Robotic Arm Dataset (RoAD): a Dataset to Support the Design and Test of Machine Learning-driven Anomaly Detection in a Production Line

Robotic Arm Dataset (RoAD): a Dataset to Support the Design and Test of Machine Learning-driven Anomaly Detection in a Production Line

Alessio Mascolini*, Sebastiano Gaiardelli†, Francesco Ponzio‡, Nicola Dall'Ora†, Enrico Macii‡, Sara Vinco*, Santa Di Cataldo*, Franco Fummi†

*Dept. of Control and Computer Engineering, Polytechnic of Turin, Italy, name.surname@polito.it

†Dept. of Engineering for Innovation Medicine, University of Verona, Italy, name.surname@univr.it

‡Dept. of Regional and Urban Studies and Planning, Polytechnic of Turin, Italy, name.surname@polito.it

***Abstract*—The early detection of anomalous behaviors from a production line is a fundamental aspect of Industry 4.0, facilitated by the collection of massive amounts of data enabled by the Industrial Internet of Things. Nonetheless, the design and validation of anomaly detection algorithms, mostly based on sophisticated Machine Learning models, heavily rely on the availability of annotated datasets of realistic anomalies, which is very difficult to obtain in a real production line. To address this problem, we introduce the Robotic Arm Dataset (RoAD), specifically designed to support the development and validation of Multivariate Time Series Anomaly Detection (MTSAD) algorithms. We collect and annotate a large number of data and metadata to characterize the motion and energy consumption of a collaborative robotic arm in a full-fledged production line and annotate a comprehensive set of healthy as well as realistic anomalies scenarios. To prove the significance of RoAD and encourage future developments, we benchmark several state-of-the-art anomaly detection algorithms on our newly introduced dataset, and we freely release it to the scientific community.**

***Index Terms*—Data acquisition, Process monitoring, Flexible manufacturing systems, Anomaly detection**

I. Introduction

Industry 4.0 refers to the digital technologies that are designed to sense, predict, and interact with production systems, so as to make decisions that support productivity, energy efficiency, and sustainability. In this context, the monitoring and early detection of irregular behaviors in a production line has emerged as a critical aspect in the manufacturing industries, to avoid disruptions that may result in delays or even complete stoppage of production.

This process is facilitated by the Industrial Internet of Things (IIoT), which involves the interconnection of industrial devices, equipment, sensors, and systems through the Internet [1]. The generation and collection of massive amounts of diverse data enabled by IIoT, consisting of multiple interdependent variables rapidly evolving over time (a.k.a. multivariate time series), opens the way to the design of sophisticated *anomaly detection* algorithms able to model the intricate interdependencies of such data, with the objective of identifying any events that deviate significantly from the "normal" distribution. These techniques fall under the category of *Multivariate Time Series Anomaly Detection* (MTSAD) algorithms [2].

While there is a growing demand for MTSAD algorithms in industrial applications, the development and validation of these algorithms, mostly based on Machine Learning (ML), heavily rely on the availability of comprehensive datasets, representing diverse types of annotated anomalies in the data sensed from realistic scenarios.

Nonetheless, such datasets are very difficult to obtain in the context of a real production line, due to the many issues in the data collection and annotation process [3]. First of all, the very definition of anomaly is loose and context-dependent: it can involve either *point anomalies*, which are single out-of-range samples, or *collective anomalies*, where entire data streams have a distribution that is different from the expected one. But in both cases, what is considered "normal" can vary according to the specific context [4]. On top of that, anomalies are, by nature, rare and difficult to capture, especially when devices are relatively new and in healthy conditions. As a consequence, the effective design and testing of anomaly detection algorithms targeting industrial manufacturing systems remain a very open problem.

To this date, the most popular multivariate time series datasets used to train and test anomaly detection techniques typically target different types of applications. Among the others: Secure Water Treatment (SWaT) [5], Water Distribution Testbed (WADI) [6], Server Machine Dataset (SMD) [7], Mars Science Laboratory (MSL) [8], Soil Moisture Active Passive (SMAP) [9]. Some of these datasets, though valuable, lack in their ability to represent a wide range of anomaly scenarios. SWaT, WADI, and SMD have a limited scope, in that they focus entirely on point anomalies. MSL and SMAP provide examples of collective anomalies, but, on the other hand, they are known to have unlabeled anomalies in the training data, hampering the effective learning of the ML models [10].

This study was carried out within the PNRR research activities of the consortium iNEST (Interconnected North-Est Innovation Ecosystem) funded by the European Union Next-GenerationEU (Piano Nazionale di Ripresa e Resilienza (PNRR) – Missione 4 Componente 2, Investimento 1.5 – D.D. 1058 23/06/2022, ECS_00000043). This manuscript reflects only the Authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

In this paper, we introduce the Robotic Arm Dataset (RoAD), specifically designed to evaluate Multivariate Time Series Anomaly Detection (MTSAD) algorithms in an industrial manufacturing setting. As the name suggests, the specific target of the dataset is a collaborative anthropomorphic manipulator, a Kuka Lightweight Robot (LR) with seven Degree of Freedom (DoF), working in a full-fledged production line.

The specific contributions of RoAD are the following:
(i) a significant collection of data sensed by a large number of heterogeneous sensors, monitoring motion and energy consumption of the robotic arm during a variety of operations;
(ii) the integration of sensed data with synchronized metadata to fully characterize the performed actions (*i.e.*, pick, place) and corresponding parameters (*i.e.*, speed, object weight, etc.);
(iii) a realistic representation of different types of fully annotated anomalies, incorporating both point and collective anomalies as separate subsets.

As a secondary contribution, in this paper, we benchmark several state-of-the-art anomaly detection algorithms on the RoAD dataset with a two-fold objective. Firstly, to provide a fair comparison of such algorithms on common ground. Secondly, to prove the effectiveness and significance of our newly introduced dataset.

Both RoAD and the benchmarking protocol are freely released to the research community to encourage future development.

II. Monitoring of a production line

In this section, we first provide an overview of the manufacturing line used as a testbed. Then, we describe in detail both the physical structure (depicted in Figure 1) and the communication infrastructure (depicted in Figure 2) of the collaborative robotic arm exploited to construct RoAD.

A. Flexible Manufacturing Line

The collaborative robot used as a testbed is a working cell of the fully-fledged manufacturing line located in the Industrial Computer Engineering (ICE) Laboratory[1], a research facility of the University of Verona. The production line resembles a real flexible manufacturing system, enabling the study and development of research methodologies that can also be applied in real systems. As such, it does not implement a specific manufacturing process, but it can be reconfigured by changing the sequence of processing steps to be performed. It contains an automatic vertical warehouse, two 3D printers, a milling machine, a robotic cell with two collaborative robots, a quality control cell, and an automatic test equipment. A mini pallet conveyor belt system and two Automated Guided Vehicles (AGVs) are in place to transport materials between the working cells. Each machine is controlled by a Programmable Logic Controller (PLC) or by an industrial PC, implementing an OPC Unified Architecture (OPC UA) server standardizing the machine interaction and exposing the machine state variables and services within the laboratory, following the Service Oriented Manufacturing (SOM) paradigm.

[1]The Industrial Computer Engineering Lab: https://www.icelab.di.univr.it/

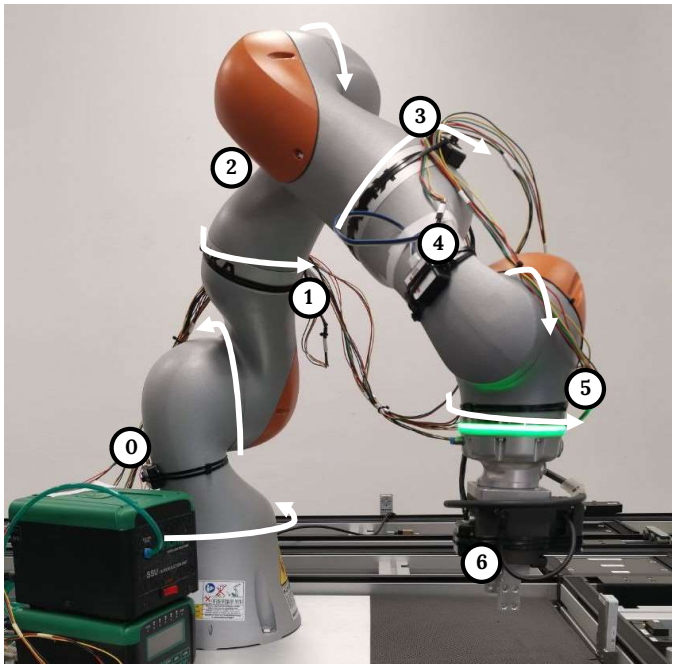

Figure 1: KUKA LBR iiwa setup inside the ICE Research Laboratory. The manipulator is instrumented with different sensors: 7 accelerometers with 6 axes (one for each joint) and one single-phase power meter.

The RoAD dataset is focused on the KUKA LBR iiwa robot, one of the two collaborative robots within the robotic cell (Figure 1). The system includes: a collaborative robotic manipulator with 7 DoF with redundant joints (Joint 3 in Figure 1), a smart gripper with a suction valve, a KUKA Sunrise Cabinet robot controller, a KUKA smartpad control panel, and the KUKA software stack that is capable of receiving the commands and controlling the robot trajectories.

B. KUKA Communication Infrastructure

The hardware components of the KUKA LBR iiwa and the IIoT communication infrastructure are depicted in Figure 2. An industrial PC, directly connected to the robot through a hard-wired field bus (top-left of Figure 2) controls the robot translating high-level trajectories to low-level movements. The official KUKA programming interface (KUKA Sunrise software stack [11]) is installed on top of the industrial PC to define such trajectories and to program the KUKA. KUKA Sunrise allows also to define background tasks always running, such as collision detection and logging tasks; the data produced by such tasks are published directly to an MQTT broker. The industrial PC is connected through a Profibus BUS with a Simatic S7-1200 PLC (bottom-left of Figure 2), which reads from the robot its state variables and sends start and stop commands. It also implements a OPC UA server exposing the robot's functionalities (*i.e.*, programs) as OPC UA methods for external components.

Through the KUKA Sunrise programming stack, it is possible to collect only a subset of the robot's parameters (either

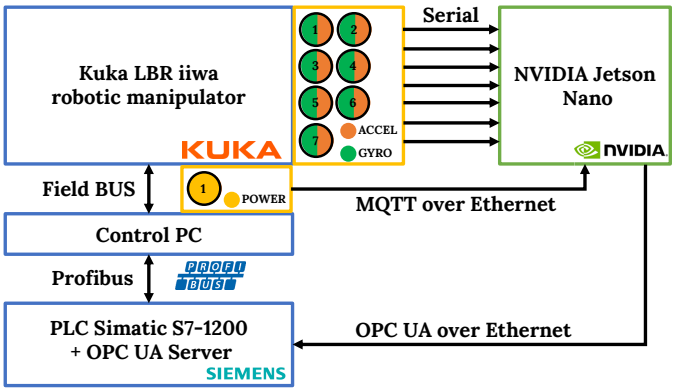

Figure 2: Block description diagram the data collection setup.

already collected by the robot or additional ones collected by sending a query message on the field bus) with limited frequency. To avoid interfering with the control commands sent over the field bus, which could cause interruptions of the robot's movements, the robot has been instrumented with additional sensors collecting extra-functional parameters. A single-phase energy meter (Eastron SDM230) monitors the energetic consumption of both the robot and the industrial PC. This energy meter is connected through hard-wired Modbus with an industrial ESP-32 (Olimex ESP32-EVB), collecting and sending data to the MQTT broker via Ethernet.

Each robot's joint's angle, angular velocity, and acceleration are collected by a series of Inertial Measurement Unit (IMU) sensors (DFRobot SEN0386) placed on each joint. The DFRobot SEN0385 module integrates a high-precision gyroscope, accelerometer, and microprocessor. Before being published, the raw data collected by the IMU are filtered by a Kalman filter to reduce the measurement noise. These sensors are configured to send data at a constant frequency (200 Hz) on a hard-wired serial.

The software that has been exploited for collecting data from both the robot and its external sensors, as well as for sending commands used to construct RoAD, runs on an Nvidia Jetson Nano directly connected to the sensors and the network with wired connections (top-right of Figure 2). Each IMU sensor is connected directly to the Nvidia Jetson Nano through a USB cable based on the chip CP2102. Data collected are buffered and written in real-time on an external memory unit into a compressed data structure to enable reading data at the maximum frequency.

III. Anomaly Scenarios

The main target of RoAD is to provide a dataset with a comprehensive set of realistic anomalies that can occur during the normal operating conditions of the robotic arm, encompassing both *point anomalies* and *collective anomalies* within various experimental scenarios. As such, we first identified a set of robotic actions to monitor, and after, we generated and annotated a set of anomalies for the chosen actions.

The Kuka Robot is programmed to move materials from a pallet in front of its working area and its working buffer. Both the pallet and the robot's buffer have different positions that can be used for placing materials identified with different integer numbers. Specifically, we considered the following actions (fully described in Table I):

- `pickFromPallet(posPlt,posBuf)`: this action picks a material from the position `posPlt` of the pallet and places it in the position `posBuf` of the robot's buffer;
- `placeToPallet(posBuf,posPlt)`: this action picks a material from the position `posBuf` of the robot's buffer and places it in the position `posPlt` of the pallet;
- `moveOverPallet(posPlt1,posPlt2)`: this action picks a material from the position `posPlt1` of the pallet and places it in the position `posPlt2` of the pallet.

We configured the robot to lift a maximum payload of 500 g. By changing the parameters described above, we defined a set of operations moving a piece of LEGO DUPLO, with an average weight of 20 g, from the pallet to the buffer, from the buffer to the pallet, and over the pallet, for a total of 30 unique actions. Then, we collected data from the robot, while introducing a number of controlled anomalies into the process. More specifically, we created three types of scenarios, exemplified in Figure 3:

(a) *Collision recordings.* The recordings under this category contain *collision events* created by manually interfering with the action of the robot during its movement in a very limited time frame. This simulates sudden collisions between a human worker and a robot, which is a very hazardous situation in a production line. The so-obtained anomalies, characterized by abrupt and isolated changes in the recorded data, fall under the category of *point anomalies*, which are the most commonly represented ones in the MTSAD benchmarking datasets. A portion of a collision recording is shown in Figure 3(a), together with the anomaly label associated with each sample. As can be seen from the plot, the anomaly variable is set to 1 (meaning: one anomaly present) in correspondence to a collision event, and 0 for the rest of the samples.

(b) *Weight recordings.* These recordings are acquired after adding an additional weight of 750 g to the robot's end effector, without changing the maximum programmed payload of 500 g. By doing so, we simulate systematic issues or long-term trends that need to be addressed. An example is reported in Figure 3(b), where the added weight determines acceleration values different than expected for the whole duration of the recording, which falls under the definition of *collective anomaly*. As a consequence, in this case, the anomaly label is set to 1 (meaning: one anomaly present) for all the samples. Since they represent slow and steady changes in the data, collective anomalies may pose a more difficult challenge for detection algorithms compared to point anomalies.

(c) *Velocity recordings.* The recordings under this category

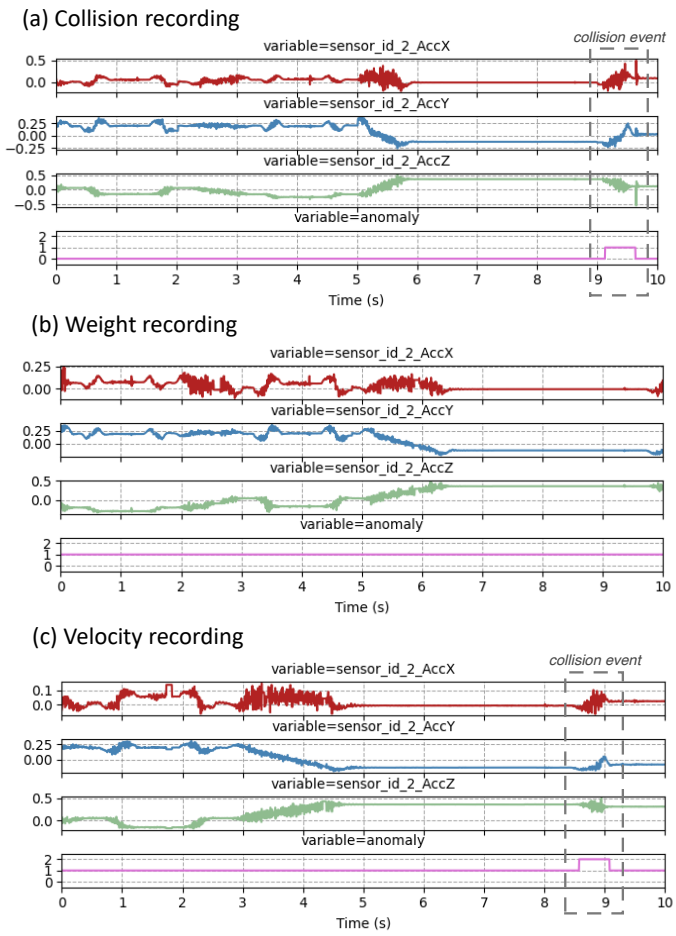

Figure 3: Examples of recordings from the accelerometers of node 2 with different types of controlled anomalies, and corresponding anomaly labels.

represent the most complex scenario of anomalies in RoAD. The anomalies are generated by changing the trajectory speed of the robot, respectively, to 50% and 75% of the standard velocity. On top of that, the acquisitions also feature sudden collision events, thus creating a mixed scenario in which purely collective anomalies may occasionally overlap with point anomalies. An example is shown in Figure 3(c): in this case, the underlying collective anomaly determined by the speed change is represented by the anomaly label 1, and the occasional occurrence of a collision event is reflected by the anomaly label suddenly changing from 1 to 2 (meaning: two anomalies present at the same time).

The number and duration of the recordings falling under the *collision*, *weight*, and *velocity* categories are reported in Table III. Details about the other categories will follow.

IV. Data processing

After recording, we processed the dataset with the aim of ensuring that it is suitable for usage with ML models. After resampling the recordings to 10 Hz, we integrated sensor measurements with information about the specific action performed by the robot, and we added the anomaly variable indicating the presence of one or more anomalies, which can serve as ground truth annotation for training and evaluation purposes.

We have converted the orientations, originally recorded as a set of three Euler angles per node to unit quaternions, a coordinate system consisting of 4-dimensional vectors commonly used in robotics. Since the originally recorded orientation is limited to +-180° and movements beyond this limit wrap around to the opposite side of the available range, this approach prevents orientation values from suddenly switching between the previously mentioned extremes when the orientation of a node is near 180°, which may be a source of confusion for pattern recognition techniques.

In detail, the published data consist of a total of 86 channels. As represented in Tables I and II, these channels include:

- An ID in the [0,30] range, indicating the current action of the robot;
- Information about supplied power, current, and voltage;
- Orientation, rotational velocity, acceleration, and temperature for each joint;
- An anomaly label indicating the number of co-occurring anomalies at a given time (either 0, 1, or 2, as explained in Section III).

V. Benchmarking of anomaly detectors

This section presents the results of benchmarking various anomaly detection algorithms on the newly introduced RoAD dataset. On the one hand, the primary objective of this exercise is to exemplify the application of existing algorithms and establish their performance in identifying the anomalies represented in our dataset. On the other hand, the experiments help demonstrate the significance of RoAD as a benchmark for designing and validating new anomaly detection algorithms.

A. Models

To achieve our purpose, we have selected a diverse set of algorithms from the literature, ranging from classical to state-of-the-art deep learning techniques, in five different groups:

1) *Traditional outlier detection* methods, where anomalies are identified based on their higher distance or dissimilarity from regular observations in the feature space. In this category, we tested two classical approaches based on *kNN algorithm* [12] and on *Isolation Forest* [13], respectively.
2) *Reconstruction-based methods*, which learn to compress the data into a latent representation and then reconstruct it, identifying anomalies by evaluating the reconstruction error. In this group, we tested different techniques for the reconstruction: *Autoencoder (AE)* [14], *Variational Autoencoder (VAE)* [15] and *Vector Quantized Variational Autoencoder (VQ-VAE)* [16]. All such techniques were implemented with transformer backbones [17], which in our preliminary testing, were found to have better

Table I: Dataset metadata description: each action performed by the robotic manipulator is identified by a unique ID varying from 0 to 30.

Action	Parameters		ID
Idle	-		0
pickFromPallet(posPlt,posBuf)	posPlt	posBuf	-
	2	1	1
	1	1	5
	3	1	9
	2	2	13
	1	2	15
	3	2	17
	2	3	19
	1	3	21
	3	3	23
	2	4	25
	1	4	27
	3	4	29
placeToPallet(posBuf,posPlt)	posBuf	posPlt	-
	1	2	2
	1	1	6
	1	3	10
	2	2	14
	2	1	16
	2	3	18
	3	2	20
	3	1	22
	3	3	24
	4	2	26
	4	1	28
	4	3	30
moveOverPallet(posPlt1,posPlt2)	posPlt	posPlt	-
	2	1	3
	1	2	4
	1	3	7
	3	1	8
	3	2	11
	2	3	12

performance than convolutional-based ones. This is likely due to their ability to capture long-range dependencies in the data, which is a critical aspect of modeling time series data with periodic components. On top of that, we performed experiments with *Omnianomaly* [7], a recent approach leveraging a combination of stochastic recurrent neural networks and VAE.

3) *Hybrid reconstruction-based methods*, where the anomalies are detected by a combination of strategies besides reconstruction error. In this category, we did experiments with *MAD-GAN* [18], where a Generative Adversarial Network is trained on the data, and the final anomaly score is a combination of the discriminator score and the reconstruction score obtained by encoding the data in the latent generator space.
4) *Structure learning methods*, where a model tries to explicitly learn the structure of existing relationships between variables, identifying anomalies when the learned relationships are not verified. In this category, we tested *Graph Deviation Network (GDN)* [19], a recent technique exploiting graph neural networks with attention weights to provide explainability for the detected anomalies.

Table II: Dataset variables description: for each sensor considered, the related variables are listed. The <X> in the variable name is a [0,6] label representing the joint where the corresponding IMU sensor is placed on the robotic manipulator.

Dataset Variable	Unit	Description
action	-	Robot action ID
apparent_power	VA	Apparent power
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
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 orientation comp. 1
sensor_id_<X>_q2	-	Quaternion orientation comp. 2
sensor_id_<X>_q3	-	Quaternion orientation comp. 3
sensor_id_<X>_q4	-	Quaternion orientation comp. 4
sensor_id_<X>_temp	°C	Temperature
anomaly	-	Anomaly label

5) *Sanity-check methods*. Besides testing state-of-the-art techniques, we performed two extra experiments as sanity checks to demonstrate the validity of the problem posed by our newly introduced dataset. On the one hand, we want to verify that the anomalies are not too trivial while providing a baseline that all methods have to beat. To do so, we employ the *naive algorithm* used by Siwon et al. [20], which consists of a simple method that models the relationship between data and anomaly score as the identity function. On the other hand, we want to verify that identifying the anomalies is possible. We achieve this by employing a supervised *logistic regression* classifier trained on the first half of the test data and evaluated on the latter half. The ratio is: if the classifier has an acceptable accuracy, the posed anomaly detection problem is, in theory, solvable.

All the models implemented from scratch by us (AE, VAE, and VQ-VAE) have approximately 1 million parameters. This ensures that they all require a similar amount of time to train. For the other methods (kNN, Isolation Forest, MAD-GAN, Omnianomaly, and GDN), we have used their default hyperparameters provided by the original paper without any further optimization. We believe this provides a fair evaluation of the performance of these algorithms out of the box.

B. Benchmarking methodology

In designing the benchmarking process, we carefully considered the various aspects of evaluating anomaly detection algorithms, with special regard to the challenges posed by RoAD.

1) Normalization: Given the diverse nature of the algorithms being benchmarked, it is essential to ensure that the

input data is preprocessed in a consistent manner across all algorithms. In this regard, we have chosen to apply min-max normalization fitted on the training data, which scales the feature values to a specific range (in our case [-1, 1]) and ensures that all features have equal importance in the computation of distances and similarity measures. This normalization technique is compatible with all the algorithms included in our benchmark and ensures a fair comparison of their performance.

2) Metrics: A critical aspect of evaluating anomaly detection algorithms is the choice of performance metrics. Traditional approaches in literature have relied on the F1 score, calculated using an empirical threshold. However, this approach is not ideal for several reasons.

Using data from the test set to optimize a threshold that separates anomalous from non-anomalous data introduces a potential bias in the evaluation process, as it allows the algorithm to gain knowledge about the distribution of anomalies implicitly. The authors of GDN [19] recognize this problem and try to correct it by designating a validation set on which to tune the threshold, but this is still a problematic approach, as it goes contrary to the principle of anomaly detection as an unsupervised learning task. Since the anomaly detection framework aims to identify any event that is out of the expected distribution of data, the threshold, if any, should be ideally set only based on the characteristics of the non-anomalous data. This is more representative of the real-world conditions in which these algorithms would be employed, where a sizeable and varied dataset of anomalies is either non-existent or difficult to obtain, and there may be no prior knowledge of the anomalies.

To address these concerns and evaluate the benchmarked algorithms' performance, we calculate the area under the Receiver Operating Characteristics (ROC) curve, a threshold-less metric well-suited for comparing algorithms in different experimental conditions. The ROC curve plots the true positive rate against the false positive rate at varying threshold values, and the area under the curve (AUC) provides a single $[0, 1]$ measure of the algorithm's ability to identify the anomalous data points, which in this case represent the positive class. The higher the AUC, the better the algorithm, irrespective of the threshold value.

3) Subsets: We have designated a number of distinct subsets to evaluate the performance of the anomaly detection algorithms. These subsets serve a specific purpose in the evaluation process, as detailed below.

- *Training Set.* This subset is used to train the anomaly detection algorithms, allowing them to learn the patterns and characteristics of the non-anomalous data. The training set, as the name suggests, does not contain any anomalies and serves as the foundation for the algorithms to build their understanding of what constitutes a "normal" data point.
- *Test Sets.* These three sets containing anomalies are used to assess the performance of the anomaly detection algorithms in terms of obtained AUC-ROC scores. They are three, one per each of the anomaly scenarios described in Section III: *Collision*, *Weight*, and *Velocity*.
- *Control Set.* As already mentioned in Section III, collisions are *point anomalies* in a data stream of regular samples, which makes the computation of false positive rates (i.e., regular samples wrongly classified as anomalies) from which to derive AUC-ROC straightforward. Conversely, the Weight and Velocity anomalies involve *collective anomalies*, where an entire data stream constitutes an anomaly. Thus, to be able to compute AUC-ROC in such cases, we define an additional *Control Set* of non-anomalous data. To avoid any bias in the validation, these samples are completely independent of the ones of the Training Set.

Table III reports the number and duration of the recordings in each described subset.

Table III: Subsets characterization

Subset	**Recordings**	**Duration (min)**
Training Set	9	389.65
Collision Test Set	2	81.92
Weight Test Set	1	31.02
Velocity Test Set	2	69.23
Control Set	1	124.17

C. Results and Discussion

The results of our experiments, in terms of the AUC-ROC scores obtained by the five groups of models, described in Section V-A, are presented in Table IV.

Table IV: AUC-ROC scores of the anomaly detection algorithms benchmarked on RoAD

	Model	**Collision**	**Weight**	**Velocity**
1)	kNN	0.62	0.47	0.54
	Isolation Forest	0.65	0.74	0.74
2)	AE	0.51	0.39	0.37
	VAE	0.52	0.20	0.23
	VQ-VAE	0.56	0.19	0.17
	OmniAnomaly	0.67	0.70	0.63
3)	MAD-GAN	0.49	0.50	0.52
4)	GDN	0.68	0.62	0.67
5)	Naive	0.55	0.36	0.27
	Logistic Regression	0.92	0.99	0.90

From the results, we can draw several interesting observations. Firstly, it is evident that the naive algorithm provides a relatively low baseline, confirming that the RoAD dataset poses a challenging problem for anomaly detection. The logistic regression classifier, on the other hand, demonstrates that the anomalies in the dataset are indeed well-separable from the non-anomalous data, as indicated by its high AUC-ROC score in all three subsets. Hence, the anomaly detection problem posed by the RoAD dataset is non-trivial but yet not impossible to solve.

The performance of the benchmarked algorithms has significant variability when applied to RoAD. One of the standout findings is the consistently good performance of Isolation Forests. The algorithm demonstrates among the highest AUC-ROC scores in all subsets, often outpacing even deep learning-based methods.

Among the other tested algorithms, two that stand out are GDN and Omnianomaly. These algorithms beat traditional techniques in the detection of point anomalies, corroborating similar performance outcomes observed in previous studies.

Autoencoders learn to reconstruct the input data by minimizing the reconstruction error during training. Ideally, this should result in higher reconstruction errors for anomalous data points compared to non-anomalous ones, allowing the algorithm to identify anomalies effectively. Yet the lower-than-random performance of autoencoders on the weights and velocity subsets highlights the limitations of pure reconstruction-based methods for anomaly detection. The fact that the autoencoder achieves lower reconstruction scores for the non-anomalous data, despite not being part of the training set, suggests that these data are easier to model due to the lower moving speed of the robot arm, leading to lower reconstruction errors compared to the control data.

VI. Data availability

RoAD, as well as the Python code implementing the normalization methodology described in Section V-B, are freely available at https://gitlab.com/AlessioMascolini/roaddataset/. To ease the ML developers, the dataset is a pip-installable package that can be directly integrated into a Python project.

Besides the ML-ready dataset, which is specifically intended for the validation of anomaly detection techniques, the same acquisitions are available on Zenodo (further details in the repository) at the original sample rate of 200 Hz. This may help in applications that typically benefit from high-frequency information, such as simulation and trajectory planning.

VII. Conclusions

In this paper, we introduced RoAD, a dataset specifically designed to support the development and validation of Multivariate Time Series Anomaly Detection (MTSAD) algorithms in an industrial manufacturing setting. For this purpose, RoAD collects a large set of heterogeneous data from a robotic arm and represents an exhaustive and realistic set of different types of anomalies, fully annotated to serve as the ground truth for the validation.

Besides characterizing RoAD, we benchmarked several existing anomaly detection algorithms on it, including classic and state-of-the-art ML models. Even the best-performing algorithms in our benchmarking study do not achieve very high AUC scores. This suggests that our dataset poses a complex and challenging problem that cannot be fully addressed by the existing approaches, providing ample opportunity for the development of more sophisticated and advanced anomaly detection algorithms.

References

[1] E. Sisinni, A. Saifullah, S. Han *et al.*, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE transactions on industrial informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[2] M. A. Belay, S. S. Blakseth, A. Rasheed *et al.*, "Unsupervised anomaly detection for iot-based multivariate time series: Existing solutions, performance analysis and future directions," *Sensors*, vol. 23, no. 5, p. 2844, 2023.

[3] Z. Liu, J. Zhang, X. He *et al.*, "Fault diagnosis of rotating machinery with limited expert interaction: A multicriteria active learning approach based on broad learning system," *IEEE Transactions on Control Systems Technology*, vol. 31, no. 2, pp. 953–960, 2023.

[4] E. Marcelli, T. Barbariol, V. Savarino *et al.*, "A revised isolation forest procedure for anomaly detection with high number of data points," in *2022 IEEE 23rd Latin American Test Symposium (LATS)*, 2022, pp. 1–5.

[5] J. Goh, S. Adepu, K. Junejo *et al.*, "A dataset to support research in the design of secure water treatment systems," in *Critical Information Infrastructures Security*. Springer International Publishing, 2017, pp. 88–99.

[6] C. Ahmed, V. Palleti, and A. Mathur, "Wadi: a water distribution testbed for research in the design of secure cyber physical systems," in *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*. ACM, 04 2017, pp. 25–28.

[7] Y. Su, Y. Zhao, C. Niu *et al.*, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2828–2837.

[8] NASA. Msl rover environmental monitoring station experiment data record. [Online]. Available: https://pds.nasa.gov/ds-view/pds/viewDataset.jsp?dsid=MSL-M-REMS-2-EDR-V1.0

[9] ——. Soil moisture active passive. [Online]. Available: https://smap.jpl.nasa.gov/data/

[10] E. D. Santis, F. Arnò, A. Martino *et al.*, "A statistical framework for labeling unlabelled data: a case study on anomaly detection in pressurization systems for high-speed railway trains," in *2022 International Joint Conference on Neural Networks (IJCNN)*, 2022, pp. 1–8.

[11] M. Safeea and P. Neto, "Kuka sunrise toolbox: Interfacing collaborative robots with matlab," *IEEE Robotics & Automation Magazine*, vol. 26, no. 1, pp. 91–96, 2019.

[12] M. Teng, "Anomaly detection on time series," in *2010 IEEE International Conference on Progress in Informatics and Computing*, vol. 1. IEEE, 2010, pp. 603–608.

[13] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422.

[14] D. H. Ballard, "Modular learning in neural networks," in *Proceedings of the sixth National conference on Artificial intelligence-Volume 1*, 1987, pp. 279–284.

[15] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[16] A. Van Den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," *Advances in neural information processing systems*, vol. 30, 2017.

[17] A. Vaswani, N. Shazeer, N. Parmar *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[18] D. Li, D. Chen, B. Jin *et al.*, "Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks," in *Artificial Neural Networks and Machine Learning–ICANN 2019: Text and Time Series: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings, Part IV*. Springer, 2019, pp. 703–716.

[19] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 5, 2021, pp. 4027–4035.

[20] S. Kim, K. Choi, H.-S. Choi *et al.*, "Towards a rigorous evaluation of time-series anomaly detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 7, 2022, pp. 7194–7201.