Anatomia del presente: riflessioni sulla 24ª Triennale di Milano

VARADE++: An Edge-Friendly Framework for Real-Time Anomaly Detection in Production Plants

Alessio Mascolini, Sebastiano Gaiardelli, *Member, IEEE*, Francesco Ponzio, *Member, IEEE*, Nicola Dall'Ora, *Member, IEEE*, Stefano Quer, *Senior Member, IEEE*, Franco Fummi, *Senior Member, IEEE*, Sara Vinco, *Senior Member, IEEE*, and Santa Di Cataldo, *Member, IEEE*

***Abstract*—Real-time anomaly detection is pivotal to the success of smart robotics, particularly in production plants, where even minor system failures can result in significant machine downtime and costly process disruptions. To address this, a specialized Machine Learning (ML) model must be seamlessly integrated into a network of interconnected machinery, sensors, and actuators, all processing vast streams of multidimensional sensor data with minimal latency that cloud-based solutions often struggle to achieve. In this work, we introduce VARADE++, an edge-optimized anomaly detection framework designed to navigate the complex trade-offs between anomaly detection accuracy, inference speed, and computational efficiency. By leveraging a lightweight auto-regressive architecture rooted in attentionless transformers, paired with a variational training paradigm, we achieve real-time processing capabilities. Our model is integrated into an advanced IoT infrastructure, enabling low-latency handling of intricate data streams. The effectiveness of VARADE++ is demonstrated across two public benchmarks and validated through a real-world case study within a sensorized industrial pilot production line, with an industrial robot as the primary focus. Our results not only highlight the superior anomaly detection capabilities of VARADE++ but also showcase its operational efficiency in a real-time edge computing environment, outperforming state-of-the-art solutions in the balance between detection performance and practical deployability.**

***Index Terms*—Anomaly detection, process monitoring, time series analysis, machine learning, industrial IoT, smart robotics.**

I. Introduction

UNEXPECTED machine failures in a production plant can lead to significant downtime and disrupt the entire process, resulting in increased production times and economic losses. To solve this issue, industries are investing heavily in upgrading their existing systems with process monitoring and anomaly detection technologies to identify and possibly solve any minor problems before they escalate into process failures. The sooner the anomaly is identified, the better the chance that it can be solved appropriately.

Real-time Anomaly Detection (AD) is enabled by the continuous collection of production data from many heterogeneous sensors. This translates into the generation of a massive stream of Multivariate Time Series (MTS), where a large number of variables (i.e., channels) are recorded over time, each with its scale and periodicity. The analysis of these MTS is tackled by specialized ML models, designed to i) learn the complex distribution of the production data in "normal" conditions and ii) identify, at run-time, any significant deviations from the learned distribution.

This scenario is ultimately made possible by the Industrial Internet of Things (IIoT), which integrates machinery, sensors, and actuators of a production plant into an interconnected network, where they can communicate and exchange both multidimensional data and commands in real time. This effectively transforms the production devices into Cyber-Physical System (CPS), where physical and software components are deeply intertwined.

Despite the recent advancements in all the technologies involved, the effective deployment of ML-based anomaly detection in a production plant is still hard to achieve due to several problems. On the one hand, detecting unknown anomalies from MTS is a non-trivial task that typically requires very complex

Received 29 September 2024; revised 3 November 2025; accepted 17 August 2026. Date of publication 27 August 2026; date of current version 8 September 2026. The work was supported by PRIN 2022T7YSHJ SMART-IC - Next Generation EU Project. *(Corresponding author: Nicola Dall'Ora.)*

Alessio Mascolini, Francesco Ponzio, Stefano Quer, Sara Vinco, and Santa Di Cataldo are with the Department of Control and Computer Engineering, Politecnico di Torino, 10129 Torino, Italy (e-mail: alessio.mascolini@polito.it; francesco.ponzio@polito.it; stefano.quer@polito.it; sara.vinco@polito.it; santa.dicataldo@polito.it).

Sebastiano Gaiardelli is with the Department of Engineering for Innovation Medicine, University of Verona, 37134 Verona, Italy, and also with the Institute for Advanced Study, Technical University of Munich, 85748 Munich, Germany (e-mail: sebastiano.gaiardelli@tum.de).

Nicola Dall'Ora is with the Department of Engineering for Innovation Medicine, University of Verona, 37134 Verona, Italy, and also with the Department of Engineering Sciences, University Guglielmo Marconi, 00193 Rome, Italy (e-mail: n.dallora@unimarconi.it).

Franco Fummi is with the Department of Engineering for Innovation Medicine, University of Verona, 37134 Verona, Italy (e-mail: franco.fummi@polito.it).

Digital Object Identifier 10.1109/TETC.2026.3726298

ML models. On the other hand, such models can be heavy on the computational side, which is incompatible with the resources available at the edge. For this reason, typical solutions rely on streaming the data to a cloud platform, where even the heaviest models can be executed with no issue [1]. This is not ideal, especially in large production plants, where the transmitted data streams can reach the order of GB/s, often resulting in communication overhead and hence latency problems [2].

Based on the above considerations, an optimal solution needs to strike a very complicated balance between anomaly detection performance, inference speed and low computational requirements, compatible with edge computing scenarios. Motivated by this aim, in our previous work we proposed *VARADE (Variational-based AutoRegressive model for Anomaly Detection on the Edge)*, and demonstrated that a light autoregressive framework based on variational inference is best suited for real-time execution on the edge [3]. This paper presents a new and improved anomaly detection system, VARADE++. The significant contributions of our current work are the following.

- While maintaining the original autoregressive implant, the architecture of VARADE++ is deeply modified to enhance its real-time performance and overall efficiency for edge deployment. More specifically, we substitute convolutions with attentionless transformers and introduce a completely new variational training paradigm based on the forecasting error and the Mean Squared Error (MSE).
- We focus not only on the ML-model, but on the overall IIoT ecosystem where the model is deployed. For this purpose, we present in detail a cutting-edge Internet of Things (IoT) infrastructure that allows for the model to interact with the production plant effectively. Preliminary versions of this infrastructure, in different contexts from the one of this paper, were described in [4], [5]. In the current work, we adapt the IoT components to the real-time anomaly detection task targeted by VARADE++.
- We present a very extensive experimental validation of our system. First, using two well-established public benchmarks, we position VARADE++ against the most popular state-of-the-art solutions for MTS anomaly detection. Then, we focus on the specific scenario targeted by VARADE++, and build a realistic case study within a pilot production line, with a sensorized industrial robot as the main target. In this case study, we compare VARADE++ and the counterpart solutions by leveraging the same IoT infrastructure and computational resources for all the models. The comparison considers not only the models' accuracy but also their efficient deployment in a real-time anomaly detection scenario. Specifically, we analyze trade-offs across a comprehensive set of metrics, including model size, memory footprint, and energy consumption, on different computing platforms. While the implementation of correction strategies for the detected anomalies is not within the specific scope of this paper, we provide insights on the feasibility of the proposed anomaly detector as a starting point for closed-loop feedback.

The rest of this paper is structured as follows. In Section II, we introduce the background of our work and discuss the related literature. In Section III we present in detail VARADE++ (in Section III-A, the ML model and in Section III-B the IoT infrastructure, respectively). In Section IV, we present and discuss our experimental results. Finally, in Section V, we draw our conclusions and present future works.

II. Background and Related Works

As introduced in Section I, real-time anomaly detection in a production plant relies on two main components. First, an ML-based *anomaly detection model* that can identify anomalous data points from a continuous input data stream. To ensure real-time processing, the model must produce its output with minimal delay. Second, an *IoT infrastructure* able to continuously feed the ML model with input data from the plant, receive the anomaly detection output and put the appropriate feedback strategies into action if required. In the following, we better analyze these two components.

A. ML-Based Anomaly Detection

Like in most applications dealing with complex multi-dimensional data, state-of-the-art MTS anomaly detection has recently been dominated by Deep Learning (DL). As the specific nature and characteristics of the anomalies may not be known a priori, a deep neural network is trained on "normal" data to identify, at inference time, any variations from the expected normal behavior. Based on their strategy, we can identify three main families.

Forecasting-based techniques are designed to predict one or more future time steps based on a sliding window of past observations. During inference, the predicted values are compared against the actual observations once they become available. An *anomaly scoring function*, often based on metrics such as mean squared error or absolute deviation, is then used to quantify the discrepancy between the predicted and observed values. Time points exhibiting larger prediction errors are assigned higher anomaly scores, as such deviations suggest that the underlying temporal dynamics have been disrupted by anomalous behavior. The backbones of these models span from traditional recurrent architectures such as RNN and LSTM [6] to more sophisticated ones leveraging graph-based embeddings and attention mechanisms [7].

Reconstruction-based techniques are trained to perform a reconstruction task rather than a forecasting one. The primary objective of these models is to learn a compact latent representation that captures the underlying structure and regularities of normal time series data. During training, the model encodes input sequences into this latent space and then attempts to reconstruct the original inputs from the encoded representations. At inference time, the same process is applied to unseen data: the model reconstructs the input based on the learned latent features, and an *anomaly score* is computed by measuring the reconstruction error, typically using metrics such as mean squared error or reconstruction likelihood. A high reconstruction error indicates that the model struggles to reproduce the input accurately, implying that the data instance deviates from the distribution of normal patterns learned during training. These

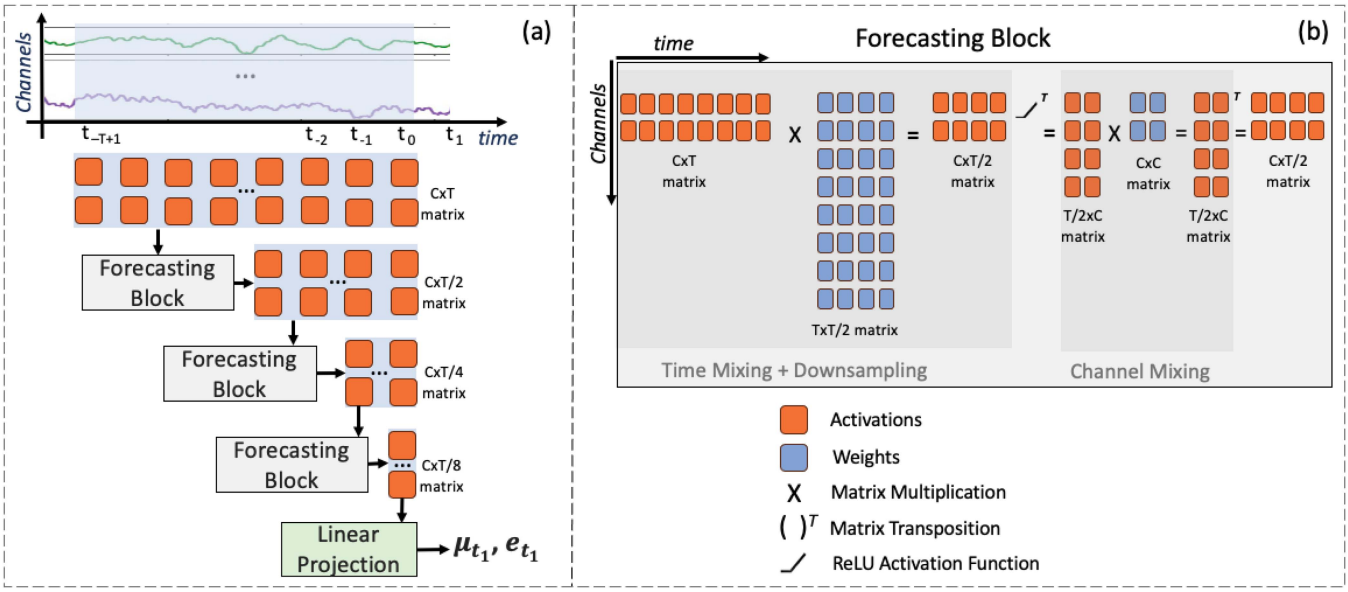

Figure 1. Backbone of the proposed AD model. (a) The Autoregressive architecture consists of a cascade of *forecasting blocks* and a final linear projection, predicting the mean value μ and the absolute forecasting error e on the next time-step t_1. (b) The details of the forecasting block. The input is represented as a 2×8 matrix only for the sake of readability.

TABLE I
TOP THREE TECHNIQUES PER CATEGORY IN SMD AND EXATHLON DATASETS

	SMD	Exathlon
Forecasting	LSTM-P [6] TCN-S2S-P [33] GDN [7]	GDN [7] DeepAnt [34] TCN-S2S-P [33]
Reconstruction	LSTM-AE [35] STGAT-MAD [37] Donut [39]	MSCRED [36] GenAD [38] STGAT-MAD [37]
Hybrid	MTAD-GAT [40] THOC [41] LSTM-AE OC-SVM [9]	MTAD-GAT [40] THOC [41] LSTM-AE OC-SVM [9]
Traditional	IsolationForest [11] kNN [10] GBRF [12]	IsolationForest [11] kNN [10] GBRF [12]

methods typically leverage generative models for time series, such as Autoencoders (AE), Variational Autoencoders (VAE), or Generative Adversarial Networks (GAN) [8].

Finally, *hybrid techniques* implement a mixture of different strategies or a completely new paradigm altogether. For example, by combining the reconstruction error of an LSTM-AE with outlier detection by One Class SVM [9].

Theoretically, DL models are best suited to handle complex patterns in MTS data. However, most are affected by two significant drawbacks. First, they require massive datasets to train. Second, they are computationally intensive. While model training can be carried out offline using high-performance computing resources, thereby mitigating its impact on practical deployment, the high computational demands during the inference phase pose a significant challenge for real-time applications on resource-constrained edge devices. Because of these limitations, several works still rely on classic non-DL methods. In this fourth family, hereby referred to as *traditional techniques*, we can include popular outlier detectors such as k-Nearest Neighbors (kNN) [10], Isolation Forest [11], and Gradient Boosted Regression Forest (GBRF) [12]. In these approaches, the anomalies are identified based on their higher distance from neighbors in the feature space, the number of binary splits of an ensemble of decision trees, or the residuals from an ensemble of weak predictions, respectively.

Despite the growing body of research, identifying a dominant model (or even a family of models) able to solve every problem in every scenario is impossible for many issues. First of all, with the variety of applications that fall under the definition of MTS anomaly detection, the model's architecture and hyperparameters are tailored to the specific data characteristics to be addressed. After that, there is no current agreement on a set of benchmarks or a common validation strategy. Recent surveys demonstrated severe flaws in some of the most used public datasets such as SWaT, WADI, SMAP, and MSL, as well as in the non-transparent evaluation strategies of most published literature, which are typically over-optimistic [8], [13]. Finally, while most works focus on the accuracy of anomaly detection, there is much lesser attention towards investigating the models in terms of their efficiency, flexibility, and scalability concerning the hardware where they are executed [14], [15]. These are key aspects for an effective real-time deployment in a CPS scenario.

B. IoT-Based Infrastructure

IIoT is a service-oriented industrial ecosystem using network interconnection of industrial resources, data, and system interoperability to enable flexible resource allocation, on-demand execution of processes, resource optimization, and rapid adaptation [16]. IIoT allows the collection of data from heterogeneous sources (e.g., sensors, IIoT-enabled equipment, management

software like Enterprise Resource Planning (ERP)), to integrate and fuse it and perform deep data analysis and mining to improve efficiency and optimization of the production process.

A typical IIoT architecture has three layers [17]: (i) the *physical layer*, including the production life cycle and the data sources (e.g., sensors, IIoT-enabled machinery), (ii) the *communication layer*, that collects and fuses such generated data, and (iii) the *application layer*, that performs data-aware monitoring, forecast, control, and optimization, leading to efficient industrial operation.

IIoT has some peculiar characteristics that go beyond being the application of IoT to an industrial scenario:

- *Interoperability:* IIoT applications must integrate with a wide range of Operational Technologies (OT) assets and devices, legacy enterprise IT solutions, and communication protocols. This requires a powerful infrastructure that is able to handle data from heterogeneous sources.
- *Adaptability:* industrial contexts require programming and reconfiguration with reliable flexibility and adaptability.
- *Timeliness:* quality must be assured in terms of accuracy and responsiveness, as deviations or lag can result in poor efficiency or downtime, which is a considerable amount of lost revenue. This implies that IIoT applications have an especially low tolerance to delay.

The low tolerance to delays, especially for time-sensitive tasks such as emergency shutdowns, makes the edge a preferred direction for analytics computations. In IoT environments, the cloud is often used to offload all data and perform analytics on powerful hardware. However, a typical smart factory implies large amounts of sensory data (in the order of GB per second) that would make a cloud architecture prohibitive in terms of bandwidth requirement, operation costs, and overall processing delay (i.e., data transmission to the cloud for processing and back could even cause severe consequences) [2]. For these reasons, it becomes very convenient to move data collection and processing *closer and closer to the edge*. This comes, however, at a cost, as despite the continuous improvement of computing resources and performance of edge devices, the edge still imposes strict resource requirements. It is thus essential to identify the proper algorithm (especially in the case of ML) by taking into account the distinctive characteristics of edge computing for IIoT [17].

III. Proposed Framework

A. Anomaly Detection Model

To best fit the stringent requirements of a generic industrial plant, our AD model is designed so as to balance detection performance, real-time response, scalability to high-dimensional data, and capability to adapt to limited computational resources at the edge.

Starting from these considerations, we chose a *forecasting-based autoregressive* backbone, where a single future time step is predicted based on a limited window of past samples. With this approach, anomalies can be identified with a simple *anomaly scoring* function, that highlights the instances with a significant prediction error. This ensures a good compromise between reconstruction-based approaches (which are typically very resource-demanding) and traditional outlier detectors (that offer low latency but may have limited capability of learning complex patterns).

A detailed scheme of our proposed architecture is shown in Figure 1. In the following, we explain our design choices in detail.

1) Autoregressive Forecasting: Anomaly detection using autoregressive time series forecasting involves predicting future values of a time series based on its past values and detecting deviations from expected patterns. To do so, a neural network is trained on historical data to understand and predict the time series' normal behavior. The model learns the dependencies and patterns over time, enabling it to forecast the next values in the series. Once the model is deployed, it continues to predict future values based on an input that consists of a sliding window of the latest past data.

The autoregressive nature of our model is well visible in Figure 1(a). The sliding time window is shown in blue, and we represent the input of the neural network as a $C \times T$ matrix, where C is the number of channels of the signal and T is the width of the time window. To obtain a prediction of the future time step (t_1, in the figure), the input matrix is fed to a cascade of *forecasting blocks* implementing a modified transformer architecture (details will follow) and a final linear projection layer. This is a significant architectural difference compared to our previous work, where the autoregressive model was implemented as a sequence of convolutional layers [3].

As with any forecasting technique, the choice of T involves a careful trade-off between detection capability and computational cost, and must be determined on a case-by-case basis. A larger T enables the model to capture longer-term temporal dependencies and context, which can be crucial for detecting complex, slow-developing anomalies. Conversely, a smaller T reduces memory usage and computational load, resulting in faster inference times. A practical strategy is to select the largest T that still allows to satisfy the system's real-time processing requirements, as defined later in this section.

2) Architectural Details: Time series forecasting architectures tend to use either fully connected, recurrent, convolutional, or transformer layers, each of them with its own set of strengths and downsides:

- *Fully Connected* networks require a very large number of parameters and can be inefficient.
- *Recurrent* networks are intrinsically sequential and, therefore, poorly parallelizable.
- *Convolutions* are parallelizable yet require translational invariance.
- *Transformers* attempt to provide a fully parallelizable architecture without the assumption of translational invariance, by employing the *attention* mechanism [18].

Let us consider the 2D data space (C, T). A fully connected (FC) layer operating on such data introduces $(T \cdot C)^2$ learnable parameters. This makes high-dimensional spaces intractable. To address this problem, transformers for time series forecasting process the data first in the time dimension (*time mixing*), and then in the channel dimension (*channel mixing*). A self-attention mechanism primarily handles the time mixing phase, which

allows mixing information from various time steps by focusing on the most relevant ones. This allows the model to capture temporal dependencies, even distant ones, by considering all time steps simultaneously rather than sequentially. The channel mixing phase involves processing and combining information across different channels or features at each time step and is typically handled by a FC layer [19].

Despite its many advantages, the conventional attention mechanism still has a complexity that is quadratic to the time dimension [18] and empirically requires very large datasets to converge due to the lower amount of inductive bias. While recent transformer-based methods may offer reduced attention time and memory complexity, they typically implement many additional operations beyond attention, with a heavy impact on the actual running time [20].

Further works have independently shown that transformers can be effective even without attention, as the time mixing can be performed by much simpler mechanisms without loss of accuracy [19], [20], [21], [22]. Building on these findings, we implement the *forecasting block* of our model as an attentionless transformer, where an FC layer is used for both the time mixing and the channel mixing phases. Nonetheless, unlike other architectures in this category, our design deliberately omits residual connections. While residual connections are well known to improve optimization stability, prior studies have shown that they are less memory-efficient during inference, which makes them less suitable for deployment on memory-bandwidth-constrained hardware [23], [24].

The resulting design of our *forecasting block* is shown in Figure 1(b). By using FC layers in both time and channel mixing phases, we reduce the operations to a simple and optimized sequence of two matrix multiplications and two matrix transpositions, with ReLU as the non-linear activation function. More specifically, we use a $(T \times T/2)$ weight matrix for time mixing and a $(C \times C)$ weight matrix for channel mixing to achieve downsampling along the time dimension. This limits the maximum number of learnable parameters to $T^2/4 + C^2$ and makes the architecture scalable to high-dimensional data.

When combining the blocks together, the time dimension halves at each forecasting step until it reaches 1 (see Figure 1(a)). Then, the final output of the network is obtained by a simple linear projection. Ultimately, this design choice has two major advantages:

- *Reduced computational load and parameter count:* by progressively shrinking the time dimension, subsequent layers require fewer computations and parameters, making the architecture scalable to longer initial time windows without a prohibitive increase in complexity.
- *Reduced memory requirements and faster inference:* the reduced activation size decreases memory usage and bandwidth demands, achieving faster inference speeds.

3) Variational Inference and Loss Function: Conventional forecasting techniques directly predict the future data values of a time series and rely on the squared difference of the predicted and the observed data to identify the anomalies. In our previous work, we pointed out that this approach is not ideal in edge applications, as it typically introduces a bad trade-off between computational efficiency and the accuracy of the model [3]. To address this issue, we employ a *variational framework:* instead of predicting a point estimate for the next time step t_1, we predict a probability distribution $P(t_1)$. By assuming a Gaussian distribution, this can be reduced to computing the mean and/or variance of $P(t_1)$.

In a Gaussian distribution, the variance is also a measure of the confidence of the prediction model, which we can reasonably expect to be lower in case of anomalous behavior. Stemming from this consideration, in our previous work, we used the predicted variance σ^2 as the anomaly score [3]. Since we do not have ground truth values for the variance to be used during the training phase, but only the observed values, we trained our model by using the negative log-likelihood (NLL) of the measured sample y belonging to the predicted distribution as the loss function.

In our current implementation, we follow a completely different approach. We train our neural network so as to directly predict μ_{t_1} (mean value of $P(t_1)$) and e_{t_1} (absolute forecasting error on the time-step t_1). To do so, we use the mean squared difference (MSE) between the predicted and the observed e as the loss function, which we found leads to better training behavior and forecasting performance compared to NLL. Interestingly, this finding is consistent with the literature on *diffusion denoising probabilistic models*, a class of generative architectures that have been successfully applied to a diverse range of problems. In many recent works, MSE was empirically found to provide better results than the variational lower bound loss [25].

At inference time, the *anomaly score* $S(y)$ for a data sample y is computed as follows:

$$S(y) = \frac{y - \mu}{e}, \tag{1}$$

where μ represents the expected (predicted) value of y, and e is a normalization factor. An anomaly is detected by setting a threshold τ on the anomaly score: if $S(y) \geq \tau$, the corresponding data point y is classified as anomalous.

The rationale behind Eq. (1) is as follows. The numerator quantifies the deviation between the observed data point y and its expected value μ. Under normal operating conditions, this deviation is expected to remain small, while larger deviations indicate potential anomalies. The denominator e serves as a normalization term that accounts for the variability or uncertainty of the signal. In practice, a higher e reflects greater experimental noise or intrinsic variability in the time series, which naturally reduces the model's predictive confidence. Consequently, in such cases, a larger deviation (numerator) is required for a data point to be deemed anomalous for a given threshold τ. Conversely, when the signal exhibits low variability (i.e., a smaller e), even minor deviations from the expected value may be sufficient to signal anomalous behavior.

B. IoT Infrastructure

To effectively support the AD algorithm, the IoT infrastructure is subject to the following constraints:

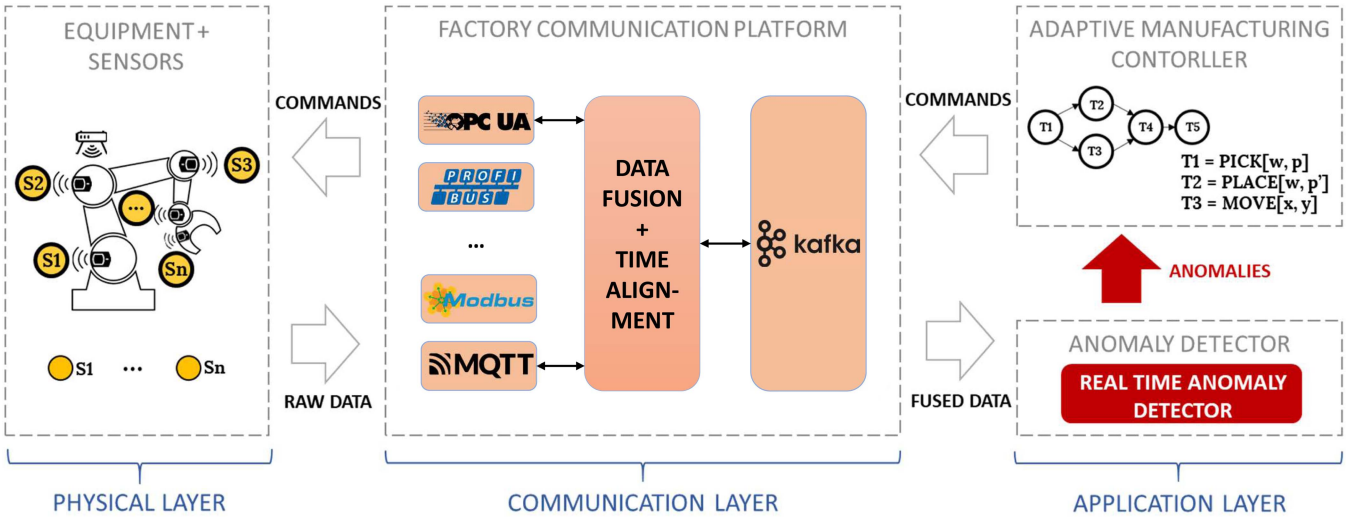

Figure 2. Schematic representation of the IIoT infrastructure of VARADE++, with the real-time anomaly detection algorithm (in red) operating on an edge node.

- *Integration of data generated by different sources*, e.g., retrieved from internal machinery variables, sensors collecting extra-functional properties, and sensors collecting environmental conditions of the plant.
- *Real-time data processing*, for guaranteeing the minimum latency time between the source of the data and the AD output.
- *Scalability of the proposed AD method*, to ensure that it can run on a large class of devices, from industrial PCs to low-power edge systems.
- *Support for a closed feedback loop* between the AD response and the industrial plant, to allow a prompt reaction to unforeseen events highlighted by the AD.

To achieve this purpose, the proposed infrastructure follows the typical organization of IIoT architectures applied to anomaly detection on MTS in industrial contexts [26]. Figure 2 outlines its organization:

- The physical layer includes sensors and IIoT-enabled machinery that produce raw data about equipment operation.
- The communication layer is implemented as a platform that integrates data through protocol management, fusion, and temporal alignment to produce coherent data.
- The application layer includes an Adaptive Manufacturing Controller (AMC) that schedules the production tasks and sends the commands to the production plant, as well as the ML model (see Section III-A) that detects the anomalies in real-time and notifies them back to the AMC.

Like any IIoT solution, this infrastructure is extremely modular, as it can be adapted to different contexts by changing the ML algorithm or extended with additional functionalities. A preliminary version has been presented in [4], but at that time, we had not finalized the real-time AD support, and we did not include a discussion on the feedback loop with the real-plant. In the following, we go more in-depth into the role and implementation of the different layers.

1) Physical Layer: The physical layer includes any data sources that collect data from the environment and the production machinery (left of Figure 2). In modern production plants, the equipment are composed by several machinery that can be interconnected through different industrial communication protocols (wired or wireless), and by IIoT sensors that monitor the environment plus non IIoT-enabled equipment [27]. In an industrial context, another important factor is the frequency at which raw data are collected by the IIoT sensors, which can be different for each sensor and can be influenced by the machinery's working conditions. Thus, each data must be collected at one particular frequency based on its specific physical nature. This translates to capturing the machinery conditions within a particular time window that may vary from milliseconds to seconds.

2) Communication Layer: The data generated by the physical layer is extremely complex due to its heterogeneity in terms of protocols and the difference in timescales. Because of this complexity, it can not be used in its raw version. Still, it rather must be fused and aligned, to guarantee that it can be fed into an ML algorithm or processed by an analytic engine to obtain meaningful information. For this purpose, all data is sent to the Factory Communication Platform (FCP), positioned in the middle of the proposed infrastructure, between all other building blocks, and responsible for data management and storage.

The first challenge of the FCP is the heterogeneity in terms of protocols. This is managed with some *IIoT brokers*, intermediary entities that allow the management of the flow of information effectively. Each data source communicates with a broker of the same protocol, e.g., an OPC Unified Architecture (OPCUA) enabled machine will send data to the OPCUA broker. At the same time, smart sensors will forward the data to a corresponding MQTT broker. Such brokers communicate correctly with their respective data sources, and send the data to a *centralized Kafka broker*, aggregating the information received from different sensors into a single repository.

The subsequent steps are data fusion and temporal alignment, which aim to comprehensively integrate data extracted from multiple sources. *Data fusion* implies merging heterogeneous data formats by preserving only measurement data and removing any protocol-specific information. *Time alignment* tackles the

TABLE II
COMPARISON BETWEEN THE ANOMALY DETECTION MODELS EXECUTED IN REAL-TIME ON THREE EMBEDDED BOARDS AND ON A DESKTOP PC WITHOUT A DEDICATED GPU

Board Model	Anomaly Detection Model	CPU Usage (%)	GPU Usage (%)	AUPRC (%)	Inference Frequency (Hz)	RAM Usage (MB)	GPU RAM Usage (MB)	Energy Consumption (mWh)
Raspberry Pi 5	VARADE++	94.710	.	42.60	626.737	459.896	.	0.00027
	LSTM-P	96.070	.	27.70	762.683	454.957	.	0.00023
	GDN	95.688	.	28.20	147.854	461.104	.	0.00116
	TCN-S2S-P	96.033	.	31.70	376.912	456.586	.	0.00046
	DeepAnt	94.700	.	35.40	878.376	454.401	.	0.00019
	LSTM-AE	96.148	.	30.30	246.650	454.963	.	0.00070
	STGAT-MAD	95.330	.	34.70	7.564	478.389	.	0.02264
	Donut	93.178	.	29.00	60.255	468.588	.	0.00281
	GenAD	95.735	.	17.50	43.771	457.621	.	0.00392
	LSTM-AE-OC-SVM	95.983	.	25.80	43.848	454.934	.	0.00392
	MTAD-GAT	94.665	.	28.00	6.368	490.198	.	0.02679
	THOC	95.200	.	35.40	37.198	463.040	.	0.00460
	GBRF	25.013	.	36.40	127.192	449.370	.	0.00078
	Isolation Forest	24.988	.	24.90	292.130	447.481	.	0.00034
	kNN	24.915	.	43.00	87.258	451.400	.	0.00114
Jetson Xavier NX	VARADE++	56.420	16.430	42.60	146.701	5,941.250	965.837	0.01135
	LSTM-P	56.680	17.190	27.70	374.594	6,076.270	1,068.040	0.00447
	GDN	48.740	56.600	28.20	57.006	5,923.790	1,068.040	0.02940
	TCN-S2S-P	18.070	37.800	31.70	194.893	6,225.480	1,111.120	0.00909
	DeepAnt	56.690	19.490	35.40	222.270	6,217.540	1,118.450	0.00746
	LSTM-AE	63.720	25.940	30.30	245.894	5,860.390	1,082.160	0.00784
	STGAT-MAD	52.340	44.760	34.70	13.132	6,357.310	1,294.38	0.14787
	Donut	57.230	19.670	29.00	272.403	5,895.440	980.012	0.00680
	GenAD	55.980	19.940	17.50	66.323	5,763.200	916.424	0.02636
	LSTM-AE-OC-SVM	53.730	33.400	25.80	92.411	6,008.780	1,027.430	0.01885
	MTAD-GAT	54.780	71.050	28.00	64.944	6,310.010	1,218.310	0.05286
	THOC	55.650	97.120	35.40	69.174	6,040.090	1,052.970	0.04920
	GBRF	52.350	.	36.40	38.059	5,027.210	338.297	0.04218
	Isolation Forest	51.800	.	24.90	6.001	5,022.290	338.297	0.26187
	kNN	52.330	.	43.00	24.084	5,004.400	338.297	0.06503
Jetson Orin AGX	VARADE++	11.430	17.330	42.60	315.816	4732.360	478.930	0.00861
	LSTM-P	14.580	14.210	27.70	894.394	4423.790	255.369	0.00321
	GDN	34.330	36.880	28.20	46.767	5112.530	250.166	0.06143
	TCN-S2S-P	12.680	43.480	31.70	459.356	4553.610	233.188	0.00612
	DeepAnt	9.930	16.540	35.40	840.797	4490.900	241.544	0.00299
	LSTM-AE	33.550	30.180	30.30	536.145	4452.830	268.540	0.00566
	STGAT-MAD	19.820	58.900	34.70	22.101	4774.590	288.196	0.13714
	Donut	10.710	21.200	29.00	653.727	4470.920	251.321	0.00454
	GenAD	10.320	25.120	17.50	130.917	4676.220	230.578	0.02081
	LSTM-AE-OC-SVM	10.800	52.770	25.80	182.454	4478.990	236.271	0.01540
	MTAD-GAT	10.710	81.680	28.00	123.198	4645.290	267.745	0.03864
	THOC	10.990	75.430	35.40	146.567	4568.310	264.915	0.02656
	GBRF	10.890	.	36.40	74.299	4599.120	102.584	0.03480
	Isolation Forest	10.340	.	24.90	189.828	4605.800	103.831	0.01347
	kNN	10.390	.	43.00	43.872	4592.620	102.696	0.05760
i7 8700	VARADE++	98.070	.	42.60	885.521	429.200	.	1,430.67000
	LSTM-P	97.480	.	27.70	994.955	435.615	.	1,210.73000
	GDN	99.500	.	28.20	315.787	436.219	.	3,492.97000
	TCN-S2S-P	99.260	.	31.70	409.673	433.752	.	2,356.73000
	DeepAnt	99.345	.	35.40	2,032.630	429.024	.	517.36300
	LSTM-AE	99.810	.	30.30	374.607	435.059	.	2,964.10000
	STGAT-MAD	99.860	.	34.70	29.226	448.274	.	40,295.10000
	Donut	98.790	.	29.00	913.899	433.434	.	1,196.68000
	GenAD	99.880	.	17.50	198.002	4676.220	.	5,914.15000
	LSTM-AE-OC-SVM	99.690	.	25.80	89.146	435.542	.	12,067.70000
	MTAD-GAT	99.950	.	28.00	44.639	473.680	.	23,864.10000
	THOC	99.800	.	35.40	121.684	445.724	.	9,419.30000
	GBRF	16.600	.	36.40	129.960	427.410	.	9,343.77000
	Isolation Forest	16.490	.	24.90	309.904	423.802	.	3,833.94000
	kNN	16.700	.	43.00	62.586	4592.620	.	18,577.50000

lack of a common notion of time in a distributed architecture so that data referring to the same instant can be correctly aligned and merged. To achieve this goal, we exploit the Precision Time Protocol (PTP) [28], a state-of-the-art industrial clock synchronization protocol that guarantees microsecond clock accuracy. Due to its low-computation requirements and software implementation, PTP is also supported by constrained devices, making it the best option for a manufacturing system. This allows data generated by different sources to be aligned temporally and to associate a timestep with the global state information, computed as a result of the data fusion steps.

Finally, the merged data is rescued by a Kafka broker that handles communication with the *application layer*.

3) Adaptive Manufacturing Controller: The first block of the application layer is the AMC. This block is responsible for scheduling the production tasks and sending the commands to the production plant through specific network brokers [29].

This module is an extension of the standard Manufacturing

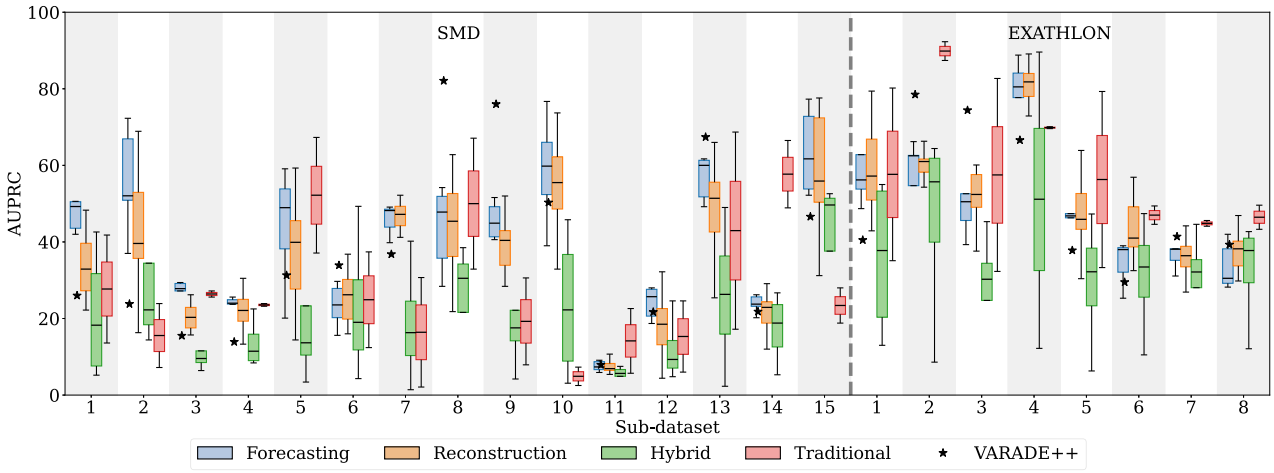

Figure 3. Distributions of AUPRC values of different categories of literature methods (forecasting-based, reconstruction-based, hybrid, and traditional) obtained on the SMD (left) and Exathlon (right) benchmarks. The AUPRC value of VARADE++ is highlighted by a star-shaped marker, in line with the boxplot of the other forecasting-based techniques.

Execution Systems (MESs) with advanced automated features, such as reconfiguration of the production line, autonomous execution of production orders, resource management, and advanced scheduling. As such, the role of the AMC is not simply sending production recipes to manufacturing equipment but also performing dynamic optimization and reorganization of production scheduling. For this reason, it has a leading part in generating appropriate counter-measures in case of a detected anomaly.

A crucial point of such countermeasures is that they must be timely, i.e., generated and actuated within some timing boundaries. If this condition is not satisfied, then the sent command may cause new anomalies instead of effectively solving the detected one.

Given the structure of our proposed IIoT architecture, we can define the total time taken from the data collection to the generation and execution of the control strategy as follows:

$$\begin{aligned} t_{tot} &= t_{msg} + t_{df} + t_{ad} + t_{amc} \\ t_{msg} &= t_{eq,df} + t_{df,ad} + t_{ad,amc} + t_{amc,eq} \end{aligned} \tag{2}$$

where t_{msg} is the total communication time, summing up the time required to send a message across all the actors within the FCP (respectively: $t_{eq,df}$ from the equipment to data fusion, $t_{df,ad}$ from data fusion to anomaly detection, $t_{ad,amc}$ from anomaly detection to AMC and $t_{amc,eq}$ from AMC back to equipment); t_{df} is the time taken to perform the data fusion, t_{ad} is the time taken to perform the anomaly detection (that is, the response time of the ML model), and t_{amc} is the time taken to elaborate a new control strategy for the machine.

4) Real-Time Anomaly Detector: The second block of the application layer is the AD algorithm, that processes the aggregated data generated by the FCP module and sends the output through specific brokers to the AMC, to allow the reaction to anomalous events that may occur in the production plant.

This block may contain any kind of AD algorithm besides the one presented in Section III-A, but with some important constraints:

- Being able to process high-dimensional MTS data with no prior information about the anomalies to be detected.
- Being able to work in real-time and enabling online feedback, i.e., to respect the timing constraints by Eq. (2).
- Being scalable to different platforms and suited for edge computation to lower latency and response time. This implies that it must not require too many resources in terms of memory and computation.

The output of the AD module is then sent to the AMC. In this way, the AMC can react to captured anomalous events by elaborating and executing appropriate counter-measures, which may include: re-scheduling production tasks, sending new commands to the production plant, scheduling maintenance, and pausing the anomalous machine, in case of parallel production phases that can be executed by other machinery [5]. As will be proven by the experimental section, VARADE++ meets all such constraints.

IV. Experimental Results

As noted in Section II-A, despite recent advances in deep learning architectures for time series processing, real-time anomaly detection remains a very open field. There is no definitive "catch-all" technique—no single method or family of methods consistently achieves optimal performance across diverse scenarios—and standardized benchmarks or evaluation metrics are still lacking. This makes a fair and thorough experimental assessment especially difficult. On top of that, most of the validations focus on the anomaly detection accuracy of the models without digging into their feasibility in industrial scenarios, where the real-time response and the scalability of edge computing applications becomes paramount.

To address this problem, we split our experimental evaluation into two parts. First, considering all the categories introduced in Section II-A, we select a list of models from the recent literature, which were shown to have promising anomaly detection performance on well-established public MTS benchmarks. We also check the positioning of VARADE++ against these counterparts on the same validation grounds. Second, we create a realistic test-bed of a production plant application by using a sensorized collaborative robotic cell in a pilot line.

Figure 4. The 3D model of the fully-fledged production line of the ICE laboratory. In the top right corner, we represent the KUKA LBR iiwa and the experimental setup used as case studies.

We integrate each counterpart model into the same IIoT infrastructure of VARADE++. Then, we compare the resulting anomaly detection frameworks on the same test bed. To assess the scalability of the methods, the comparison with VARADE++ is conducted on different types of computing platforms, considering detection accuracy, response speed, energy, and memory efficiency.

A. Benchmarking the State-of-The-Art Solutions

Anomaly detection methods are often tailored to ad-hoc applications and validated with uneven experimental settings. To establish a set of state-of-the-art methods, we rely on a recent experimental review [8]. This review tested 28 popular deep learning-based anomaly detection methods with consistent experimental settings and metrics on the most widely used public MTS datasets. More specifically, we consider the two repositories that the authors identified as the most reliable: SMD [30] and Exathlon [31], and restrict our analysis to the same sub-datasets that were used in their experiments (15 for SMD and 8 for Exathlon, respectively). To embrace all the four categories of methods introduced in Section II-A, we add three well-known *traditional* outlier-detectors to the methods already tested by [8]: kNN, Isolation Forest, and GBRF.

Among all the possible performance metrics, we choose to display the Area Under the Precision-Recall Curve (AUPRC) [8], [13], [32], for two main reasons. i) It provides a single $[0, 1]$ metric that considers all the possible threshold values set on the anomaly score. Hence, it is well-suited for comparing anomaly detection algorithms across different experimental conditions and applications. ii) It is especially useful to quantify performance in imbalanced classification tasks, such as anomaly detection. Hence, it is preferable to other widely used metrics (e.g., AUROC, which tends to be over-optimistic).

For the sake of space and readability, Figure 3 shows the results of our analysis only in aggregated form, separately for SMD (left) and Exathlon (right) benchmarks. For each analyzed sub-dataset (x-axis of the plot), we show the AUPRC of all the tested methods grouped by their category (reconstruction, forecasting, hybrid, and traditional, respectively) in box plots. By doing so, we can display the AUPRC distributions in a compact and easy-to-compare way. We use a star-shaped marker to position VARADE++ within its category (forecasting).

We can draw the following considerations:

- As already concluded by [8], the performance distribution is extremely variable. Methods that perform well in SMD may not be effective in Exathlon and vice-versa, with much variability even within the same benchmark.
- Even though there is a large within-category variability, the forecasting methods are the ones that, on average, perform the best across different datasets. Surprisingly, the traditional methods perform equally or sometimes even better than deep learning-based counterparts.
- Same as for the other methods, the performance of VARADE++ on SMD and Exathlon is variable. Nonetheless, its AUPRC values are mostly in line with the distributions of the state-of-the-art methods and, in a few cases, even superior by a good margin. This is a remarkable result, as the design of VARADE++ is specifically optimized for resource-constrained applications and not tailored to SMD and Exathlon datasets.

As already mentioned, the offline analysis on benchmarks does not allow us to draw conclusive remarks for our specific target. First, it does not consider the design characteristics of the methods, which may be tailored to very different types of data than the ones represented by SMD and Exathlon. Second, it does not consider important features such as inference speed or usability in resource-constrained scenarios. Nevertheless, benchmarking allows us to shortlist a subset of promising methods and do further testing in a near-reality context. More specifically, we select the top three methods of each category in terms of average AUPRC, excluding VARADE++. To account for the differences between the two benchmarks, we do the same for SMD and Exathlon (see Table I).

By doing so, we obtain a list of 15 distinct methods (detailed descriptions in [8]).

B. Case Study From a Production Plant

To build a realistic test bed, we leverage the ICE Laboratory,[1] a research facility of the University of Verona, Italy (depicted

[1]Industrial Computer Engineering (ICE) Laboratory - https://www.icelab.di.univr.it/

in Figure 4). The ICE Laboratory features a fully-fledged production line: a visual quality control cell to assess the quality of the manufactured pieces; a collaborative robotic cell to perform assembly operations; a CNC milling machine; two 3D printers; and an electronic functional testing machine for electronic boards. Workpieces are transported between the working cells by exploiting a mini pallet conveyor belt, 2 Automated Guided Vehicle (AGV) mobile robots, and a vertical warehouse. Each machinery is equipped with an OPCUA server, which adds smart capabilities such as data collection and command execution.

We focus our case study on a *KUKA LBR iiwa*, a 7-axis robot part of the collaborative robotic cell of ICE [42]. Each of the seven joints is instrumented with a DFRobot SEN0386 IMU sensor, measuring angles, accelerations, angular velocities, and temperatures and sending them on a serial wire at 200 Hz. Besides the IMU, the robot is monitored by a single-phase Eastron SDM230 energy meter connected through a hard-wired Modbus with an industrial Olimex ESP32-EVB ESP-32, sending data to a MQTT broker via Ethernet. Overall, the collected data stream includes 86 channels, including also a univocal ID identifying the action being performed by the robot.

To test the anomaly detection framework in a realistic scenario, we do the following. First, we monitor the KUKA during its *normal* behavior. We configure the robot to lift a maximum payload of 500 g. Then, we have it performing 30 different actions while moving a LEGO DUPLO piece of 20 g from the buffer to the pallet and back, for a total duration of 390 min. The so-obtained stream of data, normalized and rescaled between $[-1, 1]$, is used to perform the offline training of VARADE++ as well as of the 15 counterpart models. To ensure a fair comparison, all the deep learning-based models are trained implementing hyperparameters optimization, fixed 10^{-5} learning rate, and Adam optimizer; the traditional outlier detection models are trained according to their respective reference papers and the review by [8]. After training, the models are integrated into the IoT framework described in Section III-B.

To test the anomaly detectors in real-time operating conditions, we conduct a second experiment using the same robot configuration of the training phase, for a total duration of 306 minutes. During the robot's operation, we deliberately introduce controlled anomalies at random times to emulate realistic deviations in the production process. Specifically, these anomalies are generated by i) briefly interfering with the robot's movement, ii) adding extra weight to the end effector beyond the nominal payload, and iii) modifying the trajectory speed to 50% or 75% of its standard value. The first manipulation reproduces sudden, short-lived disturbances such as unintended collisions between a human worker and the robot, which appear in the data as abrupt, isolated deviations. The second simulates persistent overload conditions indicative of mechanical degradation, gradual wear, or configuration errors, resulting in long-term deviations from normal operation. The third captures gradual operational drifts associated with process variability. Together, these manipulations encompass a representative spectrum of abnormal behaviors commonly observed in industrial environments. Throughout the experiment, all anomalies are timestamped to provide ground truth for validation, while avoiding extreme conditions that could trigger the safety mechanisms of the working cell.

At inference time, the anomaly detectors are executed on the same testing data stream, employing the same computational resources. To assess the suitability to low-resource scenarios, the upper bound in terms of hardware is a standard desktop PC without a dedicated GPU. In our experiments, we test four different platforms with progressively increasing computing resources: i) Raspberry Pi 5, ii) NVIDIA Jetson Xavier NX, iii) NVIDIA Jetson AGX Orin, iv) desktop PC with an Intel i7 8700 processor and a 16 GB RAM, respectively.

All the anomaly detection frameworks are executed with Python 3.10 and PyTorch 2.0.1. The whole experimental setup, code, implementation, and dataset are made freely available via GitHub.[2]

C. Validation Metrics

To fully characterize the experimented models, in Table II, we report, for each tested platform, CPU and GPU usage at run-time, AUPRC value, inference frequency (Hz), RAM usage (MB) and Energy consumption (mWh).

To ease comparisons and improve the readability of the results, we also provide a graphical representation of a set of significant variables in the form of radar plots (see Figure 5). The five axes represent five main figures of merit, min-max scaled between zero and one:

- *Anomaly detection performance*, quantified by the AUPRC value as in the previous charts.
- *Inference frequency* (Freq) computed as $1/t$, where t is the prediction time of the model.
- *Memory efficiency* (Mem), describing the impact of the execution over the overall available memory. It is obtained as $1 - PM/M$, where PM is the peak RAM usage of the model and M is the total physical memory available.
- *Energy efficiency* (En), describing the impact of energy consumption. It is obtained as $1 - E/E_{MAX}$, where E is the energy intake in mAh and E_{MAX} the highest energy intake, taken as a reference. By doing so, the lower the energy consumption, the higher the efficiency value.
- *Size efficiency* (Size), quantifying the storage cost of the model based on its number of parameters. Same as for energy, the efficiency score is obtained as $1 - S/S_{MAX}$, where S is the number of parameters of the model and S_{MAX} is the size of the largest model.

Each tested configuration corresponds to a pentagon in Figure 5. A vertex of the pentagon provides a quick glimpse at the relative goodness of the corresponding metric (the higher the distance from the origin, the better); the area of the pentagon communicates the relative goodness of the five metrics altogether (the larger the area, the better).

D. Discussion of Results

1) Anomaly Detection Performance: This figure of merit is paramount, as it defines the capability of the anomaly detector to do its job irrespective of the available computational resources. In this regard, VARADE++ stands out from its counterparts as the most performing solution (42.6% AUPRC), together with

[2]https://github.com/alessioMas/TimeSeAD-VaradePlus

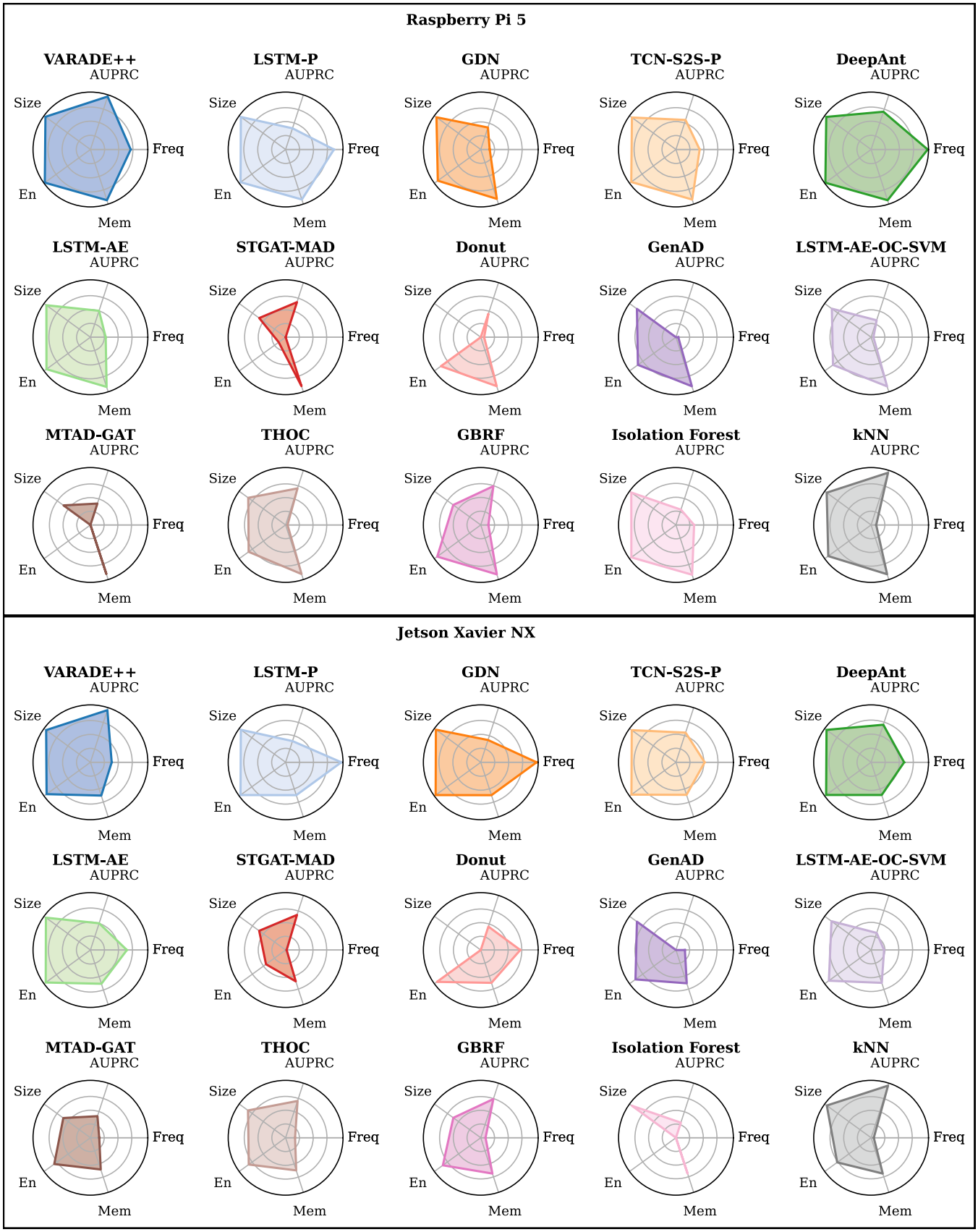

Figure 5. Radar plots of VARADE++ (always in blue, top-left) and its counterparts on different computing platforms: (a) Raspberry Pi 5, (b) Jetson Xavier NX, (c) Jetson AGX Orin, (d) desktop PC with Intel i7 8700 processor and 16 GB RAM. Axes represent AUPRC, inference frequency, memory efficiency, energy efficiency, and size efficiency. The details are reported in Section IV-C).

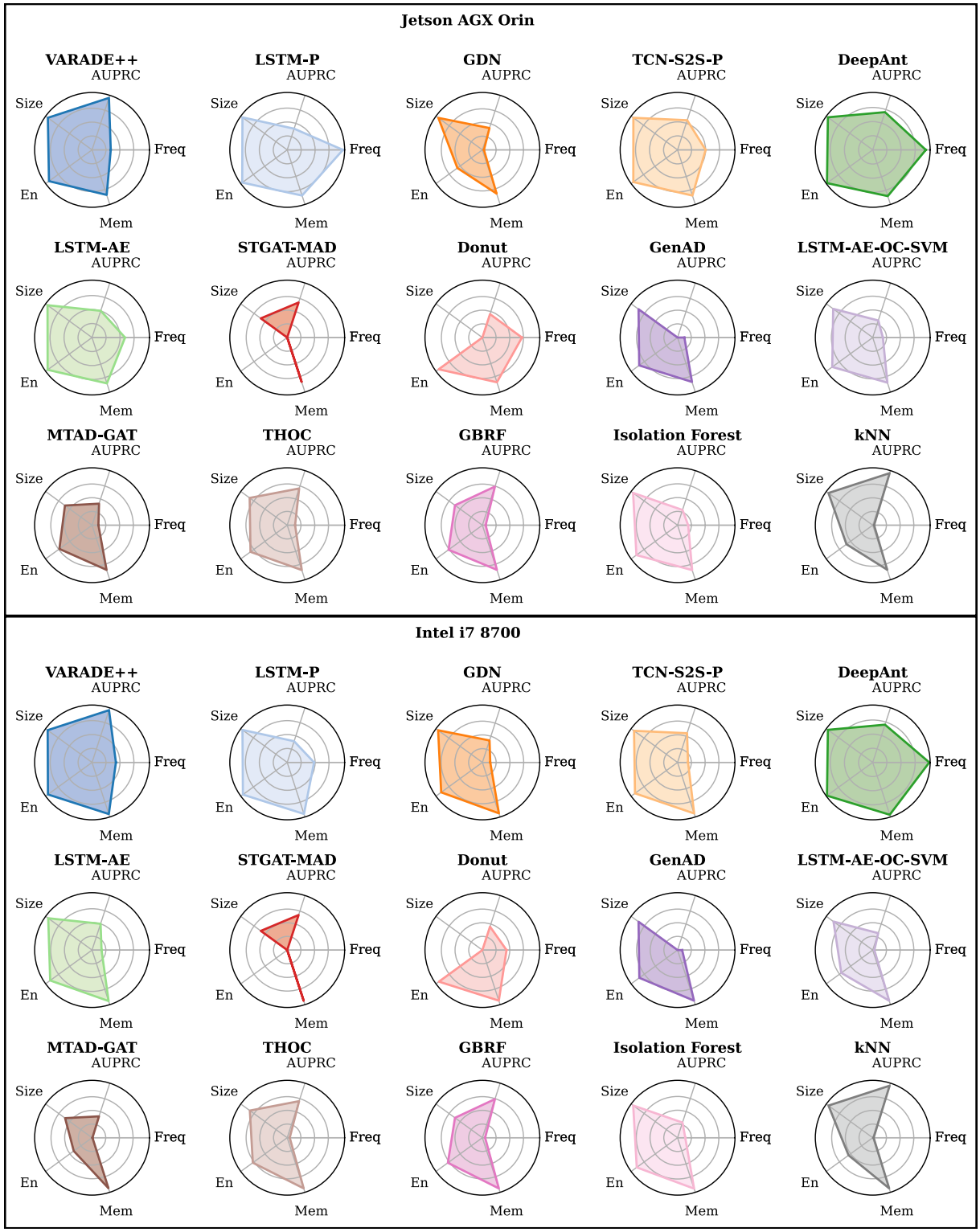

Fig. 5. *(Continued).*

kNN (43% AUPRC). There is no apparent relation between the AUPRC value and the category of the anomaly detection paradigm. Besides VARADE++, non-deep learning-based methods like kNN and GBRF have the best AUPRC values.

2) Inference Frequency: Inference frequency is related to the prediction speed and, hence, to the real-time capabilities of the anomaly detector. This is especially important in industrial applications, where detecting an anomaly may be the first step of a closed-loop feedback strategy to avoid a process failure. The optimal value of the inference frequency is application-dependent. In our case study, data collection happens at 200 Hz, corresponding to the frequency of the IMU sensors on the robot.

Starting from this consideration, we can reasonably argue that any frequency value around or higher than 200 Hz ensures processing data without delays because the model can provide an output before the next data point is collected. By looking at Table II, we can see that VARADE++ exceeds 200 Hz in all the computing boards except Jetson Xavier NX, besides VARADE++: TCN-S2S-P, DeepAnt, LSTM-P, Donut, and LSTM-AE.

Always looking at Table II, it is interesting to note that the highest inference frequency values are obtained by executing the anomaly detection models on the CPU (specifically, on Raspberry Pi 5 and i7 8700). More in-depth considerations can be made by looking at the CPU usage and GPU usage: when executing the models on the CPU, we can see that mean CPU usage is higher than 90% for most models except for GBRF, Isolation Forest, and kNN, whose implementations are not parallelizable. On the other hand, when GPU execution is available, the mean GPU usage is around 40%. From these results, we can reasonably conclude that, in our use case, CPU execution leads to better results, as it avoids the overhead of copying data from CPU to GPU and vice-versa. Furthermore, examining CPU usage on both the i7 8700 and the Raspberry Pi 5 reveals that parallelizable models can better exploit the i7's more powerful multi-core architecture, leading to near-complete CPU saturation (mean usage greater than 99%). This clearly illustrates the interplay between the software's design (specifically, its ability to parallelize workloads) and the architectural capabilities of the underlying hardware.

The radar plots of Figure 5 show a typical trade-off between inference frequency and detection performance: the fastest models like DeepAnt and LSTM-P have AUPRC values 25% to 50% lower than VARADE++ and kNN. On the other hand, kNN has a good AUPRC, but a very low inference frequency. Remarkably, VARADE++ is the only method with good anomaly detection performance at an inference speed equal to or above 200 Hz.

3) Memory Efficiency: This metric reflects the overall impact on the system memory of executing the anomaly detector. Based on our experiments, we can see similar values of RAM usage for most methods when executing on the same platform. Interestingly, all the methods require more memory on the Jetson Xavier NX board, corresponding to a memory efficiency of around 0.65.

4) Energy Efficiency: Energy intake is another important parameter to assess the suitability of a method to edge applications. We can again see from the results that VARADE++ is among the most performing methods, irrespective of the computing platform. As expected, the platform with the lowest absolute consumption values is the Raspberry Pi 5.

5) Size Efficiency: Most tested methods are similar in size, with efficiency always close to 1. This confirms that we are not trying to compare models that are too different from each other regarding the number of learnable parameters. One of the few exceptions is Donut, whose backbone based on deep variational autoencoder is considerably larger than all the other counterparts [8].

6) Trade-Off Between Efficiency and Accuracy: As anticipated, the area of the radar plot provides a glimpse at all the figures of merit. It ultimately allows us to appreciate the trade-off between accuracy and efficiency metrics.

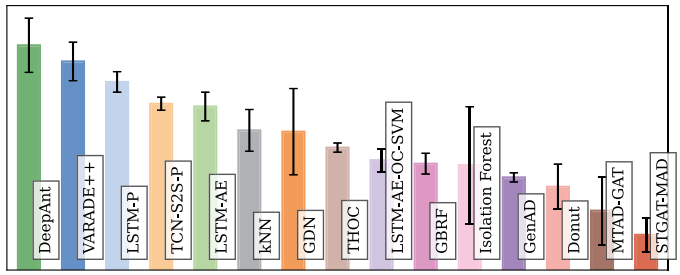

Figure 6. Mean and standard deviation of the area of the radar charts over the different computing platforms.

Figure 6 shows each model's mean and standard deviation of the radar plot area over the different computing platforms, allowing us to derive the following considerations. The three top-ranked methods are, at a short distance from one another: DeepAnt, VARADE++ and LSTM-P. As pointed out before, the performance of DeepAnt and LSTM-P is mainly driven by speed, at the cost of a sub-optimal detection performance. On the other hand, VARADE++ provides high AUPRC, at an inference frequency higher than 200 Hz in most configurations. Hence, we can conclude that VARADE++ offers the best trade-off for the given anomaly detection task.

E. Feasibility of the Control Feedback

In the previous section, we concluded that VARADE++, especially when executed on a low-cost computing platform like the Raspberry Pi 5, provides an optimal balance of detection accuracy, use of computational resources, and inference speed. Now, we dive more into the actual usability of the experimented anomaly detection for control strategies. As mentioned before, the anomalies in this experiment do not include hazardous conditions that are usually handled by the safety mechanism implemented directly on the working cells.

Considering the structure of our proposed architecture presented in Section III-B, we can apply Eq. (2) to estimate the total time taken from the data collection to the generation and execution of the control strategy on the Kuka robot. The communication time has been extensively explored in [5], where the authors tested different architecture configurations and network workloads. This analysis estimates an upper-bound value for exchanging messages within the FCP of 8 ms under heavy network workloads. As such, we can consider the total time needed to transmit all the messages from data collection to anomaly feedback $t_{msg} = 32$ ms. Sensor data is generated at a frequency of 200 Hz, leading to a data fusion process that can take up to $t_{df} = 5$ ms. For the anomaly detection task, we can consider a value of $t_{ad} = 1.59$ ms if we use VARADE++ deployed on the Raspberry Pi 5 (see Table II). From these considerations, we can simplify Equation (2) as follows:

$$t_{tot} = 39ms + t_{amc} \tag{3}$$

The remaining undefined variable is t_{amc}, whose value depends on the type of strategy applied, ranging from milliseconds to seconds or even minutes. Exhaustive strategies could take minutes to compute the optimal solution among all the possible ones. In contrast, greedy approaches may be able to find sub-optimal and yet feasible solutions in a shorter time. The choice depends on the use case and its specific requirements. Let us consider the simplest case of a control strategy that pauses the robot when an anomaly is detected. In this specific scenario, the contribution of t_{amc} in Equation (3) is negligible, allowing the robot to stop almost immediately. Specifically, assuming the robot moves at its maximum speed of 0.15 m/s for the entire duration of 39 ms, the displacement of each joint from its initial position would be at most 0.585 mm, which can be considered negligible in most applications.

From a broader perspective, it is important to note that, with only 39 ms spent on communication and data management, our system can react to any anomaly within fewer than 10 data samples. This meets the real-time timing constraints defined in the literature [43].

V. Conclusions and Future Work

Our experiments demonstrate that VARADE++ strikes an effective balance between anomaly detection accuracy and computational efficiency, making it well suited for deployment in production environments and advanced smart robotics systems. While no single method can address every possible anomaly detection scenario or data modality, our results provide strong evidence of VARADE++'s readiness for real-world industrial applications. To promote transparency, reproducibility, and future research, both the dataset and the source code are publicly released.

Despite these achievements, a few aspects warrant further investigation. By learning what is *normal* directly from training data, VARADE++ is inherently robust to typical noise patterns arising during data acquisition. However, as with any unsupervised approach, distinguishing true process anomalies from previously unseen noise remains a domain-dependent challenge requiring careful calibration. Domain-specific tuning is also needed to optimize key hyperparameters, such as the forecasting window and the anomaly score threshold. Furthermore, although our findings confirm VARADE++'s real-time performance and its potential to support corrective control strategies, the design and implementation of such strategies is again application-specific, and fall beyond the present study's scope.

Looking ahead, we plan to extend VARADE++ by incorporating closed-loop feedback mechanisms and expanding its applicability to additional devices and heterogeneous production data. These developments will move us closer to realizing a fully integrated Digital Twin of the production plant.

Acknowledgment

This manuscript reflects only the Authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

References

[1] K. Kubiak, G. Dec, and D. Stadnicka, "Possible applications of edge computing in the manufacturing industry–systematic literature review," *Sensors*, vol. 22, no. 7, 2022, Art. no. 2445.

[2] L. Bacchiani et al., "Low-latency anomaly detection on the edge-cloud continuum for industry 4.0 applications: The SEAWALL case study," *IEEE Internet Things Mag.*, vol. 5, no. 3, pp. 32–37, Sep. 2022.

[3] A. Mascolini et al., "VARADE: A variational-based AutoRegressive model for anomaly detection on the edge," in *Proc. 61st ACM/IEEE Des. Automat. Conf.*, San Francisco, CA, USA, 2024, pp. 1–6.

[4] S. Gaiardelli et al., "A data fusion service-oriented infrastructure for production line monitoring," in *Proc. 25th IEEE Int. Conf. Ind. Technol.*, 2024, pp. 1–8.

[5] S. Gaiardelli et al., "Enabling service-oriented manufacturing through architectures, models, and protocols," *IEEE Access*, vol. 12, pp. 85259–85274, 2024.

[6] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," *Esann*, vol. 2015, 2015, Art. no. 89.

[7] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 5, pp. 4027–4035.

[8] D. Wagner, T. Michels, F. C. Schulz, A. Nair, M. Rudolph, and M. Kloft, "TimeSeAD: Benchmarking deep multivariate time-series anomaly detection," *Trans. Mach. Learn. Res.*, Apr. 2023. [Online]. Available: https://openreview.net/forum?id=iMmsCI0JsS

[9] M. Said Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Network anomaly detection using LSTM based autoencoder," in *Proc. 16th ACM Symp. QoS Secur. Wireless Mobile Netw.*, 2020, pp. 37–45.

[10] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PLoS one*, vol. 11, no. 4, 2016, Art. no. e0152173.

[11] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discov. from Data*, vol. 6, no. 1, pp. 1–39, 2012.

[12] H. Huang, L. Yang, Y. Wang, X. Xu, and Y. Lu, "Digital twin-driven online anomaly detection for an automation system based on edge intelligence," *J. Manuf. Syst.*, vol. 59, pp. 138–150, Apr. 2021.

[13] S. Kim, K. Choi, H.-S. Choi, B. Lee, and S. Yoon, "Towards a rigorous evaluation of time-series anomaly detection," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 7194–7201.

[14] X. Yu, X. Yang, Q. Tan, C. Shan, and Z. Lv, "An edge computing based anomaly detection method in IoT industrial sustainability," *Appl. Soft Comput.*, vol. 128, 2022, Art. no. 109486.

[15] G. Nain, K. Pattanaik, and G. Sharma, "Towards edge computing in intelligent manufacturing: Past, present and future," *J. Manuf. Syst.*, vol. 62, pp. 588–611, 2022.

[16] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in industrial Internet of Things: Architecture, advances and challenges," *IEEE Commun. Surv. Tuts.*, vol. 22, no. 4, pp. 2462–2488, Fourth quarter 2020.

[17] H. Xu, W. Yu, D. Griffith, and N. Golmie, "A survey on industrial Internet of Things: A cyber-physical systems perspective," *IEEE Access*, vol. 6, pp. 78238–78259, 2018.

[18] A. Vaswani et al., "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*2023, pp. 6000–6010.

[19] W. Yu et al., "MetaFormer is actually what you need for vision," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10809–10819.

[20] Z. Li, Z. Rao, L. Pan, and Z. Xu, "MTS-Mixers: Multivariate time series forecasting via factorized temporal and channel mixing," 2023, *arXiv:2302.04501*.

[21] J. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ontanon, "FNet: Mixing tokens with fourier transforms," in *Proc. 2022 Conf. North Amer. Chapter Assoc. Comput. Linguist.: Hum. Lang. Technol.*, M. Carpuat, M.-C. de Marneffe, and I. V. M. Ruiz, Eds., Jul. 2022, pp. 4296–4313. [Online]. Available: https://aclanthology.org/2022.naacl-main.319

[22] I. Tolstikhin et al., "MLP-Mixer: An all-MLP architecture for vision," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 24261–24272.

[23] B. Steiner, M. Elhoushi, J. Kahn, and J. Hegarty, "MODeL: Memory optimizations for deep learning," in *Proc. 40th Int. Conf. Mach. Learn.*, Honolulu, HI, USA, 2023, pp. 32618–32632.

[24] O. Weng et al., "Tailor: Altering skip connections for resource-efficient inference," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 17, no. 1, Sep. 2023, Art. no. 11.

[25] A. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8162–8171.
[26] S. F. Chevtchenko et al., "Anomaly detection in industrial machinery using IoT devices and machine learning: A systematic mapping," *IEEE Access*, vol. 11, pp. 128288–128305, 2023.
[27] N. Dall'Ora, K. Alamin, E. Fraccaroli, M. Poncino, D. Quaglia, and S. Vinco, "Digital transformation of a production line: Network design, online data collection and energy monitoring," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 1, pp. 46–59, Jan.–Mar. 2022.
[28] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Standard 1588-2019, 2019.
[29] S. Gaiardelli, S. Spellini, M. Panato, M. Lora, and F. Fummi, "A software architecture to control service-oriented manufacturing systems," in *Proc. Des., Automat. Test Eur. Conf. Exhib.*, 2022, pp. 40–43.
[30] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 2828–2837.
[31] V. Jacob, F. Song, A. Stiegler, B. Rad, Y. Diao, and N. Tatbul, "Exathlon: A benchmark for explainable anomaly detection over time series," in *Proc. VLDB Endowment*, vol. 14, no. 11, Jul. 2021, pp. 2613–2626.
[32] S. Schmidl, P. Wenig, and T. Papenbrock, "Anomaly detection in time series: A comprehensive evaluation," *Proc. VLDB Endowment*, vol. 15, no. 9, pp. 1779–1797, 2022.
[33] Y. He and J. Zhao, "Temporal convolutional networks for anomaly detection in time series," *J. Phys.: Conf. Ser.*, vol. 1213, no. 4, 2019, Art. no. 042050.
[34] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "DeepAnT: A deep learning approach for unsupervised anomaly detection in time series," *IEEE Access*, vol. 7, pp. 1991–2005, 2018.
[35] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "LSTM-based encoder-decoder for multi-sensor anomaly detection," 2016, *arXiv:1607.00148*.
[36] C. Zhang et al., "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 01, pp. 1409–1416.
[37] J. Zhan et al., "STGAT-MAD: Spatial-temporal graph attention network for multivariate time series anomaly detection," in *Proc. 2022 IEEE Int. Conf. Acoust., Speech Signal Process.*, 2022, pp. 3568–3572.
[38] X. Hua et al., "GenAD: General representations of multivariate time seriesfor anomaly detection," 2022, *arXiv:2202.04250*.
[39] H. Xuet al., "Unsupervised anomaly detection via variational auto-encoder for seasonal KPIS in web applications," in *Proc. World Wide Web Conf.*, 2018, pp. 187–196.
[40] H. Zhao et al., "Multivariate time-series anomaly detection via graph attention network," in *Proc. IEEE Int. Conf. Data Mining*, 2020, pp. 841–850.
[41] L. Shen, Z. Li, and J. Kwok, "Timeseries anomaly detection using temporal hierarchical one-class network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 13016–13026.
[42] "Sensitive robotics LBR IIWA," KUKA, 2016. Accessed: Sep. 17, 2025. [Online]. Available: https://www.kuka.com/en-de/products/robot-systems/industrial-robots/lbr-iiwa
[43] M. Castellano-Quero, M. Castillo-Lopez, J.-A. Fernandez-Madrigal, V. Arévalo-Espejo, H. Voos, and A. Garcia-Cerezo, "A multidimensional bayesian architecture for real-time anomaly detection and recovery in mobile robot sensory systems," *Eng. Appl. Artif. Intell.*, vol. 125, 2023, Art. no. 106673.

Alessio Mascolini received the bachelor's and master's degrees in biomedical engineering from Politecnico di Torino, Italy, in 2017 and 2019, respectively. He is currently working toward the PhD degree (3rd year) with Italian National PhD Program in artificial intelligence, hosted by Politecnico di Torino in a federation with 61 universities and research institutions. His research focuses on the development and application of generative models for industrial use cases, particularly where data scarcity is a constraint.

Sebastiano Gaiardelli (Member, IEEE) received the PhD degree in computer science and engineering from the University of Verona, Verona, Italy, in 2025. He is currently a postdoctoral researcher with the Technical University of Munich, Munich, Germany. He is also a co-founder and the scientific advisor with FACTORYAL S.r.l., a startup specializing in factory automation software that originated as a spin-off from the University of Verona. His research interests include the optimization, reconfiguration, and verification of cyber-physical production systems, deterministic execution semantics for distributed embedded systems, and on-device machine learning for industrial monitoring.

Francesco Ponzio (Member, IEEE) received the PhD degree in control and computer engineering from the Polytechnic University of Turin, Italy, in 2021. He is currently an assistant professor (RTD-A) with the Department of Control and Computer Engineering, Polytechnic University of Turin. His research interests include artificial intelligence, machine learning, and image processing in medical and smart manufacturing applications.

Nicola Dall'Ora (Member, IEEE) received the PhD degree in computer science from the University of Verona, Italy, in 2023. He is currently an assistant professor with the Department of Engineering Sciences, Guglielmo Marconi University, Rome, Italy. He also collaborates with the Department of Engineering for Innovation Medicine (Section of Engineering and Physics), University of Verona, Italy, as a research consultant. His research interests include the modeling and simulation of cyber-physical and embedded systems, digital twins, reliability, and functional safety of critical systems, and focuses on abstraction and fault injection techniques for analog and mixed-signal circuits, and IoT and wearable sensing architectures for healthcare and human performance monitoring.

Stefano Quer (Senior Member, IEEE) received the MS degree in electronic engineering, in 1991, and the PhD degree in computer engineering, in 1996. He was a visiting Faculty with the Department of Electronic Engineering and Computer Science, University of California at Berkeley, CA, USA, and an intern with the "Advanced Technology Group", Synopsys Inc., Mountain View, CA, USA, and with the Alpha Development Group, Compaq Computer Corporation, Shrewsbury, MA, USA. He was also a compaq computer corporation consultant. He is currently a professor with the Department of Control and Computer Engineering, Politecnico di Torino, Italy. His research interests mainly include systems and tools for CAD for VLSI, formal methods for hardware and software systems, and embedded systems, and focuses on sequential and concurrent algorithms and optimization techniques able to achieve acceptable solutions with limited resources.

Franco Fummi (Senior Member, IEEE) received the Laurea degree in electronic engineering and the PhD degree in electronic and communication engineering from the Polytechnic of Milan, in 1990 and 1994, respectively. Since Mar. 2001, he has been a full professor of computer architecture with Università di Verona. He is also leading the Cyber-Physical and IoT Systems Design (CISD) Group, Università di Verona, where he is composed of more than 20 people and working on hardware description languages and electronic design automation methodologies for modeling, verification, testing, and optimization of cyber-physical systems. He is also the co-founder of two spin-off companies, such as EDALab, focused on networked embedded systems design, and the automation control software company FACTORYAL.

Sara Vinco (Senior Member, IEEE) received the PhD degree in computer science from the University of Verona, Verona, Italy, in 2013. Since 2021, she has been an associate professor with Politecnico di Torino, Turin, Italy. Her research interests include digital twins, energy efficient electronic design automation, efficient simulation and optimization of energy systems, and techniques for simulation and validation of heterogeneous embedded systems.

Santa Di Cataldo (Member, IEEE) received the MSc degree in biomedical engineering and the PhD degree in computer and systems engineering from the Politecnico di Torino, Turin, Italy, in 2006 and 2011, respectively. She is currently an associate professor with the Department of Control and Computer Engineering, Politecnico di Torino. Her research interests mainly include artificial intelligence, machine learning, and computer vision, with applications ranging from medical to industry.

Open Access funding provided by ‘Universita Degli Studi di Verona’ within the CRUI CARE Agreement