VARADE: a Variational-based AutoRegressive model for Anomaly Detection on the Edge

VARADE: a Variational-based AutoRegressive model for Anomaly Detection on the Edge

Alessio Mascolini[1], Sebastiano Gaiardelli[2], Francesco Ponzio[1], Nicola Dall'Ora[2], Enrico Macii[1], Sara Vinco[1], Santa Di Cataldo[1], Franco Fummi[2]
[1]Politecnico di Torino, Turin, Italy, {name.surname}@polito.it
[2]University of Verona, Verona, Italy, {name.surname}@univr.it

ABSTRACT

Detecting complex anomalies on massive amounts of data is a crucial task in Industry 4.0, best addressed by deep learning. However, available solutions are computationally demanding, requiring cloud architectures prone to latency and bandwidth issues. This work presents VARADE, a novel solution implementing a light autoregressive framework based on variational inference, which is best suited for real-time execution on the edge. The proposed approach was validated on a robotic arm, part of a pilot production line, and compared with several state-of-the-art algorithms, obtaining the best trade-off between anomaly detection accuracy, power consumption and inference frequency on two different edge platforms.

KEYWORDS

Anomaly detection, Edge computing, Deep learning, Cyber-physical systems, Process monitoring.

ACM Reference Format:
Alessio Mascolini[1], Sebastiano Gaiardelli[2], Francesco Ponzio[1], Nicola Dall'Ora[2], Enrico Macii[1], Sara Vinco[1], Santa Di Cataldo[1], Franco Fummi[2]. 2024. VARADE: a Variational-based AutoRegressive model for Anomaly Detection on the Edge. In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3655691

ACKNOWLEDGMENTS

The work has been partially supported by PRIN 2022T7YSHJ SMART-IC - Next Generation EU project. This manuscript reflects only the Authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

1 INTRODUCTION

In any production context, the downtime of a machine due to the sudden breakdown of mechanical, hydraulic, or electrical components leads to severe losses in terms of time and money. For this reason, efforts are spent for the early detection of any irregular behavior of the production line to avoid sudden stops, enable specific preventive maintenance actions, and reduce the environmental impact. This evolution is enabled by transforming traditional production machinery into Cyber-Physical Systems (CPSs), where sensor devices, communication technologies, and data analytics cooperate to manage production failures in advance [25].

In an industrial CPS scenario, the most crucial resource is the availability of data reflecting the different aspects of production. Such data consist of multiple interdependent variables rapidly evolving over time, thus falling under the typical definition of *Multivariate Time Series (MTS)* [14]. After collection, the time series, originated by heterogeneous sensors and data sources, are integrated through Industrial Internet of Things (IIoT) technologies and made available for anomaly detection, visualization, and analysis [27].

Although extensive research has been carried out on Multivariate Time Series Anomaly Detection (MTSAD), current solutions typically lack the flexibility and scalability that is required for an effective real-time deployment [19, 28]. In most proposed solutions, the raw data are in fact streamed through the IIoT network to a cloud platform [11], where an expert data-driven system is in charge of the anomaly detection stage. This typically results in high latency owed to communication overhead [2]. Unlike Internet of Things (IoT) networks, IIoT networks are characterized by sensors transmitting a massive amount of data that must be processed in real-time [8, 20]. The order of magnitude of the transmitted data can be GB/s for large production plants, making cloud processing impracticable due to the bandwidth requirements and impairments [20]. All such considerations highlight that anomalies should be detected *as soon as possible* and *as close as possible* to the monitored CPS, preferably with real-time or near real-time response, rather than on the cloud. This makes *edge computing* strategic to maintain the overall system functionally safe [14, 26].

On top of these considerations, in our study, we propose VARADE, a novel real-time and *edge-friendly* anomaly detection solution that provides a new efficient training paradigm for *light* MTSAD. The autoregressive framework of VARADE allows handling streaming data with minimal latency. Furthermore, its variational formulation obtains the best compromise between model compactness and anomaly detection accuracy, making it best suited for real-time execution on the edge.

To prove the effectiveness of VARADE in a real industrial CPS scenario, we employ a collaborative robot working in a fully-fledged manufacturing line and providing a continuous stream of heterogeneous sensor data. On this testbed, we compare VARADE with a comprehensive set of state-of-the-art *light* (i.e., edge-suitable) MTSAD solutions on two different edge platforms. Our experiments demonstrate that VARADE performs well even with limited computational resources, with an optimal balance between required power, anomaly detection accuracy, and inference frequency of the model, which can be varied according to the industrial machinery being monitored.

DAC '24, June 23–27, 2024, San Francisco, CA, USA

https://doi.org/10.1145/3649329.3655691

This paper is organized as follows. Section 2 presents the necessary background and related works. Section 3 details the proposed VARADE anomaly detection method, as well as the other benchmarked solutions. Section 4 describes the collaborative robot case-study and discusses the experimental results. Finally, section 5 draws our concluding remarks.

2 BACKGROUND AND RELATED WORKS

MTSAD scenarios are best addressed with Deep Learning (DL) methodologies, that recently proved to be more effective in tackling complex anomalies in MTS data [13, 23] than traditional anomaly detection methods (e.g., based on clustering or statistical indexes [29]). Nonetheless, most DL-based solutions present major drawbacks in terms of required data transmission and/or high computational cost [16, 28]. On the other hand, *light* MTSAD models compatible with edge computing are typically based on tiny and scaled Convolutional Neural Networks (CNNs), that need to be trained with huge sets of annotated anomalies [16, 19, 24, 28]. Collecting and annotating such training sets is however unfeasible in most industrial applications [13, 19].

To circumvent this problem, the most promising approach is to learn the characteristics of a "normal" behavior from a large amount of non-anomalous data so as to be able to identify any events that significantly deviate from the normality, with three different strategies: i) forecasting-based, ii) reconstruction-based, and iii) outlier detection methods.

Forecasting-based methods learn to predict a number of time steps leveraging a current context window. Then, they compare the predicted values with the observed ones to identify anomalies [23]. A large number of studies in this group employ autoregressive Long Short-Term Memory (LSTM) networks, a type of recurrent network able to learn long-term time dependencies in multivariate data [4, 12, 17]. A recent work leverages instead a forest of gradient boosted regression trees to detect anomalies in a Digital Twin-driven industrial context, by examining the residuals from the forecasts of an ensemble of weak predictors [9].

Reconstruction-based methods encode the characteristics of a normal time series into a latent representation and learn to reconstruct new data starting from it. The reconstruction error is then exploited to discriminate the anomalous values from the normal ones. The most popular methods in this group are built on top of autoencoders (AEs), encoder-decoder neural networks where the encoder learns a compressed version of the input data, and the decoder learns to recreate the input starting from the encoded representation. Among the others, [10] employed convolutional AEs for anomaly detection in an IoT-inspired environment, and proved that reducing the size, complexity, and training cost of the AE did not lower its ability to identify anomalies.

Outlier detectors identify anomalies based on their dissimilarity from regular data points in the feature space. Popular edge-friendly examples in this group are based on k-Nearest Neighbors (kNN), identifying anomalous values based on the distance from their neighbours, and Isolation Forest, that uses the number of binary splits necessary for an ensemble of decision trees to isolate the point from the rest of the data [15].

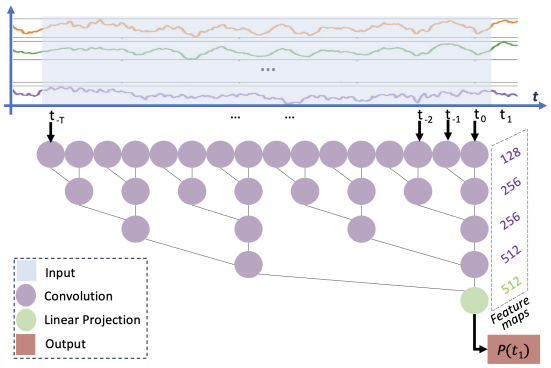

Figure 1: Architecture of VARADE. Current (t_0) and past time steps ($t_{-1} \dots t_{-T}$) are processed by a cascade of convolutional layers and a final linear projection. The output is the estimated probability distribution of the next time step, $P(t_1)$.

3 METHODS

3.1 Proposed solution

VARADE works to strike a balance between traditional techniques, that offer quick inference but with limited accuracy, and DL models, that learn complex patterns but require substantial computational resources, not available at the edge. Our design choice is to employ a *forecasting-based autoregressive framework*: by predicting samples one at a time based on previous ones, this framework is naturally suited to handle streaming data with minimal latency.

Figure 1 illustrates in principle the proposed architecture. The model takes as input the samples at the current (t_0) and past time steps (t_{-1},...t_{-T}, with $T = 16$ for visualization purposes), and passes them through a set of convolutional layers with ReLU activations and a linear projection, to finally predict a single future time step, t_1. The reason behind this architectural choice lies in the consideration that model inference speed of CNNs is commonly limited by memory bandwidth, especially with SIMD implementations for CPUs and CUDA kernels for GPUs [5]. By using convolutions with kernel size and stride of 2, we obtain that the time-dimension is halved at every new layer, leading to very limited memory usage and bandwidth requirements compared to the number of parameters, and hence to faster inference. On the other hand, the number of feature maps is doubled every two layers (see Figure 1), helping the network to learn more complex and abstract features.

Conventional forecasting-based anomaly detectors work by considering an *anomaly score*, measured as the euclidean norm between the forecasted value and the measured one. In our experiments, we have observed that a DL-based autoregressive model compact enough for real-time execution on edge fails to deliver satisfactory forecasting performance, dramatically affecting the quality of the anomaly scores. This lead us to a probabilistic approach, where the model outputs a probability distribution of the possible values for the next data point in the sequence ($P(t_1)$ in Figure 1).

Predicting a probability distribution using a neural network is a complex problem, which can be greatly simplified by constraining the distribution to be Gaussian. This approach, known as *variational inference* [3], leads to a simpler optimization problem where the objective is to find the mean and variance which minimize the

loss function (details will follow). The additional advantage of the Gaussian constraint is that the variance can be interpreted as the uncertainty of the prediction: since we expect the model to be more confident in its prediction when the system is operating normally, and less confident when an anomaly is occurring, the variance can be directly used as an *anomaly score*.

Summarizing, the proposed architecture is composed by N convolutional layers, with time-dimension halved at every layer. Hence, N strictly depends on the input window size T. In our work, we set $T = 512$, resulting into a total of 8 layers. Conversely, the number of feature maps is doubled every two layers starting from 128, which leads to $1,024$ in the final layer. The output mean and variance values of the estimated probability distribution $P(t_1)$ are obtained at last by linear projection.

3.2 Derivation of the loss function

As loss function we employ the inverse of the Evidence Lower BOund (ELBO), that provides a lower bound on the log evidence, $log\, p(x)$, where x represents the observed data. Thus, maximizing the ELBO leads to a better approximation of the true posterior.

The ELBO can be decomposed into two terms:

- The expectation of the log-likelihood under the approximate posterior, which pushes the approximate distribution to put more probability mass on configurations of the latent variables that explain the observed data well.
- The negative divergence between the approximate and prior distribution, which encourages the approximate distribution to be close to the prior.

By presuming a Gaussian distribution, our model predicts both the mean and the logarithm of the distribution's variance. We opt for the logarithm over the simple variance, as the latter can only be positive. Hence, the reconstruction loss is essentially computing the negative log-likelihood of the observed data under a Gaussian distribution assumption.

Let us assume that our data y is normally distributed with a mean of μ and a variance of σ^2. The probability density function (PDF) of a normal distribution is given by:

$$p(y|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right) \tag{1}$$

Taking the negative logarithm of the PDF to get the negative log-likelihood (NLL), we have:

$$NLL(y|\mu, \sigma^2) = -\log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)\right) \tag{2}$$

After simplifying the above equation, we get:

$$NLL(y|\mu, \sigma^2) = \frac{1}{2}\log(2\pi\sigma^2) + \frac{(y-\mu)^2}{2\sigma^2} \tag{3}$$

Given that $\log(2\pi)$ is just a constant, we can ignore it during optimization (as it depends on the derivative of the loss, and the derivative of a constant is zero). This simplifies to:

$$NLL(y|\mu, \sigma^2) = \frac{1}{2}\log(\sigma^2) + \frac{(y-\mu)^2}{2\sigma^2} \tag{4}$$

So, in our case, the reconstruction loss we use is:

$$L_{\text{recon}} = \frac{1}{2}\left(\log(\sigma^2_{\text{pred},i}) + \frac{(y_i - \mu_{\text{pred},i})^2}{\sigma^2_{\text{pred},i}}\right) \tag{5}$$

where σ^2_{pred} represents the predicted variance, μ_{pred} the predicted mean and i the current step. This formula encourages our model to predict a distribution close to the actual data.

The next part of the loss calculation introduces the Kullback–Leibler (KL) divergence, which quantifies the difference between our predicted distribution and our prior, a standard Gaussian distribution. This is computed as:

$$D_{KL} = -\frac{1}{2}\left(1 + \log(\sigma^2_{\text{pred}}) - \mu^2_{\text{pred}} - \sigma^2_{\text{pred}}\right) \tag{6}$$

The D_{KL} term encourages the model to predict the data mean and variance when it is uncertain. This helps regularize our model and is critical to employ our anomaly detection method.

The final loss function L is a weighted sum of the reconstruction loss L_{recon} and the KL divergence D_{KL}:

$$L = L_{\text{recon}} + \lambda D_{KL} \tag{7}$$

Thanks to the KL divergence term, our model learns to predict a higher variance when it is uncertain about the next value, and a low variance when it is confident. During inference, the mean prediction is removed, and the variance is directly used as an *anomaly score*: the higher the score, the larger the detected anomaly.

3.3 Baseline solutions

As a baseline for our proposed method, in this study, we implement and analyze a representative sample of *light* anomaly detectors that have been successfully deployed in edge computing scenarios, by considering the approaches in section 2.

- *Autoregressive Long Short-Term Memory (AR-LSTM).* A recurrent architecture featuring 5 LSTM recurrent layers with 256 feature maps each, followed by 2 fully connected layers. The anomaly score is then calculated as the euclidean norm of the difference between predicted and real value, as in many previous works [4, 12, 17, 21]. To find the best configuration for our specific task, we follow the memory-efficient paradigm introduced by [22], and pick a number of layers equal to 5 based on past experiments at a similar window size [1].
- *Gradient Boosted Regression Forest (GBRF).* The technique presented in [9], with minor modifications to boost the anomaly detection capabilities: the number of decision trees is increased from 5 to 30 and the dimensionality reduction step is removed. The anomaly score is computed in the same way as for AR-LSTM.
- *Autoencoder (AE).* A convolutional autoencoder featuring 6 ResNet blocks [7]. The anomaly score is the euclidean norm of the difference of reconstructed and real value.
- *kNN.* Past works show kNN as the best performing nearest neighbour based algorithm for anomaly detection, with anomaly score computed either as the average or the maximum distance from the neighbors [6]. We employ maximum distance with k=5, as it has the best compromise between accuracy and execution time.
- *Isolation Forest.* An ensemble of 100 individual decision trees, that isolate each data point into a leaf. The anomaly score of a data point is based on the average path length [15]. As recommended by [15], we use a contamination value of 0.1, which defines the proportion of outliers in the dataset.

3.4 Implementation details

All the models were implemented in TensorFlow 2.11.0 and Sklearn 1.1.2. For a fair comparison, all the anomaly detection frameworks were trained in the same experimental conditions and implementing hyperparameters tuning strategies [1]. More specifically: the neural network-based frameworks were optimized using Adam with a fixed 10^{-5} learning rate. GBRF and Isolation Forest were trained using the mean squared error criterion and recursive binary splitting, by strictly following the respective reference papers.

4 INDUSTRIAL CASE STUDY

4.1 Kuka anthropomorphic manipulator

To create a realistic scenario for the anomaly detection methods, we focused our case study on a KUKA LBR iiwa collaborative industrial robot, part of a fully-fledged production line[2]. The robot performs pick and place operations and it is controlled by a Simatic S7-1200 Programmable Logic Controller (PLC), directly connected to the robot through a hard-wired field bus. The PLC runs an OPC Unified Architecture (OPC UA) server, that exposes the KUKA state and functionality as services: the activation of such services in a given order constitutes a production process.

The KUKA robot allows collecting the robot's parameters through its programming interface. However, this limits the frequency with which such parameters can be collected to 5 Hz. At higher frequencies, queries interfere with the controlling process, causing stuttering in the robot trajectories. For this reason, we instrumented the KUKA robot with seven Inertial Measurement Unit (IMU) sensors (DFRobot SEN0386), one on each robot joint, to measure the joint's angle, acceleration, and angular velocity. These sensors send data at 200 Hz on a serial wire after applying a Kalman filter to reduce noise. In addition to physical data, we collected also extra-functional data from a single-phase energy meter (Eastron SDM230) monitoring the energy consumption of both the robot and the industrial PC. This energy meter is connected through a hard-wired Modbus with an industrial ESP-32 (Olimex ESP32-EVB), collecting and sending data to a MQTT broker via Ethernet.

Figure 2 depicts the experimental setup, consisting of: the KUKA robot, seven IMU sensors, an energy meter, and an embedded board connected to the sensors and executing the anomaly detection model (further described in Section 4.4).

4.2 Data stream characterization

The data stream collected from the robotic manipulator consist of 86 channels in total (reported in Table 1), including signals to monitor the action currently performed by the robot (*i.e.*, *action ID*), its kinematic behavior (*Joint Channels*) and its extra-functional parameters (*Power Channels*) [18].

The Joint Channels consist of data related to the seven joints collected from the IMUs sensors, each having the same eleven components that monitor various aspects of motion and temperature. Originally, the IMU collect angles in the $[-180, +180]$ $^{\circ}C$ range, causing high value changes when rotating near the two extremes. Since this may be a source of confusion for pattern recognition

[1]Source code and dataset available at https://gitlab.com/AlessioMascolini/varade
[2]Industrial Computer Engineering Laboratory - https://www.icelab.di.univr.it/

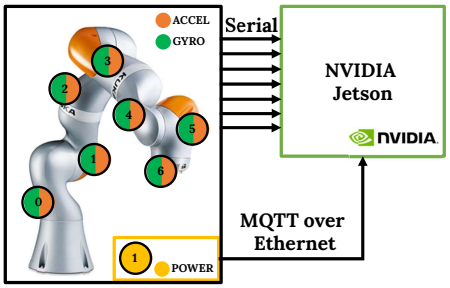

Figure 2: Case study setup. The KUKA manipulator is instrumented with different sensors: 7 accelerometers with six axes (one for each joint) and one single-phase power meter. The sensors are connected directly with an embedded board for detecting different classes of anomalies.

techniques, we had the orientations converted to quaternions, a 4-dimensional coordinate system commonly used in robotics.

The Power channels consist of eight quantities monitored by the energy meter. These channels allow detecting anomalies that could be transparent with respect to the robot trajectories, such as high power draw from a motor.

4.3 Experimental setup

To train VARADE and the baseline anomaly detection models, we created a dataset by recording the robot performing 30 unique actions (*i.e.*, its machine services) executed in a cycle for a total duration of 390 minutes. The resulting dataset contains all the possible actions supported by the robot, distributed uniformly within its duration. This allows the offline training of the anomaly detection models on the *"normal behavior"* of the robot in all the possible production processes supported by the manufacturing system. Given the diverse nature of the anomaly detection models, the collected

Table 1: Channels description: for each sensor considered, the related variables are listed. The <X> in the variable name is a label representing the [0,6] index of the joint in which the corresponding IMU sensor is placed on the robot.

	Channel name	Unit	Description
	action ID	-	Robot action ID
Joint Channels	sensor_id_X_AccX	m/s^2	X-axis acceleration
	sensor_id_X_AccY	m/s^2	Y-axis acceleration
	sensor_id_X_AccZ	m/s^2	Z-axis acceleration
	sensor_id_X_GyroX	deg/s	X-axis angular velocity
	sensor_id_X_GyroY	deg/s	Y-axis angular velocity
	sensor_id_X_GyroZ	deg/s	Z-axis angular velocity
	sensor_id_X_q1	-	Quaternion orient. comp. 1
	sensor_id_X_q2	-	Quaternion orient. comp. 2
	sensor_id_X_q3	-	Quaternion orient. comp. 3
	sensor_id_X_q4	-	Quaternion orient. comp. 4
	sensor_id_X_temp	°C	Temperature
Power Channels	current	A	Current
	frequency	Hz	Frequency
	phase_angle	degree	Phase angle
	power	W	Power
	power_factor	-	Power factor
	reactive_power	VAr	Reactive power
	voltage	V	Voltage

data are normalized in the range [-1, 1] based on the minimum and maximum values of each sensor's data, ensuring that all the features have equal importance avoiding unfair comparison.

To test the trained models in real-time conditions, we designed a "collision experiment" of 82 minutes in total. During this experiment, the robot performed all the 30 possible actions. During the robot operations, 125 collision anomalies were randomly generated by a human operator, by manually interfering with the robot during its movement in a very limited timeframe. This simulates sudden collisions between a human worker (or an object) and the robot, which is a realistic hazardous situation in a production line.

To test the suitability to an edge scenario, we selected two edge devices, connected to the robotic system depicted in Figure 2: a Nvidia Jetson Xavier NX (with 6 cores and 16 GB of RAM) and a Jetson AGX Orin (with 12 cores and 32 GB of RAM). Each anomaly detection model has been tested by a software script that continuously reads data from the sensors, prepares the data by applying a preprocessing function, and calls the inference function.

During each test, the anomaly detection accuracy was evaluated in terms of Area Under the Receiver Operating Characteristic Curve (AUC-ROC) value. The ratio is to interpret an anomaly detector as a binary classifier, where points are classified as anomalous if the anomaly score exceeds a certain threshold. The Receiver Operating Characteristic (ROC) curve plots the true positive rate against the false positive rate at varying values of this threshold, and the area under this curve provides a single threshold-less $[0, 1]$ measure of the algorithm's ability to identify the anomalous data points.

Besides the AUC-ROC score, we measured the inference frequency, and we collected all the most relevant board's metrics (*e.g.*, power consumption, RAM usage, GPU RAM usage) by exploiting the *jetson-stats* library. These metrics were collected not only during the execution of the anomaly detection tasks, but also with the boards in *Idle state* for 6 minutes: the mean value was computed as baseline to evaluate the load introduced by the anomaly detection models *w.r.t.* the standard state.

4.4 Experimental results

Table 2 reports the results obtained by VARADE and by the baseline detectors described in Section 3.3 on the two selected edge devices.

Jetson Xavier NX. VARADE placed first in terms of accuracy, with an AUC-ROC score of 0.84, with an improvement of 4% *w.r.t.* the second most performing model (AE) and of 13% *w.r.t.* the third most performing one (AR-LSTM). Interestingly, these improvements correspond also to a higher inference frequency, improved 7 times *w.r.t.* AE and 3 times *w.r.t.* AR-LSTM.

Considering inference frequency, VARADE placed second, with 15 Hz against 20 Hz obtained by GBRF. However, when considering the AUC-ROC scores, VARADE offers an improvement of almost 20% *w.r.t.* GBRF. Looking at RAM usage, we note that all the models use almost the same amount of memory, while VARADE uses a higher amount of GPU RAM (500 MB). This is not a limitation for the applicability of VARADE, as the total amount of memory used is under 40%, thus leaving enough space for larger anomaly models or other applications. Another important parameter for an edge device is the power required to operate, as the device could operate in conditions with limited power. Almost all the models have comparable performance in terms of power consumption, except AR-LSTM (for its high usage of the GPU), and kNN (for its high usage of CPU).

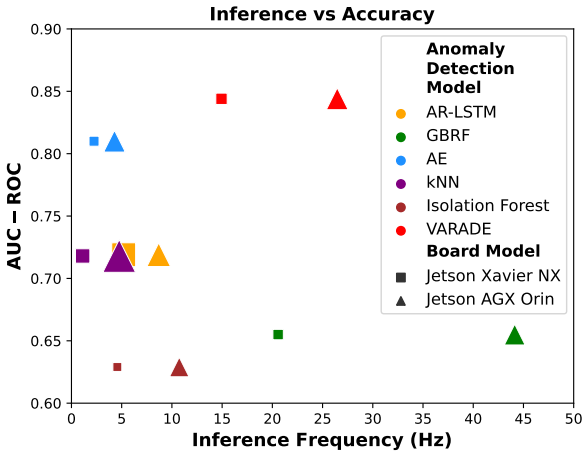

Figure 3: Inference Frequency vs Accuracy of the anomaly detection models (identified by the marker color). Marker shape represents the adopted edge device (square for Jetson Xavier NX, and triangle for Jetson AGX Orin), while marker size is proportional to power consumption.

Jetson AGX Orin. Analyzing the results obtained on the Jetson AGX Orin, we can note that the results are similar to the ones obtained with the Jetson Xavier NX but with a different scale. We can see that inference frequency is more or less doubled for all the anomaly detection models, but the overall ranking remains the same, with GBRF in the first position and VARADE in the second position. A significant difference relies in the GPU usage, as in this case the TensorFlow planner decided to run kNN and Isolation Forest on the CPU due to the higher number of CPU cores.

Figure 3 highlights the characteristics of the different configurations by depicting the ratio between the Inference Frequency and the AUC-ROC scores of the tested anomaly detection models.

Looking at the results obtained on the two boards, we can draw the following conclusions. The two anomaly detection models less suitable (in our case study) for anomaly detection on the edge are the kNN and the AR-LSTM. kNN is an algorithm that cannot fully benefit from GPU parallelism (especially with a few channels, as in our case study). On one side, this problem can be solved by exploiting CPU parallelism, but on the other side, edge devices have limited computation power, leading to high power draw and limited CPU available to run other jobs. AR-LSTM is based on a memory-intensive architecture that is not designed to work in a constrained environment with high throughput requirements. In fact, with both the boards, we can note a high GPU usage, which could seem a positive factor but leads in fact to low inference speed.

At the same time, we can note that VARADE (in red) shows the best accuracy without sacrificing too much performance on the inference speed, thus offering the best trade-off on both edge devices. This demonstrates its applicability on constrained devices such as the Jetson Xavier NX, as it offers a significant improvement in accuracy with a minimal loss in the inference speed, still meeting the resource constraints imposed by an edge device.

Table 2: Comparison between the Anomaly Detection models executed in real-time on the two edge processing units.

Board Model	Anomaly Detection Model	CPU Usage (%)	GPU Usage (%)	RAM Usage (MB)	GPU RAM Usage (MB)	Power Consumption (W)	AUC-ROC	Inference Frequency (Hz)
Jetson Xavier NX	Idle	36.465	52.100	5,130.219	537.235	5.851	.	.
	AR-LSTM	62.311	97.700	5,669.830	872.374	11.288	0.719	5.200
	GBRF	61.499	53.000	5,518.050	528.416	6.108	0.655	20.575
	AE	53.023	79.400	5,276.139	807.528	6.010	0.810	2.247
	kNN	92.547	55.700	5,076.605	526.844	7.208	0.718	1.116
	Isolation Forest	51.122	64.700	4,859.356	526.673	5.777	0.629	4.568
	VARADE	52.420	70.600	5,488.874	1,005.369	6.333	0.844	14.937
Jetson AGX Orin	Idle	4.875	0.000	3,916.715	243.289	7.522	.	.
	AR-LSTM	10.744	87.200	4,741.666	761.107	11.139	0.719	8.687
	GBRF	10.475	15.900	4,279.286	245.287	9.741	0.655	44.128
	AE	10.548	51.800	4,882.850	699.010	10.168	0.810	4.284
	kNN	91.506	0.000	4,201.195	243.289	16.887	0.718	4.754
	Isolation Forest	10.648	0.000	3,990.171	243.289	9.169	0.629	10.732
	VARADE	10.399	70.100	5,167.490	954.701	10.220	0.844	26.461

5 CONCLUSIONS AND FUTURE WORKS

In this research we introduced VARADE, a variational based autoregressive system, to address the challenges posed by real-time anomaly detection on the edge. When benchmarked against conventional algorithms, VARADE demonstrated superior performance, while maintaining a significantly higher inference speed compared to other anomaly detection techniques. This positions VARADE as a promising solution, especially in applications that can benefit from the ability to detect complex anomalies. Future works will include experimenting with a larger set of different use cases, to stress the flexibility of our method. Thus, we plan to integrate VARADE within the manufacturing control loop, enabling preventive anomaly detection to activate high-level reconfiguration strategies.

REFERENCES

[1] Abdelhadi Azzouni and Guy Pujolle. 2017. A Long Short-Term Memory Recurrent Neural Network Framework for Network Traffic Matrix Prediction. (2017). arXiv:1705.05690 [cs.NI]

[2] Lorenzo Bacchiani, Giuseppe De Palma, Luca Sciullo, Mario Bravetti, Marco Di Felice, Maurizio Gabbrielli, Gianluigi Zavattaro, and Roberto Della Penna. 2022. Low-Latency Anomaly Detection on the Edge-Cloud Continuum for Industry 4.0 Applications: the SEAWALL Case Study. *IEEE Internet of Things Magazine* 5, 3 (2022), 32–37. https://doi.org/10.1109/IOTM.001.2200120

[3] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. 2017. Variational inference: A review for statisticians. *Journal of the American statistical Association* 112, 518 (2017), 859–877.

[4] Andrew A Cook, Göksel Mısırlı, and Zhong Fan. 2019. Anomaly detection for IoT time-series data: A survey. *IEEE Internet of Things Journal* 7, 7 (2019), 6481–6494.

[5] Deepak Ghimire, Dayoung Kil, and Seong-heum Kim. 2022. A survey on efficient convolutional neural networks and hardware acceleration. *Electronics* 11, 6 (2022), 945.

[6] Markus Goldstein and Seiichi Uchida. 2016. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. *PLOS ONE* 11, 4 (04 2016), 1–31. https://doi.org/10.1371/journal.pone.0152173

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778. https://doi.org/10.1109/CVPR.2016.90

[8] Long Hu, Yiming Miao, Gaoxiang Wu, Mohammad Mehedi Hassan, and Iztok Humar. 2019. iRobot-Factory: An intelligent robot factory based on cognitive manufacturing and edge computing. *Future Generation Computer Systems* 90 (2019), 569–577. https://doi.org/10.1016/j.future.2018.08.006

[9] Huiyue Huang, Lei Yang, Yuanbin Wang, Xun Xu, and Yuqian Lu. 2021. Digital Twin-driven online anomaly detection for an automation system based on edge intelligence. *Journal of Manufacturing Systems* 59 (04 2021), 138–150. https://doi.org/10.1016/j.jmsy.2021.02.010

[10] Dohyung Kim, Hyochang Yang, Minki Chung, Sungzoon Cho, Huijung Kim, Minhee Kim, Kyungwon Kim, and Eunseok Kim. 2018. Squeezed convolutional variational autoencoder for unsupervised anomaly detection in edge device industrial internet of things. In *2018 International conference on information and computer technologies (icict)*. IEEE, 67–71.

[11] Kacper Kubiak, Grzegorz Dec, and Dorota Stadnicka. 2022. Possible Applications of Edge Computing in the Manufacturing Industry–Systematic Literature Review. *Sensors* 22, 7 (2022). https://doi.org/10.3390/s22072445

[12] Rohit Kumar and Neha Agrawal. 2023. Analysis of multi-dimensional Industrial IoT (IIoT) data in Edge-Fog-Cloud based architectural frameworks: A survey on current state and research challenges. *Journal of Industrial Information Integration* (2023), 100504.

[13] Gen Li and Jason J Jung. 2022. Deep learning for anomaly detection in multivariate time series: Approaches, applications, and challenges. *Information Fusion* (2022).

[14] Zheng Li, Mingxing Duan, Bin Xiao, and Shenghong Yang. 2023. A Novel Anomaly Detection Method for Digital Twin Data Using Deconvolution Operation With Attention Mechanism. *IEEE Transactions on Industrial Informatics* 19, 5 (May 2023), 7278–7286.

[15] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2012. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6, 1 (2012), 1–39.

[16] Yumeng Liu, Hongan Wang, Xu Zheng, and Ling Tian. 2023. An efficient framework for unsupervised anomaly detection over edge-assisted internet of things. *ACM Transactions on Sensor Networks* (2023).

[17] Tianyuan Lu, Lei Wang, and Xiaoyong Zhao. 2023. Review of Anomaly Detection Algorithms for Data Streams. *Applied Sciences* 13, 10 (2023), 6353.

[18] Alessio Mascolini, Sebastiano Gaiardelli, Francesco Ponzio, Nicola Dall'Ora, Enrico Macii, Sara Vinco, Santa Di Cataldo, and Franco Fummi. 2023. Robotic Arm Dataset (RoAD): A Dataset to Support the Design and Test of Machine Learning-Driven Anomaly Detection in a Production Line. In *IECON 2023-49th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 1–7.

[19] Garima Nain, KK Pattanaik, and GK Sharma. 2022. Towards edge computing in intelligent manufacturing: Past, present and future. *Journal of Manufacturing Systems* 62 (2022), 588–611.

[20] Tie Qiu, Jiancheng Chi, Xiaobo Zhou, Zhaolong Ning, Mohammed Atiquzzaman, and Dapeng Oliver Wu. 2020. Edge Computing in Industrial Internet of Things: Architecture, Advances and Challenges. *IEEE Communications Surveys & Tutorials* 22, 4 (2020), 2462–2488. https://doi.org/10.1109/COMST.2020.3009103

[21] Koushik Roy, Abtahi Ishmam, and Kazi Abu Taher. 2021. Demand Forecasting in Smart Grid Using Long Short-Term Memory. In *2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI)*. 1–5. https://doi.org/10.1109/ACMI53878.2021.9528277

[22] Haşim Sak, Andrew Senior, and Françoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Proc. Interspeech 2014*. 338–342. https://doi.org/10.21437/Interspeech.2014-80

[23] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. 2022. Anomaly detection in time series: a comprehensive evaluation. *Proceedings of the VLDB Endowment* 15, 9 (2022), 1779–1797.

[24] Yuting Sun, Tong Chen, Quoc Viet Hung Nguyen, and Hongzhi Yin. 2023. TinyAD: Memory-efficient anomaly detection for time series data in Industrial IoT. *IEEE Transactions on Industrial Informatics* (2023).

[25] Manu Suvarna, Ken Shaun Yap, Wentao Yang, Jun Li, Yen Ting Ng, and Xiaonan Wang. 2021. Cyber–Physical Production Systems for Data-Driven, Decentralized, and Secure Manufacturing—A Perspective. *Engineering* 7, 9 (2021), 1212–1223.

[26] Fabrizio De Vita, Giorgio Nocera, Dario Bruneo, and Sajal K. Das. 2023. A Novel Echo State Network Autoencoder for Anomaly Detection in Industrial IoT Systems. *IEEE Transactions on Industrial Informatics* 19, 8 (Aug. 2023), 8985–8994.

[27] Jingyu Yang and Zuogong Yue. 2023. Learning Hierarchical Spatial-Temporal Graph Representations for Robust Multivariate Industrial Anomaly Detection. *IEEE Transactions on Industrial Informatics* 19, 6 (2023), 7624–7635. https://doi.org/10.1109/TII.2022.3216006

[28] Xiang Yu, Xianfei Yang, Qingji Tan, Chun Shan, and Zhihan Lv. 2022. An edge computing based anomaly detection method in IoT industrial sustainability. *Applied Soft Computing* 128 (2022), 109486.

[29] Hao Zhou, Ke Yu, Xuan Zhang, Guanlin Wu, and Anis Yazidi. 2022. Contrastive autoencoder for anomaly detection in multivariate time series. *Information Sciences* 610 (2022), 266–280.