training phase.

#### E. System Setup & Data Circulation

**Edge devices.** We utilize the Nicla Sense ME edge development board, developed by Bosch, featuring a 64 MHz Arm Cortex M4 microcontroller. The board includes BHI260AP (accelerometer and gyroscope) and BMM150 (magnetometer) IMU sensors, enabling the generation of 9 degrees-of-freedom (9DOF) IMU data. Quaternion data are generated via the Mahony [62] algorithm. The edge board supports BLE connectivity for wireless data transfer but lacks a WiFi chip. It is powered via a Raspberry Pi 4B (fog device) over USB 3.0 and is capable of running ML models while being strategically positioned on a bridge between the upper (wrist 1, 2, and 3) and lower (base, shoulder, and elbow) joints of the robotic arm using a breadboard attached with clamps.

**Fog device.** The fog device employed in our setup is a Raspberry Pi 4B running DietPi OS, a lightweight version of the Raspberry Pi OS designed for energy efficiency. It is powered by a network switch connected to the nearest power socket, offering Power over Ethernet (PoE) capability. This setup simplifies power distribution by eliminating the need for individual power supplies for each fog device. The Raspberry Pi 4B has sufficient capacity to simultaneously power multiple edge devices (four via USB: two 3.0 and two 2.0). The positioning of the fog device ensures that its movements are not disturbed by the USB cable powering the edge. In our system, the fog device serves as a Human-Machine Interface (HMI) and is referred to as PiHMI. The screen on the PiHMI is used for real-time monitoring, playing a key role in indicating anomalies, as demonstrated in previous incidents [4].

**Cloud device.** The cloud device in our setup is a data science workstation equipped with an NVIDIA RTX A6000 GPU. This high-performance GPU enables efficient and fast model training, making it suitable for handling resource-intensive tasks such as training neural network models. The cloud device accomplishes five major tasks: (I) training the machine learning models, (II) determining if an update of the ML

TABLE III: Device Tech Specs Comparison

	Nicla Sense Me	Raspberry Pi 4B	Cloud
GPU	N/A	N/A	NVIDIA RTX A6000
CPU	Arm® Cortex M4	Quad core Cortex-A72	Intel(R) Xeon(R)
RAM	64KB	4GB	128GB
Storage	2MB	32GB	2TB
Connectivity	Bluetooth 4.2	WiFi 802.11b/g/n - Bluetooth 5.0	Ethernet

model is required, (III) monitoring and tracking the active manufacturing task, (IV) initiating the process of updating the machine learning model, and (V) storing the data on a cloud server to mimic data historians seen in legacy industrial systems. The workstation is located in a restricted-access room and is accessible only via VPN connection to ensure data security and privacy. Figure 5 demonstrates the utilized devices, their locations, and the way they are powered. Table III compares the technical specs of each asset.

**Automated data flow and utilized software tools.** The proposed system operates autonomously without the need for further human intervention after the initial model deployment. USB 3.0 provides ample power output of 1.5 Amperes, sufficient for most edge devices, negating the need for additional power sources. Using batteries would offer greater flexibility and portability for edge devices but would require a custom case. We use a PoE HAT to power both the screen and the Pi from a single source. The model is initially trained in the cloud, converted into a TFLite file, then into a C++ file, and transmitted over WiFi to the fog device. The Python script on PiHMI sends IMU data and anomaly status to the cloud, while data in InfluxDB is visualized via a Grafana dashboard. The cloud-based Python script initiates new model training when anomalous windows fall below a specific threshold. Model update decisions depend on the severity of anomalies over a set timeline.

Edge data are securely transmitted over BLE using encryption to ensure data confidentiality. Node-RED is utilized on PiHMI to facilitate streamlined control, while Grafana is used for monitoring, with data fetched from the InfluxDB time series database. This setup enables real-time visualization and analysis of edge data. Python scripts run on both the cloud device and PiHMI to transfer the required data. Cloud data are logged and stored in Google Drive, serving as an essential repository for data analytics and historical analysis. This approach resembles the concept of data historians commonly employed in industrial systems to maintain comprehensive records and enable in-depth insights into system performance and behavior. The model deployment and system architecture are presented in Figure 5.

#### F. IoT Monitoring System

We introduce a method for real-time monitoring in industrial settings, focusing on environments with industrial robotic arms. Our system utilizes an IoT device to gather IMU data from the arm, which is then wirelessly transmitted to a local fog device (PiHMI) for task classification and supervision. This data transmission is facilitated over BLE, with the edge using an nRF52832 microcontroller for BLE 4.2 connectivity. For the development of our real-time monitoring system, we

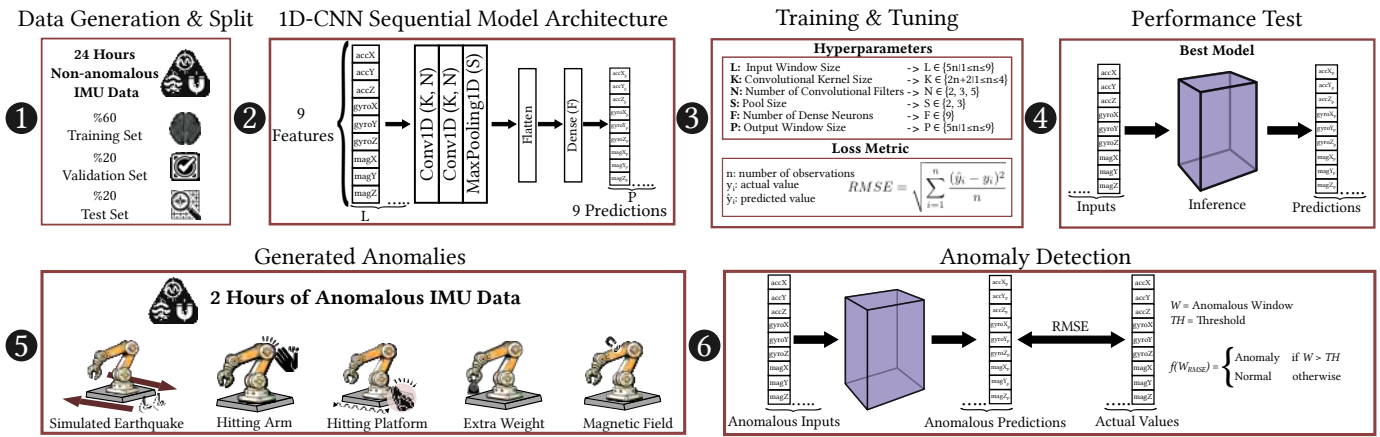


Fig. 4: Architecture of the utilized 1D-CNN model and anomaly detection methodology. The 1D-CNN processes IMU data, uses grid search for hyperparameter optimization, and detects anomalies with RMSE as the loss metric.

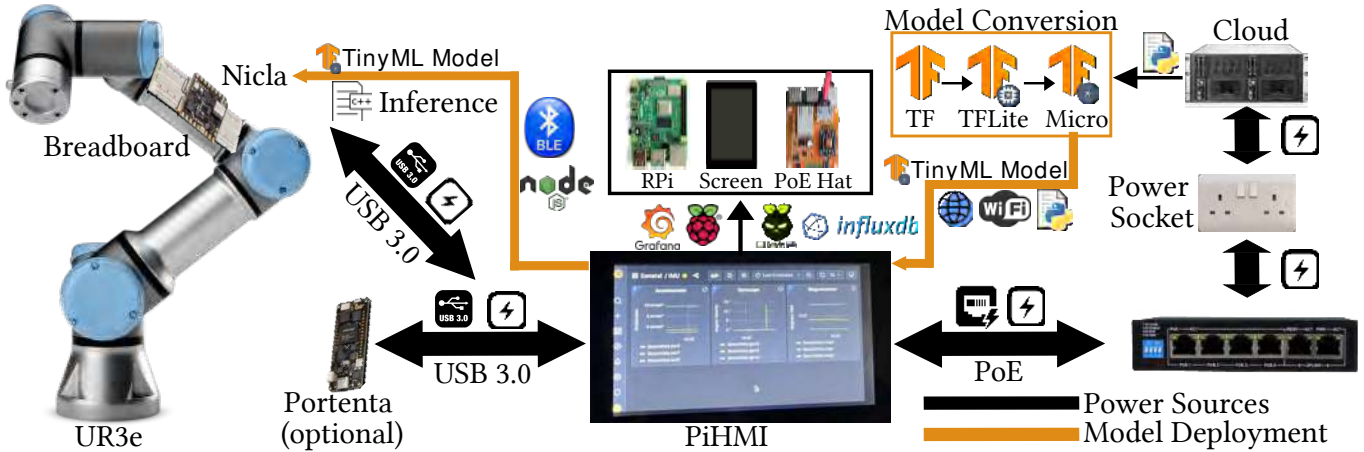


Fig. 5: End-to-end system architecture with power management, model training and deployment, and data visualization setup.

employed Node-RED, an open-source flow-based programming tool, and developed a Node-RED package<sup>5</sup> to facilitate the reception of data from the edge device at the fog layer allowing continuous monitoring of IMU data. We have chosen InfluxDB for data storage, similar to ICPS data historians. Grafana extracts IMU data from InfluxDB and displays it in real-time on the PiHMI screen, with value mapping handling type conversion. We also modified the Node.js package to include an additional anomaly status. Figure 6 demonstrates the fog screen during arm operations.

#### IV. ANOMALY GENERATION

In this paper, we focus on anomalies that generate physical consequences representing potential data integrity threats like cyber-physical attacks or degradation-induced faults. An essential physical characteristic of industrial robotic arms is their trajectory, as even minor deviations can lead to severe consequences. We also introduce an invisible movement-based anomaly involving unexpected magnetic fields due to their potential impacts and imperceptibility, rendering continuous on-site monitoring inefficient. We implement five distinct

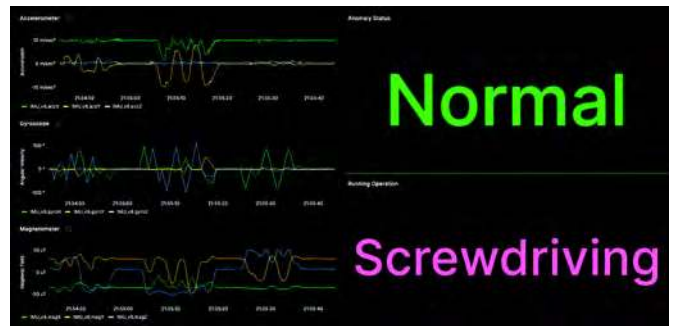


Fig. 6: Demonstrates the screen of the fog node.

anomalies in a controlled setting to safeguard all involved assets. While the earthquake simulation is ongoing, the other anomalies occur once during each task. Each anomaly scenario is documented in its respective CSV file.

The introduced anomalies include an earthquake simulation, where the platform is periodically shaken to emulate earthquake effects. Another anomaly involves hitting the arm, where the edge board positioned between the elbow and wrist 1 joints is directly impacted on the elbow with a gentle slap to generate a collision anomaly. Additionally, hitting the platform

<sup>5</sup><https://www.npmjs.com/package/node-red-contrib-ble-sense>

directly generates another anomaly. Adding extra weight is simulated by the gripper holding a custom-made object filled with ball bearings, adding a total weight of 4kg. Lastly, an unexpected magnetic field is generated by moving a magnet past the edge board.

Figure 7 demonstrates each anomaly case. For example, the Y-axis of the accelerometer shows noise-like consequences for the earthquake simulation, distinct peaks for hitting the arm and platform, a noise-filtering effect for the extra weight, and a prominent flipping effect for the magnetic field anomaly. Among these, only the earthquake simulation persists throughout the entire test, whereas the others are generated on a per-task basis.

## V. EVALUATION

### A. Quaternion and IMU Data Comparison

The edge board offers two modalities for IMU data acquisition: calibrated and uncalibrated/raw. It captures sequential tasks of screwdriving, painting, and pick-and-place, executed with the home position marking the transition. During these transitions, edge produces either calibrated IMU data or Mahony quaternion data. We executed this sequence over 24 hours for both data types. Our observations, as shown in Figure 8, indicate that Mahony quaternions exhibit inconsistencies, such as drift and oscillatory patterns, which are absent in the calibrated IMU data that displays periodicity. Additionally, externally generated quaternions differ from those produced by the edge. Despite verifying the correctness of our code and setup, we could not identify the reason for the lack of periodic behavior in the quaternion data. However, this investigation falls outside the scope of our work. Our primary focus is to evaluate the readiness and suitability of the data types for industrial robotic arm-related anomaly detection applications. This evaluation led us to favor calibrated IMU data over Mahony quaternion data due to its ability to detect linear motion-related anomalies.

The generated quaternions (Figure 8a) exhibit oscillatory behavior, specifically in the  $qY$  component, accompanied by drift. This behavior must be removed to create efficient sample windows for our ML models, but the removal process introduces computational overhead, and there is no guarantee that the resulting sample windows will exhibit the desired characteristics such as periodicity. Conversely, the IMU data (Figure 8b) clearly shows periodicity, which is crucial as we define our input size for the ML algorithms based on periodicity, which is also affected by generated anomalies.

### B. Correlation Analysis

The intrinsic characteristics of IMU attributes result in interrelations among them. The high degree of correlation among IMU features can pose multiple challenges: (I) an elevated risk of overfitting due to redundant information, (II) diminished interpretability, and (III) increased computational complexity. The performance of the same ML model can vary when applied to correlated versus non-correlated IMU data, potentially leading to misinterpretations. Therefore, we examine the extent of correlation in the IMU data used. Figure

9 presents a heatmap demonstrating the correlations among calibrated IMU data, with the values representing the Pearson correlation coefficient. Given that the highest coefficient is just above 0.7, we do not employ dimensionality reduction, as our evaluation indicates a moderate level of correlation, allowing us to retain the essential information within the data without concern for excessive redundancy or overfitting.

### C. Data Frequency Analysis

The sampling rate is a critical factor in real-time anomaly detection systems as it sets an upper threshold for data frequency, enabling more detailed analysis but also increasing computational complexity. Edge development board manufacturers preset key parameters like sampling rate, offset, range, and resolution to ensure stability. This precaution is necessary because components in close proximity, such as a temperature sensor positioned near the CPU, may generate elevated values. Edge utilizes two different IMU sensors (BHI260AP for accelerometer and gyroscope, and BMM150 for magnetometer). An unexpected case is revealed during the data frequency test. When the sampling rate is increased beyond 10Hz, regardless of the extent of the increase (e.g., 20Hz or 100Hz), the frequency eventually drops back to 10Hz. Experiments conducted exclusively with quaternion data or 9-DoF IMU data sampled at a rate of 10Hz exhibited no occurrence of a frequency drop. Therefore, we utilize the dataset with a 10Hz frequency for the remainder of the work. All data transmission is performed over BLE.

### D. Task Classification

In the proposed scenario, the same industrial robotic arm performs three consecutive tasks. The first step in the classification process is accurate window sampling. The arm returns to its home position and stays idle for 5 seconds after completing each task. Feature visualization (see Figure 8b) reveals that the Z-axis of the gyroscope exhibits a distinctive pattern with minimal noise interference, with data points revolving around zero when the arm is idle. We sample windows by evenly distributing the idle points, with each window starting and ending with approximately 2.5 seconds of idle data, corresponding to 25 data points at a collection rate of 10 Hz. We verify the sampling integrity by comparing the mean of each sample window per task. There are a total of 4422 sample windows, equally distributed among the tasks (1474 samples per task). The sample analysis (see Table IV) shows that painting is the longest process, followed by screwdriving and pick-and-place, while idle points between tasks range from 48 to 73. This variation is primarily due to the near-zero initialization of the painting task. In some samples, the near-zero values are considered idle because the magnitude of actual values is less than the noise. Figure 10 demonstrates each task.

The goal is to classify tasks in real-time, enabling immediate identification of the current task when an anomaly is detected. We compare two computationally efficient rule-based heuristic methods: peak detection using parabolic interpolation and

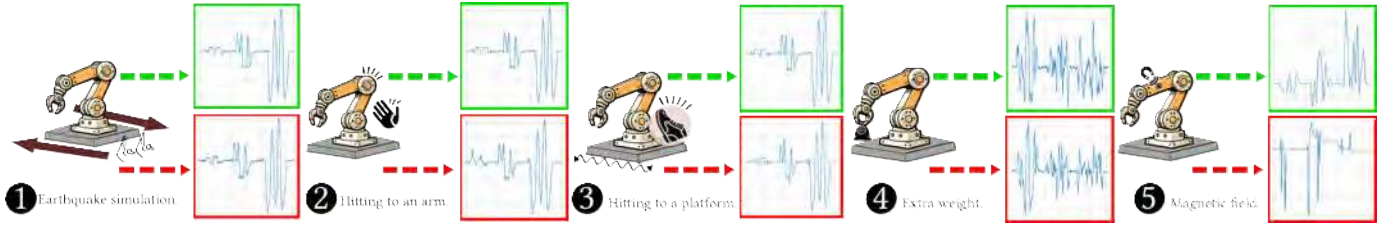
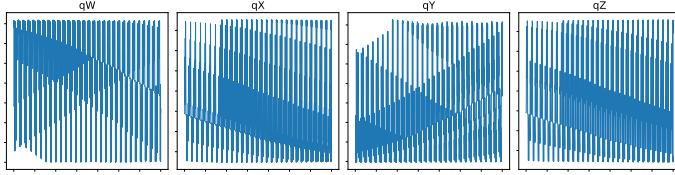
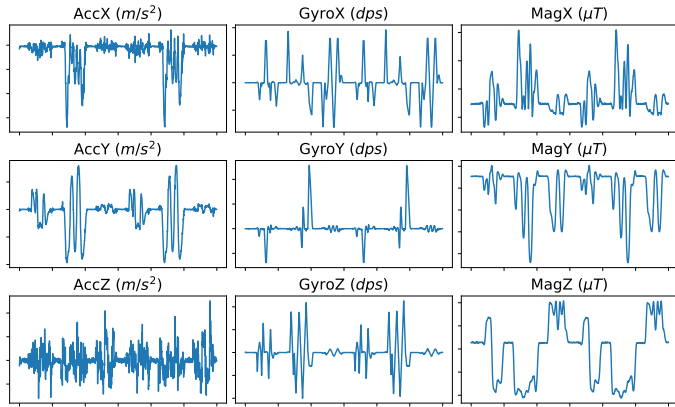


Fig. 7: Anomalies and their generation mechanisms, showing examples of non-anomalous and anomalous samples.



(a) Mahony quaternions generated by Nicla.



(b) Calibrated IMU generated by Nicla.

Fig. 8: Comparison of IMU data acquisition modes: Mahony quaternions (a) and calibrated IMU data (b).

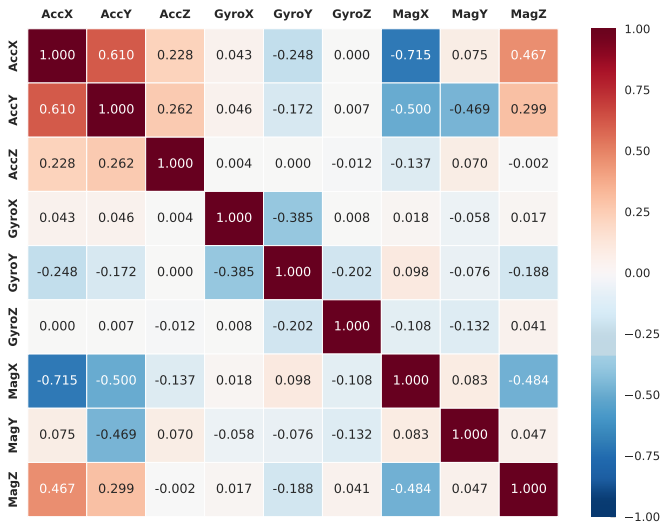


Fig. 9: The correlation heatmap of IMU data. The highest correlated feature pairs are  $AccX - AccY$ , and  $AccX - MagX$ . The map confirms that the features do not highly correlate as nearly all Pearson coefficients are between 0.7 and -0.7.

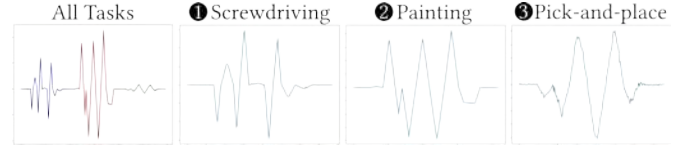


Fig. 10: Example sample windows per task. Data correspond to a Z-axis of gyroscope.

TABLE IV: Sample Analysis

Metric / Task	Screwdriving	Painting	Pick-and-place	Idle Points
Maximum Length / Mean	205 / 0.08238	214 / 0.58915	192 / 0.12051	73 / -
Minimum Length / Mean	191 / -0.25615	207 / -0.44017	179 / -0.04802	48 / -
Mean Length / Mean	197.94 / -0.04329	210 / 0.07074	184.62 / 0.02435	56.28 / -

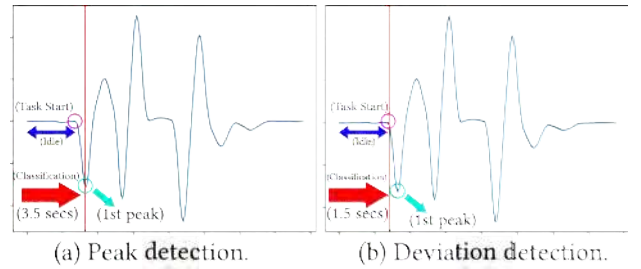


Fig. 11: Comparison of classification methods.

deviation detection through first derivative thresholding. Applying a rolling mean filter enhances the performance of both methods when utilizing rolling windows. The activation of rolling windows is governed by a rule that identifies idle periods within the filtered gyroscope data on the Z-axis. These idle periods are detected based on a specific threshold determined by the magnitude of noise; setting a threshold lower than the noise magnitude could lead to incorrect window sampling, as the process may misinterpret noise as arm activity. By accurately distinguishing between idle and active periods, we create sample windows that are then evenly distributed among the three tasks for further analysis. Peak detection relies on the presence of peaks, whereas deviation detection depends on idle points. We observe that the second method classifies tasks earlier, as deviations are detected before peaks arise.

Figure 11 compares these methods. The deviation detection method allows for quicker task classification, with screwdriving taking around 1.5 seconds, painting requiring approximately 2.5 seconds, and pick-and-place being identified in just 0.6 seconds. A moving average filter is applied to the data to enhance the accuracy of detection.

TABLE V: Evaluation of Common Thresholding for 1D-CNN

Performance Metrics	RMSE Window Size					
	45	50	100	150	200	265
F1-Score	0.675	0.644	0.807	0.912	0.924	0.946
Precision	0.98	0.993	0.971	0.979	1	1
Recall	0.514	0.477	0.691	0.853	0.859	0.898
Accuracy	0.867	0.859	0.912	0.956	0.962	0.973
FPR	0.003	0.001	0.007	0.006	0	0

### E. Anomaly Detection

Our goal is to simulate real-world scenarios where anomalies happen unexpectedly. To achieve this, we introduce anomalies in each task without recording their timestamps. This creates uncertainty about when an anomaly starts or stops. Anomalies might span from one task to another, leading to potential mislabeling with shorter window sampling. The minimum window length guaranteed to be an anomalous window is 265. This number matches the sum of the longest sample and idle points. We also experiment with shorter windows because using them allows quicker detection which is essential for ICPS. The minimum RMSE window length we test is 45 as it must exceed the number of idle points to avoid windows filled solely with idle points while it also matches the input window size. This prevents data from two tasks mixing and ensures a task-independent anomaly system since 45 matches the minimum number of idle points.

As mentioned earlier, we evaluated two neural network models: 1D-CNN and LSTM, chosen for their confirmed efficiency and compatibility with TensorFlow Lite Micro. Our approach to anomaly detection involves regression, with thresholding techniques employed to identify anomalies (see Algorithm 1). Given the variety of anomaly types, we implemented both common and pairwise thresholding methods, enabling an assessment of the detectability of different anomaly types.

1) *1D-CNN*: Table V demonstrates our results using common thresholding for anomaly detection with a 1D-CNN. This approach employs a sliding window algorithm to label sample windows, as shown in Figure 4. We observed improved performance in the anomaly detection model as the sliding window size increased, as when the window contained 265 points which is the minimum size required to reliably capture an anomalous event. When selecting the sliding window size, multiple considerations arise. Increasing the window size decreases the sensitivity of the model to noise and enhances its ability to detect extended anomalies, as demonstrated by our continuous earthquake simulation. However, there is a trade-off to consider: extended windows might overlook brief anomalies due to their minimal influence on the overall RMSE. Our evaluation suggests that smaller windows result in a higher number of false positives, indicating that a larger window size is preferable for improving detection accuracy in our use case.

Table VI presents the outcomes of implementing thresholding in pairs, involving a non-anomalous set paired with one of the anomalous sets. The results are promising, especially for the earthquake and magnetic field cases, even with smaller window sizes. The continuous behavior of the earthquake sim-

### Algorithm 1 Sliding Window-based Anomaly Detection

---

**Require:**

- Test data  $X \in \mathbb{R}^{n \times 9}$
- Mean  $\mu_{training} \in \mathbb{R}^{1 \times 9}$
- Std. dev.  $\sigma_{training} \in \mathbb{R}^{1 \times 9}$
- Threshold list  $T \in \mathbb{R}^k$
- Window size  $W \in \{45, 50, 100, 150, 200, 265\}$

**Ensure:** A label list  $P \in \{0, 1\}^{|S|}$  for each threshold in  $T$ , where  $S$  is the rolling-sum array of RMSE values

**(A) Generate RMSE array  $R$  with first window:**

- 1:  $\hat{X} \leftarrow \frac{X - \mu_{training}}{\sigma_{training}}$  ▷ Normalize data
- 2:  $R \leftarrow []$  ▷ Will store RMSE for each step
- 3: **for**  $i \leftarrow 1$  to  $n - W$  **do**
- 4:  $W_{win} \leftarrow \hat{X}[i : i + W - 1, :] \in \mathbb{R}^{W \times 9}$
- 5:  $\hat{y} \leftarrow f_{ML}(W_{win}) \in \mathbb{R}^{1 \times 9}$  ▷ Model predicts next point
- 6:  $y \leftarrow \hat{y} \cdot \sigma_{training} + \mu_{training}$  ▷ Inverse-normalize
- 7:  $r_i \leftarrow \sqrt{\frac{1}{9} \sum_{j=1}^9 (y_{i,j,target} - y_{i,j})^2}$  ▷ RMSE for time step  $i$
- 8:  $R.append(r_i)$
- 9: **end for**

**(B) Apply second window to  $R$ : Rolling sum array  $S$ :**

- 10:  $S \leftarrow []$
- 11: **for**  $i \leftarrow W$  to  $|R|$  **do**
- 12:  $S_i \leftarrow \sum_{j=i-W+1}^i R_j$
- 13:  $S.append(S_i)$
- 14: **end for** ▷  $|S| = |R| - W + 1$

**(C) Thresholding to produce anomaly labels:**

- 15: **for**  $t \in T$  **do**
- 16:  $P \leftarrow []$  ▷ List of anomaly labels for threshold  $t$
- 17: **for**  $i \leftarrow 1$  to  $|S|$  **do**
- 18: **if**  $S_i > t$  **then**
- 19:  $P.append(1)$  ▷ Anomaly
- 20: **else**
- 21:  $P.append(0)$  ▷ Normal
- 22: **end if**
- 23: **end for**
- 24: (Evaluate  $P$  for threshold  $t$ )
- 25: **end for**

---

ulation and the deviations seen in Figure 7 when a magnet is near a magnetometer demonstrate the role of the magnetometer in orientation and detecting unexpected magnetic fields. On the other hand, the anomaly detection in the extra weight scenario is less effective, as added weight tends to dampen arm vibrations, resembling the effect of a smoothing filter, and thus poses a greater challenge for detection. The scenario of hitting a platform shows lower effectiveness in anomaly detection, due to the absence of direct impact on the arm. Despite these variations, accuracy remains high (no less than 92%) in most scenarios. Next, we explore the performance of the LSTM model.

2) *LSTM*: The same methodology used for the 1D-CNN, containing window size and thresholding techniques, was applied to the LSTM model, with results detailed in Table VII. Our analysis indicates that LSTM consistently surpasses 1D-CNN across all RMSE window sizes, especially at smaller window dimensions. Additionally, LSTM approaches near-perfect accuracy with an RMSE window size of approximately 200. Table VIII further confirms the superior performance of LSTM over 1D-CNN through pairwise thresholding analysis. Next, we will focus on the edge implementations of both algorithms.

TABLE VI: Evaluation of Pairwise Thresholding for 1D-CNN

Anomaly Type	Window Size	F1-Score	Precision	Recall	Accuracy	FPR
Hitting Arm	45	0.646	0.836	0.526	0.963	0.006
	50	0.651	0.854	0.526	0.964	0.006
	100	0.736	0.912	0.617	0.972	0.003
	150	0.884	0.938	0.836	0.986	0.003
	200	0.984	0.987	0.98	0.998	0.0008
	265	0.978	0.996	0.96	0.997	0.0002
Hitting Platform	45	0.548	0.57	0.528	0.931	0.0337
	50	0.552	0.577	0.528	0.932	0.0329
	100	0.641	0.701	0.591	0.948	0.021
	150	0.851	0.865	0.836	0.977	0.0109
	200	0.974	0.982	0.966	0.996	0.0014
	265	0.965	0.981	0.95	0.994	0.0014
Earthquake Simulation	45	0.745	0.767	0.725	0.969	0.014
	50	0.764	0.789	0.74	0.972	0.0127
	100	0.913	0.903	0.924	0.989	0.0063
	150	0.99	0.986	0.994	0.998	0.00089
	200	1	1	1	1	0
	265	1	1	1	1	0
Extra Weight	45	0.418	0.39	0.45	0.927	0.043
	50	0.413	0.376	0.459	0.924	0.046
	100	0.493	0.426	0.584	0.93	0.048
	150	0.735	0.753	0.718	0.97	0.014
	200	0.914	0.93	0.9	0.99	0.0041
	265	0.926	0.948	0.905	0.991	0.002
Magnetic Field	45	0.999	1	0.998	0.999	0
	50	0.999	1	0.999	0.999	0
	100	1	1	1	1	0
	150	1	1	1	1	0
	200	1	1	1	1	0
	265	1	1	1	1	0

TABLE VII: Evaluation of Common Thresholding for LSTM

Performance Metrics	RMSE Window Size					
	45	50	100	150	200	265
F1-Score	0.897	0.911	0.926	0.980	0.999	0.999
Precision	0.996	0.991	0.999	0.973	1.0	0.999
Recall	0.815	0.843	0.863	0.986	0.998	1.0
Accuracy	0.949	0.956	0.963	0.989	0.999	0.999
FPR	0.0009	0.0026	0	0.0096	0	0.0003

TABLE VIII: Evaluation of Pairwise Thresholding for LSTM

Anomaly Type	Window Size	F1-Score	Precision	Recall	Accuracy	FPR
Hitting Arm	45	0.748	0.941	0.621	0.973	0.0026
	50	0.768	0.935	0.651	0.975	0.0030
	100	0.866	0.970	0.782	0.984	0.0016
	150	0.941	0.995	0.893	0.993	0.0002
	200	0.999	1	0.998	0.999	0
	265	1	1	1	1	0
Hitting Platform	45	0.865	0.951	0.793	0.980	0.0034
	50	0.877	0.952	0.814	0.982	0.0034
	100	0.920	0.967	0.878	0.988	0.0024
	150	0.996	0.996	0.997	0.999	0.0003
	200	1	1	1	1	0
	265	1	1	1	1	0
Earthquake Simulation	45	0.991	0.991	0.991	0.998	0.0005
	50	0.994	0.994	0.994	0.999	0.0003
	100	1	1	1	1	0
	150	1	1	1	1	0
	200	1	1	1	1	0
	265	1	1	1	1	0
Extra Weight	45	0.826	0.957	0.726	0.982	0.0019
	50	0.830	0.972	0.724	0.982	0.0012
	100	0.874	0.961	0.801	0.986	0.0019
	150	0.998	0.999	0.997	0.999	0
	200	1	1	1	1	0
	265	1	1	1	1	0
Magnetic Field	45	0.995	1	0.991	0.999	0
	50	0.996	1	0.992	0.999	0
	100	0.998	1	0.997	0.999	0
	150	1	1	1	1	0
	200	1	1	1	1	0
	265	1	1	1	1	0

F. Edge Deployment and Evaluation

One of the significant engineering contributions of this study is developing a TFLite Micro port for the edge, which utilizes an ARM-based CPU. To the best of our knowledge, Nicla lacks official support from TFLite Micro, as it has not undergone testing, nor is a pre-existing port available, according to TensorFlow documentation. We are among the first to evaluate its compatibility with TFLite Micro, beyond commercial off-the-shelf products, and provide a functional, verified port. The porting process required adjustments in the libraries used to eliminate conflicts. Hence, the offered seamless functionality is guaranteed only with the specific library versions employed in our project. Future updates to these libraries may require further changes to the port.

The primary limitation for edge computing in our setup is the RAM, restricted to 64KB. There are two main components consuming RAM: (I) TFLite Micro operations, which allocate specific RAM space through the **tensor arena** parameter, and (II) the functionalities necessary for OTA updates. Our investigation revealed that enabling OTA updates requires initializing Nicla with BLE activated, as shown below:

```
while(!BHY2.begin(NICLA_BLE));
```

Listing 1: Memory Error

Activating BLE for its intended use in controlling the Nicla, such as for sensor initialization/configuration or displaying sensor data on a web server, led to conflicts. To resolve these issues and achieve additional RAM savings, we modified the source code. Subsequent testing to evaluate RAM availability after loading essential libraries revealed an increase in total available space for necessary operations to 10,128 bytes, compared to 9,088 bytes with the unmodified code. To enhance RAM efficiency further, we implemented a sliding window approach with iterative RMSE calculation, conserving more RAM. This strategy is important, as our analysis (see Table V) indicates that larger window sizes improve anomaly detection accuracy. The iterative RMSE calculation method helps minimize the impact of RMSE window size on memory.

We determined that the maximum allocatable memory for the tensor arena is 7,012 bytes, a significant increase from 5,260 bytes in the unaltered version. This conclusion came from tests involving different memory allocations within the arena for the required operations/libraries. However, accurately forecasting the memory needs is complex and requires actual implementation. Therefore, we examined the impact of model parameters on the tensor arena size required. Table IX displays the results of our model size evaluation, with key metrics defined as follows: **Input** represents the number of time steps inputted into the model; **Output** specifies the model output length; **Shift** refers to the number of steps advanced to prevent re-predicting any time step during inference. The dimensions of **Filter** and **Pooling** correspond to the lengths of one-dimensional filters. **Compatibility** shows cases where the model size surpasses the available RAM.

TensorFlow Lite is designed for mobile devices and single-board computers such as the Raspberry Pi, whereas TFLite Micro is aimed at edge development boards equipped with 32-

TABLE IX: The Effect of Hyperparameters on Model Size

Model	Input	Output	Shift	Number of filters	Filter Size	Pooling Size	Model Size (bytes)	Compability	Tensor Arena (bytes)
1D-CNN	45	1	1	8	3	3	8772	✓	5760
1D-CNN	45	1	1	8	5	3	9668	✓	5568
1D-CNN	45	1	1	16	3	3	15844	✓	6952
1D-CNN	45	1	1	32	3	3	34404	X	12584
1D-CNN	45	1	1	8	3	2	10884	✓	5768
1D-CNN	45	5	5	8	3	3	24004	✓	5768
1D-CNN	45	10	10	8	3	3	42904	✓	5768
1D-CNN	45	45	45	8	3	3	175204	X	5768
1D-CNN	40	1	1	8	3	3	8596	✓	5256
1D-CNN	30	1	1	8	3	3	7444	✓	4264
LSTM	45	1	1	8	N/A	N/A	9608	✓	5976
LSTM	45	1	1	16	N/A	N/A	20648	X	8920
LSTM	45	1	1	32	N/A	N/A	61160	X	15184
LSTM	45	45	45	8	N/A	N/A	8876	✓	6976

bit microcontrollers. Contrary to our initial belief that models with greater computing power and storage would surpass their lighter counterparts, we found that, without quantization, performance is comparable between TFLite and TFLite Micro models, and their sizes remain the same. This observation is important, indicating that under specific conditions, where storage and inference time are within acceptable thresholds, TFLite Micro models can be just as effective. However, due to RAM constraints on the edge, we explored quantization. In the absence of quantization, model weights and biases are stored as 32-bit floating-point numbers, each consuming 4 bytes of memory. While the ARM Cortex-M4 processor in the Nicla technically supports 32-bit floating-point operations, practical RAM limitations may render this impractical.

### G. Quantization

TensorFlow Lite introduces three post-training quantization techniques, each impacting model size and performance distinctively:

- **Dynamic Range Quantization:** In this method, weights are converted from floating points to integers during model conversion, while activations are dynamically quantized to 8 bits at runtime. Computations utilize 8-bit integers, with outputs in 32-bit floating-point format.
- **Full Integer Quantization:** This technique offers the most significant reduction in model size by converting all model elements, including weights, biases, activations, and inputs, to 8-bit integers. It is anticipated to slightly reduce detection accuracy.
- **Float16 Quantization:** It halves model size by converting 32-bit floating-point parameters to 16-bit floating points, with a lesser expected impact on performance compared to full integer quantization.

Table X compares the sizes of models using different quantization methods with those of the main and non-quantized TFLite models. To evaluate the effect of quantization on model performance, we calculated the MSE for an anomaly scenario (hitting arm). Our findings suggest that dynamic range quantization, which does not decrease model size, may not be beneficial for smaller 1D-CNN models. The Float16 quantization technique reduces the model size by **2220** bytes and

maintains performance nearly equivalent to the unquantized model. In contrast, full integer quantization, saving **3308** bytes, results in a higher MSE, indicating a possible decrease in model accuracy.

TABLE X: Model Size Comparison

Model	Post-training Quantization Method	Size (bytes)	MSE
TensorFlow	N/A	51,607	N/A
TFLite - 1	No Quantization	11692	$\approx 0$
TFLite - 2	Dynamic Range	11696	$\approx 0$
TFLite - 3	Float16	9472	$4.182 \times 10^{-7}$
TFLite - 4	Full Integer	8384	1.169

MSE: Mean Squared Error.

We evaluated the effectiveness of the **Float16** and **full integer** quantization methods, comparing them with the main model. Applying the same evaluation methodology across all models, Table XI presents our results, showing that the performance of the main model and the non-quantized TFLite model is equivalent. This indicates that in situations where the TFLite model size meets the required constraints, its performance can match that of the main model. However, this does not extend to all cases, particularly when a smaller model size is necessary due to RAM limitations. The fully integer-quantized model exhibits the lowest accuracy, failing to classify anomalies effectively. The low precision and high false positive rate (FPR) across different RMSE window sizes point to a frequent occurrence of false positives, rendering it unsuitable for industrial use. The Float16 quantization method provides results almost identical to the main model, showing its viability for applications requiring both efficiency and accuracy.

With Float16 quantization emerging as the preferred solution, we wanted to implement the model but faced another challenge. At the time of writing this work, the dequantize operation in TFLite Micro only accepts 8-bit signed integer, 16-bit signed integer, or 8-bit unsigned integer types as inputs, not aligning with 16-bit binary floating-point input, leading to compatibility issues. We used the Netron application [63] to examine the compatibility of the model and its structure. Faced with quantization barriers, we investigated RAM conservation alternatives, finding that modifying certain BLE-related buffers via firmware recompilation could free up about

TABLE XI: Performance Comparison of TFLite Models

Model	Performance Metrics	RMSE Window Size					
		45	50	100	150	200	265
Main Model	F1-Score	0.675	0.644	0.807	0.912	0.924	0.946
	Precision	0.98	0.993	0.971	0.979	1	1
	Recall	0.514	0.477	0.691	0.853	0.859	0.898
	Accuracy	0.867	0.859	0.912	0.956	0.962	0.973
	FPR	0.003	0.001	0.007	0.006	0	0
No Quantization	F1-Score	0.675	0.644	0.807	0.912	0.924	0.946
	Precision	0.98	0.993	0.971	0.979	1	1
	Recall	0.514	0.477	0.691	0.853	0.859	0.898
	Accuracy	0.867	0.859	0.912	0.956	0.962	0.973
	FPR	0.003	0.001	0.007	0.006	0	0
Full Integer	F1-Score	0.422	0.422	0.421	0.440	0.437	0.419
	Precision	0.276	0.267	0.269	0.351	0.312	0.275
	Recall	0.891	1.0	0.975	0.591	0.732	0.877
	Accuracy	0.347	0.267	0.287	0.600	0.501	0.357
	FPR	0.851	1.0	0.963	0.395	0.582	0.829
Float16	F1-Score	0.675	0.645	0.809	0.913	0.924	0.945
	Precision	0.984	0.993	0.970	0.978	1	1
	Recall	0.514	0.477	0.694	0.856	0.859	0.897
	Accuracy	0.867	0.859	0.912	0.956	0.962	0.972
	FPR	0.003	0.0011	0.0076	0.0066	0	0

3KB of RAM, as detailed in our GitHub repository. Because our edge device has only 64KB of RAM, even the modest memory savings from quantization were insufficient, and full integer quantization resulted in unacceptable accuracy losses. Due to compatibility challenges with quantized models, we concentrated on evaluating the non-quantized model for its compatibility with TFLite Micro, confirming all essential layers were supported. The absence of quantization removed the need for a dequantize step. Furthermore, given the lack of native support for 1D-CNN layers, we adapted by using 2D-CNN layers instead. To evaluate the performance, we employed two strategies. Initially, we conducted tests using the provided scripts. Then, data was transmitted via serial to perform inference via the TFLite Micro model and capture its outputs/predictions. Our results demonstrate that by dropping quantization and implementing firmware-level optimizations, the anomaly detection performance remains consistent with that of the main model. Hence, this should be considered as a primary task as it also prevents any sacrifice on the anomaly detection performance.

H. Power Usage & Detection Latency

Finally, we evaluated the inference time and power usage. Our observations showed that both the 1D-CNN and LSTM models have a similar power consumption of approximately **12.8 mA** during inference. The power consumption is measured via the digital USB power meter as shown in Figure 12. This implies that, in the absence of power usage optimization and with a constant current draw, a device equipped with a **10,000 mAh** battery could function for **32.55 days** (781.25 hours). Concerning inference times, enabling BLE significantly impacts them. For example, without BLE, the inference duration for the 1D-CNN model is between **115 ms** and **116 ms**, but with BLE enabled, it fluctuates from **125 ms** to **250 ms**. Similarly, for the LSTM model, inference time without BLE spans **176 ms** to **176.1 ms**, increasing to **180 ms** to **250 ms** with BLE active. Therefore, the 1D-CNN model is faster. These inference durations indicate that achieving real-time anomaly detection at 10Hz, as targeted for the provided use case, is not viable due to potential delays. The most practical frequency attainable, given the inference times, is **3Hz**.

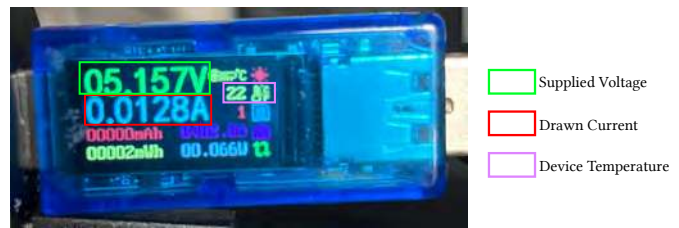


Fig. 12: Digital USB power meter used to measure power consumption during model inference.

I. Generalizability

To show the generalizability of our work, we extended our previous work [12] where we introduced a context-aware anomaly detection system that used externally gathered IMU data. We used a 1D-CNN model to detect movement-based anomalies in robotic arms that perform repetitive pick-and-place tasks as many robotic arms follow periodic patterns. We found that deviations in joint velocities appear clearly in externally attached IMU sensors. This periodicity makes the anomaly detection approach from our previous work applicable to similar industrial robotic arm scenarios.

In this paper, we further evaluate the generalizability of our approach using another publicly available robotic dataset [23]. This dataset provides joint velocity data categorized into four distinct classes based on variations in payload (1.6,lb and 4.5,lb) and operational speed (full-speed and half-speed), as illustrated in Fig.13. To adapt our previously developed 1D-CNN model for this classification task, we modified only the final dense layer and trained the network using a cross-entropy loss function. Despite the subtle nature of the differences introduced by varying payloads (see Fig. 13), our adapted model successfully achieved 100% classification accuracy. This outcome demonstrates that even a relatively simple neural network model can effectively identify subtle changes in robotic arm behavior caused by payload variations. The achieved high accuracy, as reflected clearly by the confusion matrix shown in Fig. 14, confirms that joint velocity-based analysis is sufficient to detect operational variations. Hence, our results reinforce the potential of deploying such lightweight yet effective algorithms on resource-constrained edge devices which enables real-time detection of deviations in robotic arm performance across different operational payloads and speeds.

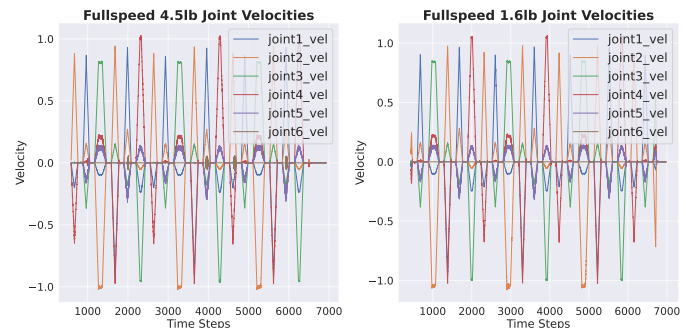


Fig. 13: Joint velocities when arm carries different payloads.

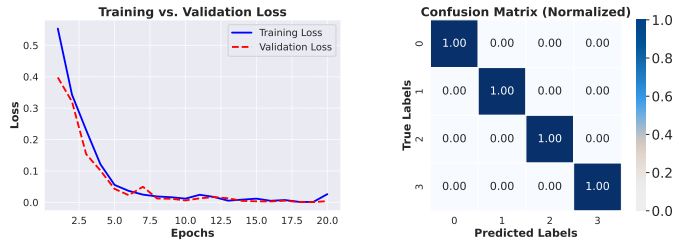


Fig. 14: The loss graph and the confusion matrix of the classification task.

## VI. LESSONS LEARNED & LIMITATIONS

(1) **Necessity of the presence of idle points.** Task classification approaches based on rule-based heuristics depend solely on the presence of consecutive idle points. The absence of these points can lead to failures in task classification and anomalies appearing post-idle points can cause misclassification. While anomaly detection remains possible, associating them with specific tasks becomes challenging. Noise further complicates classification by introducing delays. A rolling mean filter, sized based on noise duration, can mitigate this; however, if noise persists beyond the number of idle points, classification approaches fail.

(2) **Drawbacks of Tensorflow Lite Micro.** Edge development boards are resource-constrained devices designed to minimize power usage, resulting in limited support for certain operations in ML frameworks. Previous research (see Section II) shows that TFLite Micro is a common choice among embedded ML applications. This framework supports essential neural network elements like dense networks, 2D-CNNs, depthwise separable convolutions, and pooling layers. To implement 1D-CNNs, we tweak the 2D-CNN layer since one can simulate the other. Official tutorials for TFLite Micro primarily focus on performance comparisons between TFLite and TensorFlow models, lacking comprehensive guides for specific edge development boards, such as the Nicla. This requires customizing the framework for various use cases, introducing numerous bugs that require intensive debugging, especially when dealing with proprietary APIs. These issues often stem from APIs designed by different teams without considering the applicability of embedded ML applications while TFLite Micro is developed by an open-source community. In this paper, we ported TFLite Micro to the Nicla, resulting in numerous conflicts and a labor-intensive process. The provided code is tailored to meet the unique requirements of the Nicla edge development board.

(3) **Drawbacks due to limited RAM.** We investigated the possibility of creating an edge-based anomaly detection system. Determining feasibility is challenging, but we started by trying to use existing solutions before making any changes. A major challenge we faced was the limited RAM on the Nicla board. Our tests showed that running even a simple 1D-CNN model was difficult because a lot of RAM was needed for the BLE functionality, which was essential for sending data to the fog device. To manage this, we recompiled the main and BLE stacks to fit both our model and BLE communication,

identifying some settings that could be tweaked to save more RAM. These changes seemed not to affect functionality. A key issue is predicting how changes to the source code might affect the system, which is tricky without any guidance on buffer sizes. Therefore, a full system implementation is required for thorough testing. We found that limited RAM greatly impacts the feasibility, requiring specialized knowledge in embedded programming to overcome these limitations.

(4) **Edge-to-fog-to-cloud.** We have shown that it is feasible to construct a fully operational automated pipeline using solely open-source IoT tools. This approach not only enhances the customization of the pipeline but also broadens the applicability of our IoT-based anomaly detection system. A Node.js BLE package was the main component we found lacking, leading us to develop one that fits our specific needs. While this package was initially designed for our work, its open-source status means it can be easily modified to fit other boards or systems with little effort. One key challenge in deploying such edge anomaly detection systems is the limited means for providing insights or feedback. Edge development boards generally rely on LEDs for visual cues, which offer limited information compared to what is available through cloud-based tools. We addressed this limitation by developing a fog node as a HMI to visualize anomaly status, IMU data, ongoing tasks while enabling the adaptivity via OTA. However, the task of simultaneously monitoring multiple edge assets poses a challenge. The use of multiple fog nodes could provide a solution; however, implementing this approach in a real industrial setting is essential to evaluate its practical challenges and establish its viability.

(5) **Drawbacks of a sliding-window approach.** Sliding window techniques facilitate real-time anomaly detection through regression. In our examined use cases, a longer window consistently improved anomaly detection, particularly for contextual anomalies where context is time-bound, and anomalies occur over time. Earthquake and unexpected magnetic field scenarios provided better results with relatively shorter windows, while the hitting arm scenario showed the poorest performance due to the transient nature of the anomaly. Therefore, employing a sliding window approach with a fixed window size poses challenges for short-duration anomalies, as their impact on overall threshold parameters (RMSE in our case) becomes minimal. Considering a dynamic window size, as suggested by [64], could address this issue. Evaluating this method on industrial benchmark datasets would offer valuable insights.

(6) **Limited scope due to attachment location of an edge development board.** The robotic arm comprises six distinct joints, with each joint possessing individual mobility and the capability to move independently. The lower three joints of the arm are employed for executing larger, more robust movements, while the upper three joints are dedicated to carrying out finer, more precise motions. In order to ensure the accuracy and reliability of data collection, we choose to mount the edge development board onto the section of the arm that connects the upper and lower joints. However, this configuration presents challenges when the arm performs movements that result in minimal displacement in that specific part. A limitation is obvious during the initialization of a

painting task. Certain data points appear as idle because the movement is not clearly reflected on IMU data. Therefore, the position for attaching the edge development board to the industrial robotic arm needs predetermination based on the movement of the arm. A task change might require moving the board to a different joint or location.

(7) **Consistency across TensorFlow frameworks.** The TensorFlow has two sub-frameworks: TensorFlow Lite, and TFLite Micro. TensorFlow Lite is designed for mobile devices and single-board computers like the Raspberry Pi while TensorFlow Lite Micro targets edge development boards with 32-bit microcontrollers. Despite the expectation that the main model, with its greater computing power and storage, would outperform others, the loss values across all frameworks remain closely matched. The difference in loss is negligible in a way that it does not affect the accuracy. This is very promising as it means under certain circumstances where storage and inference time are between the acceptable limits the lite models can perform pretty well.

(7) **Observed oscillation and shift on quaternion data.** The built-in functions provided by Nicla (Bosch) developers are used to generate quaternions. The quartering representation displays a shifted oscillatory behavior. The idle position is not identifiable. The periodicity is not observed. While there might be several reasons for this (e.g., low sampling rate, drift, measurement noise, computation lag) it is obvious the use of quaternions to detect movement-based anomalies on an industrial robotic arm requires several pre-processes that introduce computational overhead to the edge development board.

(8) **Challenges in model update strategies.** The current implementation uses a placeholder threshold to trigger new model training when anomalous windows fall below a set value. This provides a starting point but does not fully address the complexities of deciding the best timing for updates. In security-critical applications, where conditions and threats change over time, update decisions must be specific to the domain. Factors such as operational constraints, anomaly severity and frequency, and the changing nature of industrial environments must be considered. More research is needed to develop adaptive update strategies that can decide when retraining is necessary. This will help maintain detection accuracy and system reliability.

## VII. CONCLUSION & FUTURE WORK

In this paper, we focused on the implementation of a low-power, edge-based ML system for real-time anomaly detection on industrial robotic arms, targeting disruptions in arm trajectory and magnetic field deviations. Utilizing a compact edge development board, we collected IMU data and performed inference using 1D-CNN and LSTM models. This data was then transmitted to a fog device for HMI visualization and subsequently relayed to the cloud. An updated model was sent back to the edge device via BLE, facilitated by a customized fog node. Our approach demonstrates the feasibility of building an automated anomaly detection pipeline on resource-constrained edge devices, providing an alternative to commercial ML-as-a-service solutions. This is achieved by utilizing

simple yet effective models, 1D-CNN and LSTM, that are highly compatible across both cloud and edge environments. As a result, we observed nearly identical anomaly detection performance across TensorFlow, TFLite, and TFLite Micro models when no quantization is applied. However, challenges such as adapting TFLite Micro to various edge boards due to potential firmware conflicts require extensive porting and embedded coding efforts. Specifically, limitations in RAM required source code modifications to decrease the size of pre-allocated buffers. This adjustment was essential to implement the LSTM model effectively, ensuring it could outperform the 1D-CNN while maintaining reliable data transmission over BLE. The entire process, utilizing open-source tools, is documented in our GitHub repository. Future work will focus on expanding compatibility to other ARM-based boards with IMU sensors and integrating additional sensor types. We also consider exploring online learning mechanisms to remove the need for a cloud node after training. Developing adaptive model update strategies to decide when retraining is necessary could help maintain detection accuracy and system reliability.

## VIII. ACKNOWLEDGMENT

This work is partially supported by EPSRC PETRAS (Grant No. EP/S035362/1) and the GCHQ National Resilience Fellowship. We would like to acknowledge the scholarship and support provided by Republic of Turkey Ministry of National Education.

## REFERENCES

- [1] J. Krumm, *Ubiquitous computing fundamentals*. CRC Press, 2018.
- [2] A. G. Frank, L. S. Dalenogare, and N. F. Ayala, "Industry 4.0 technologies: Implementation patterns in manufacturing companies," *International Journal of Production Economics*, vol. 210, pp. 15–26, 2019.
- [3] S. Jeschke, C. Brecher, T. Meisen, D. Özdemir, and T. Eschert, *Industrial internet of things and cyber manufacturing systems*. Springer, 2017.
- [4] H. Kayan, M. Nunes, O. Rana, P. Burnap, and C. Perera, "Cybersecurity of industrial cyber-physical systems: A review," *ACM Comput. Surv.*, vol. 54, no. 11s, sep 2022. [Online]. Available: <https://doi.org/10.1145/3510410>
- [5] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial internet of things," in *Proceedings of the 52nd Annual Design Automation Conference*, ser. DAC '15. New York, NY, USA: Association for Computing Machinery, 2015.
- [6] J. Giraldo, D. Urbina, A. Cardenas, J. Valente, M. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg, and R. Candell, "A survey of physics-based attack detection in cyber-physical systems," *ACM Comput. Surv.*, vol. 51, no. 4, jul 2018. [Online]. Available: <https://doi.org/10.1145/3203245>
- [7] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, jul 2009.
- [8] Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, and M. S. Hossain, "Deep anomaly detection for time-series data in industrial iot: A communication-efficient on-device federated learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6348–6358, 2020.
- [9] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge cloud offloading algorithms: Issues, methods, and perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, feb 2019. [Online]. Available: <https://doi.org/10.1145/3284387>
- [10] C. Banbury, C. Zhou, I. Fedorov, R. Matas, U. Thakker, D. Gope, V. Janapa Reddi, M. Mattina, and P. Whatmough, "Micronets: Neural network architectures for deploying tinyml applications on commodity microcontrollers," *Proceedings of machine learning and systems*, vol. 3, pp. 517–532, 2021.
- [11] P. Warden and D. Situnayake, *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*. O'Reilly, 2019.

- [12] H. Kayan, R. Heartfield, O. Rana, P. Burnap, and C. Perera, "Casper: Context-aware iot anomaly detection system for industrial robotic arms," *ACM Trans. Internet Things*, jun 2024. [Online]. Available: <https://doi.org/10.1145/3670414>
- [13] H. Yun, H. Kim, Y. H. Jeong, and M. B. Jun, "Autoencoder-based anomaly detection of industrial robot arm using stethoscope based internal sound sensor," *Journal of Intelligent Manufacturing*, vol. 34, no. 3, pp. 1427–1444, 2023.
- [14] V. Narayanan and R. B. Bobba, "Learning based anomaly detection for industrial arm applications," ser. CPS-SPC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 13–23.
- [15] Z. Luo, M. Yan, W. Wang, and Q. Zhang, "Non-intrusive anomaly detection of industrial robot operations by exploiting nonlinear effect," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 6, no. 4, jan 2023. [Online]. Available: <https://doi.org/10.1145/3569477>
- [16] F. Farbiz, Y. Miaolong, and Z. Yu, "A cognitive analytics based approach for machine health monitoring, anomaly detection, and predictive maintenance," in *2020 15th IEEE conference on industrial electronics and applications (iciea)*. IEEE, 2020, pp. 1104–1109.
- [17] D. Popov, A. Klimchik, and N. Mavridis, "Collision detection, localization & classification for industrial robots with joint torque sensors," in *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2017, pp. 838–843.
- [18] M. A. Costa, B. Wullt, M. Norrlöf, and S. Gunnarsson, "Failure detection in robotic arms using statistical modeling, machine learning and hybrid gradient boosting," *Measurement*, vol. 146, pp. 425–436, 2019.
- [19] P. Ferrari, S. Rinaldi, E. Sisinni, F. Colombo, F. Ghelfi, D. Maffei, and M. Malara, "Performance evaluation of full-cloud and edge-cloud architectures for industrial iot anomaly detection based on deep learning," in *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0&IoT)*. IEEE, 2019, pp. 420–425.
- [20] M. Usman, A. Jolfaei, and M. A. Jan, "Rasec: an intelligent framework for reliable and secure multilevel edge computing in industrial environments," *IEEE Transactions on Industry Applications*, vol. 56, no. 4, pp. 4543–4551, 2020.
- [21] S. Zhu, K. Ota, and M. Dong, "Green ai for iiot: Energy efficient intelligent edge computing for industrial internet of things," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 1, pp. 79–88, 2021.
- [22] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge cloud offloading algorithms: Issues, methods, and perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–23, 2019.
- [23] G. Qiao and B. A. Weiss, "Accuracy degradation analysis for industrial robot systems," in *International Manufacturing Science and Engineering Conference*, vol. 50749. American Society of Mechanical Engineers, 2017, p. V003T04A006.
- [24] M. S. Islam, M. S. Rakha, W. Pourmajidi, J. Sivaloganathan, J. Steinbacher, and A. Miranskyj, "Anomaly detection in large-scale cloud systems: An industry case and dataset," *arXiv preprint arXiv:2411.09047*, 2024.
- [25] L. Bacchiani, G. De Palma, L. Sciuillo, M. Bravetti, M. Di Felice, M. Gabbriellini, G. Zavattaro, and R. Della Penna, "Low-latency anomaly detection on the edge-cloud continuum for industry 4.0 applications: The seawall case study," *IEEE Internet of Things Magazine*, vol. 5, no. 3, pp. 32–37, 2022.
- [26] X. Yu, X. Yang, Q. Tan, C. Shan, and Z. Lv, "An edge computing based anomaly detection method in iot industrial sustainability," *Applied Soft Computing*, vol. 128, p. 109486, 2022.
- [27] T. T. Huong, T. P. Bac, D. M. Long, T. D. Luong, N. M. Dan, B. D. Thang, K. P. Tran *et al.*, "Detecting cyberattacks using anomaly detection in industrial control systems: A federated learning approach," *Computers in Industry*, vol. 132, p. 103509, 2021.
- [28] L. Dutta and S. Bharali, "Tinyml meets iot: A comprehensive survey," *Internet of Things*, vol. 16, p. 100461, 2021.
- [29] P. P. Ray, "A review on tinyml: State-of-the-art and prospects," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 4, pp. 1595–1623, 2022.
- [30] H. Doyu, R. Morabito, J. Höller *et al.*, "Bringing machine learning to the deepest iot edge with tinyml as-a-service," *IEEE IoT Newsl*, vol. 11, pp. 1–3, 2020.
- [31] M. Antonini, M. Pincheira, M. Vecchio, and F. Antonelli, "An adaptable and unsupervised tinyml anomaly detection system for extreme industrial environments," *Sensors*, vol. 23, no. 4, p. 2344, 2023.
- [32] E. Njor, J. Madsen, and X. Fafoutis, "A primer for tinyml predictive maintenance: Input and model optimisation," in *Artificial Intelligence Applications and Innovations: 18th IFIP WG 12.5 International Conference, AIAI 2022, Hersonissos, Crete, Greece, June 17–20, 2022, Proceedings, Part II*. Springer, 2022, pp. 67–78.
- [33] H. Ren, D. Anicic, and T. A. Runkler, "Tinyol: Tinyml with online-learning on microcontrollers," in *2021 international joint conference on neural networks (IJCNN)*. IEEE, 2021, pp. 1–8.
- [34] P. Andrade, I. Silva, G. Signoretti, M. Silva, J. Dias, L. Marques, and D. G. Costa, "An unsupervised tinyml approach applied for pavement anomalies detection under the internet of intelligent vehicles," in *2021 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0&IoT)*. IEEE, 2021, pp. 642–647.
- [35] B. Fyntanidou, M. Zouka, A. Apostolopoulou, P. D. Bamidis, A. Billis, K. Mitsopoulos, P. Angelidis, and A. Fourlis, "Iot-based smart triage of covid-19 suspicious cases in the emergency department," in *2020 IEEE Global Communications Workshops (GC Wkshps)*, 2020, pp. 1–6.
- [36] P. Mohan, A. J. Paul, and A. Chirania, "A tiny cnn architecture for medical face mask detection for resource-constrained endpoints," in *Innovations in Electrical and Electronic Engineering: Proceedings of ICEEE 2021*. Springer, 2021, pp. 657–670.
- [37] I. Fedorov, M. Stamenovic, C. Jensen, L.-C. Yang, A. Mandell, Y. Gan, M. Mattina, and P. N. Whatmough, "Tinylstms: Efficient neural speech enhancement for hearing aids," *arXiv preprint arXiv:2005.11138*, 2020.
- [38] B. Coffen and M. S. Mahmud, "Tinydl: edge computing and deep learning based real-time hand gesture recognition using wearable sensor," in *2020 IEEE International Conference on E-health Networking, Application & Services (HEALTHCOM)*. IEEE, 2021, pp. 1–6.
- [39] A. N. Roshan, B. Gokulapriyan, C. Siddarth, and P. Kokil, "Adaptive traffic control with tinyml," in *2021 Sixth International Conference on Wireless Communications, Signal Processing and Networking (WiSP-NET)*. IEEE, 2021, pp. 451–455.
- [40] W. Raza, A. Osman, F. Ferrini, and F. D. Natale, "Energy-efficient inference on the edge exploiting tinyml capabilities for uavs," *Drones*, vol. 5, no. 4, p. 127, 2021.
- [41] S. G. Patil, D. K. Dennis, C. Pabbaraju, N. Shaheer, H. V. Simhadri, V. Seshadri, M. Varma, and P. Jain, "Gesturepod: Enabling on-device gesture-based interaction for white cane users," in *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 403–415.
- [42] C. Nicolas, B. Naila, and R.-C. Amar, "Tinyml smart sensor for energy saving in internet of things precision agriculture platform," in *2022 Thirteenth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, 2022, pp. 256–259.
- [43] P. Andrade, I. Silva, M. Silva, T. Flores, J. Cassiano, and D. G. Costa, "A tinyml soft-sensor approach for low-cost detection and monitoring of vehicular emissions," *Sensors*, vol. 22, no. 10, p. 3838, 2022.
- [44] H. Han and J. Siebert, "Tinyml: A systematic review and synthesis of existing research," in *2022 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*. IEEE, 2022, pp. 269–274.
- [45] J. Kim, H. Kang, and P. Kang, "Time-series anomaly detection with stacked transformer representations and 1d convolutional network," *Engineering Applications of Artificial Intelligence*, vol. 120, p. 105964, 2023.
- [46] J. Yu, Y. Zheng, X. Wang, W. Li, Y. Wu, R. Zhao, and L. Wu, "Fastflow: Unsupervised anomaly detection and localization via 2d normalizing flows," *arXiv preprint arXiv:2111.07677*, 2021.
- [47] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.
- [48] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.
- [49] L. Eren, T. Ince, and S. Kiranyaz, "A generic intelligent bearing fault diagnosis system using compact adaptive 1d cnn classifier," *Journal of Signal Processing Systems*, vol. 91, pp. 179–189, 2019.
- [50] H. Cho and S. M. Yoon, "Divide and conquer-based 1d cnn human activity recognition using test data sharpening," *Sensors*, vol. 18, no. 4, p. 1055, 2018.
- [51] J. Zhao, X. Mao, and L. Chen, "Speech emotion recognition using deep 1d & 2d cnn lstm networks," *Biomedical signal processing and control*, vol. 47, pp. 312–323, 2019.
- [52] X. Xie, B. Wang, T. Wan, and W. Tang, "Multivariate abnormal detection for industrial control systems using 1d cnn and gru," *IEEE Access*, vol. 8, pp. 88 348–88 359, 2020.
- [53] A. Nanduri and L. Sherry, "Anomaly detection in aircraft data using recurrent neural networks (rnn)," in *2016 Integrated Communications Navigation and Surveillance (ICNS)*. Ieee, 2016, pp. 5C2–1.

- [54] P. J. Freire, S. Srivallapanondh, A. Napoli, J. E. Prilepsky, and S. K. Turitsyn, "Computational complexity evaluation of neural network applications in signal processing," *arXiv preprint arXiv:2206.12191*, 2022.
- [55] D. Baylor, E. Breck, H.-T. Cheng, N. Fiedel, C. Y. Foo, Z. Haque, S. Haykal, M. Ispir, V. Jain, L. Koc *et al.*, "Tfx: A tensorflow-based production-scale machine learning platform," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1387–1395.
- [56] E. Bisong and E. Bisong, "Kubeflow and kubeflow pipelines," *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, pp. 671–685, 2019.
- [57] S. Hymel, C. Banbury, D. Situnayake, A. Elium, C. Ward, M. Kelcey, M. Baaijens, M. Majchrzycki, J. Plunkett, D. Tischler *et al.*, "Edge impulse: An mlops platform for tiny machine learning," *arXiv preprint arXiv:2212.03332*, 2022.
- [58] STMicroelectronics, "Stmicroelectronics nanoedge ai," <https://stm32ai.st.com/nanoedge-ai/>, Accessed May 1, 2023.
- [59] E. W. Weisstein, "Euler angles," <https://mathworld.wolfram.com/>, 2009.
- [60] H. Kayan, R. Heartfield, O. Rana, P. Burnap, and C. Perera, "CASPER: Context-Aware IoT Anomaly Detection System for Industrial Robotic Arms," Cardiff University, Technical Report, 2023, available online at ORCA. [Online]. Available: <https://orca.cardiff.ac.uk/id/eprint/165098>
- [61] H. Kayan, O. Rana, P. Burnap, and C. Perera, "Casper: Context-aware anomaly detection system for industrial robotic arms," in *2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, 2023, pp. 282–284.
- [62] R. Mahony, T. Hamel, and J.-M. Pfimlin, "Nonlinear complementary filters on the special orthogonal group," *IEEE Transactions on automatic control*, vol. 53, no. 5, pp. 1203–1218, 2008.
- [63] Netron, "Netron," <https://netron.app/>, Accessed: 31-January-2024.
- [64] J. Ortiz Laguna, A. G. Olaya, and D. Borrajo, "A dynamic sliding window approach for activity recognition," in *User Modeling, Adaption and Personalization: 19th Int'l Conf., UMAP 2011, Girona, Spain, July 11-15, 2011. Proceedings 19*. Springer, 2011, pp. 219–230.



**Omer Rana** is Professor of Performance Engineering and the Dean of International for Physical Sciences and Engineering at Cardiff University. He holds a PhD in Neural Computing and Parallel Architectures from Imperial College (University of London, UK). His research focus is distributed systems (cloud and edge computing), cybersecurity and machine learning.



**Pete Burnap** is the Director of the Cardiff Centre for Cyber Security Research, and Airbus's only global Centre of Excellence in Cyber Security Analytics. He was appointed to the UK Government's AI Council, advising on AI and the data economy. He is Co-Director of Cardiff University's flagship Digital Transformation Innovation Institute. His research covers the development of artificial intelligence approaches to automated cyber attack detection and response, and the fusion with risk and human factors of cybersecurity. He has been involved in research grants worth in excess of £23m, leading large awards from EPSRC, ESRC and industry – publishing dozens of articles in top journals and conferences, as well as translating research into commercial products. Professor Burnap also directs the Wales Cyber Innovation Hub – a £13.8m programme aiming to grow the Wales cybersecurity sector by 50% over the next 5 years through innovations in IP commercialisation, and to develop a world class suite of hands-on cybersecurity skills programmes aligned to local industry needs – reskilling over 1500 people in South East Wales.



**Hakan Kayan** received the B.Sc. degree (Hons.) in Electrical and Electronics from Izmir University of Economics, Izmir, Turkey, in 2017, and the M.Sc. degree in Information Security from the University of Surrey, Surrey, UK, in 2019. He is currently pursuing the Ph.D. degree at Cardiff University. He is currently a Research Assistant with Cardiff University, Cardiff. His research interests include anomaly detection, time-series analysis, predictive maintenance, edge computing, and AI/ML for IoT.



**Ryan Heartfield** received the Ph.D. degree in AI and Cyber security from the University of Greenwich in 2017. He is currently CEO of Industrial AI startup Exalens and a visiting professor of AI in Cybersecurity at Kingston University. He has published over 20 articles in top academic journals in AI, Cyber-Physical Systems and Cybersecurity. He has led and co-led U.K. and European research projects on the security of autonomous vehicles, measuring the trustworthiness of human sensor platforms, cyber-physical systems anomaly detection,

and studying cyber threats and the emotional impact of security breaches in smart home environments.



**Charith Perera** received the B.Sc. (Hons.) degree in computer science from Staffordshire University, Stoke-on-Trent, U.K., the M.B.A. degree in business administration from the University of Wales, Cardiff, U.K., and the Ph.D. degree in computer science from Australian National University, Canberra, ACT, Australia, in 2015. He is currently a Reader at Cardiff University, Cardiff. Previously, he was with the Information Engineering Laboratory, ICT Centre, CSIRO, Canberra.