Methods for Monitoring and Detecting Faults in IoT Dosimetry Instrumentation Based on Machine Learning on Edge Computing Devices

Dora Pavelić
University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
dora.pavelic2@fer.hr

Luka Pavelić, Branko Petrinec, Ivica Prlić
Institute for Medical Research and Occupational Health
Ksaverska c. 2, 10001 Zagreb, Croatia

ABSTRACT

Due to the importance of safety, reliability and efficiency of dosimetry instrumentation, as well as increasing complexity of the technologies, we are proposing a method for early failure detection that could enable the necessary prompt response. IoT dosimetry sensors are usually required to operate for several years on a single battery and they are often installed in large numbers which place high energy and cost constraints. Therefore, the analysis and prediction itself is increasingly performed on devices that are close to the sensors. The concept of bringing analytical computational resources closer to the sensors themselves is called edge computing. In this work we will consider the application of machine learning for the purpose of fault detection in IoT dosimetry instrumentation as well as the various approaches with which these detections are realized with the help of edge computing devices.

Keywords: *Radiation Dosimetry, Machine Learning, Fault Detection*

1 INTRODUCTION

Predictive analysis is one of the key innovations proposed by Industry 4.0. The Industrial Internet of Things (IIoT), a subset of the Internet of Things, connects large number of industrial devices, sensors, and actuators. IIoT also increasingly includes artificial intelligence technology that enables more efficient and accurate process analysis such as predictive analysis that includes fault detection, predictive maintenance, demand prediction [1]. Dosimeters are used in area monitoring and in various occupations such as radiology, oncology, nuclear medicine, nuclear power, etc. It is of utmost importance for the safety of the workers and general public for dosimeters to work properly and to alert when there is a possibility that the fault is likely to happen or has already happen.

IIoT coupled with machine learning algorithms requires appropriate computing nodes that enable fast response, data security, and high level of personalization. Edge computing offers distributed computing services through smaller, local data centres at the edge of the network. Unlike cloud computing, edge computing allows lower bandwidth usage, lower latency and data security. Data is no longer transferred to remote servers but stored locally within the industrial environment. Advances in technology also enable more advanced methods of monitoring dynamic industrial systems. .The aim of this paper is to consider application od machine learning based fault detection in dosimetry IoT using edge computing. This paper is divided into several sections, in section 2

some of the most frequently used methods for predictive analysis are briefly described. Within the 3rd section, we introduce the concept of computing at the edge of the network and its role in the fault detection process. Section 4 presents the possible application od machine learning in IoT dosimetry on edge computing devices.

2 METHODS USED FOR FAULT DETECTION

The term failure in this paper refers to any of the possible three states of the device [2]:

i. The condition of the device in which the device deviates in at least one feature from its original, intended configuration.
ii. The state of the device in which all functions are disabled and thus the operation of the device itself.
iii. An occasional irregularity that disrupts the intended functions of the device.

Fault detection can generally be reduced to two main steps. The first step involves considering the signal generated at the input and output of the observed system. This signal is called residual. If the system is faulty, the residual should be zero or close to zero, otherwise the residual should be greater than zero. The residual can be a scalar that contains information about a single fault point or a vector that contains information about multiple fault points. The residual generation itself can range from a precisely defined mathematical model to a "black box" model. More on this will be in the following sections. The second step includes analysis of the residual signal. This step includes various decision logic that will determine whether the fault is likely to exist in the observed system.

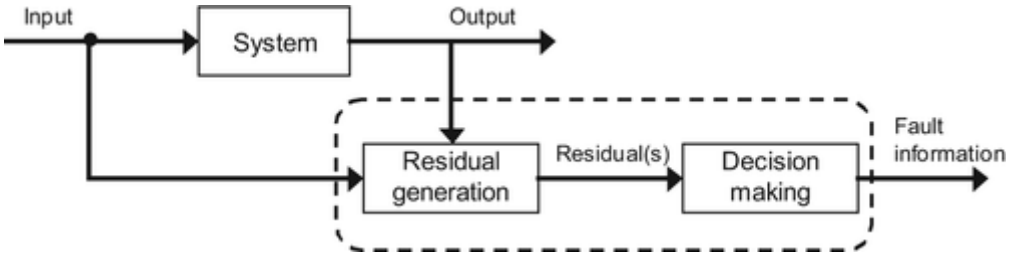

Figure 1: General scheme of model-based fault detection.

2.1 Methods classification

Depending on the type of failure, the amount and type of data we have available, and the complexity of the system itself, we can use different methods for detecting failure. In the literature on dynamic systems, fault detection is realized with exclusively analytical methods all the way to methods that use machine learning [3]. Some realizations require a good knowledge of the whole system and interdependence of the components on a physical level while some like machine learning require historical data of the considered system. In practice, these methods are often combined.

2.2 Model based methods

Model based methods make use of knowledge of systems at their more physical, technical level. They can be qualitative or quantitative depending on the way the input-output relationship is modelled. Quantitative models are represented by mathematical functions (see Figure 1) while qualitative methods are linked to a series of system representations using e.g., digraphs, fault trees

or fuzzy logic. The advantage of these approaches is the explainability of the results, which is not the case in some other methods.

Fault trees graphically show the connection between the cause of the system failure and the fault itself, where the causes are interconnected with the logic gate. Fault analysis is a deductive process from which the goal is to get from the state of the fault to its most probable cause. This approach allows a detailed view of the entire system architecture which allows everyone involved in the process a better understanding of the system. The analysis can identify combinations of the most likely causes that can be further used for optimization purposes. In order to properly build a fault tree, it is necessary to have knowledge about the whole system and how each system component affects the other. In practice, the final fault is often incorrectly defined, which further leads to wrong decisions during fault correction [4]. Furthermore, the fault tree defines the states of individual components as a binary value (faulty or non-faulty) while in practice the components may be of varying degrees of faultiness or degradation. The previous draws on the fact that it is not possible to define all possible faults and causes of faults with a fault tree. The degree of coverage of cases also largely depends on the expertise of the experts designing the fault tree.

Fuzzy logic can be used in cases where processes cannot be expressed by mathematical functions. Unlike a fault tree, the state of components can be represented with the degree of belongingness to the faultiness. With the help of fuzzy logic, it is possible to define complex systems for which we can further define fuzzy rules that will be used in diagnostic logic. In order to define fuzzy rules, it is necessary to quantize a signal reading into a limited set of fuzzy values.

Fuzzy logic can be used as an additional filter during the processing of a signal, where the noise level affects the degree of belongingness of the initial reading[5]. Eissa et al. in [6] use fuzzy logic to upgrade observer functionality where the observer is a simulated system that models a physical system to evaluate its internal state.

Analytical redundancy is a quantitative method that defines the mathematical model of the system. Using a comparison of the output of the real system and the modelled one we can derive the residual. The residual signal is subsequently analysed by an arbitrarily defined diagnostic logic that determine further action such as raising the alarm. If the mathematical model is not properly modelled or if there is additional interference that is not included in the analysis, frequent false alarms can occur.

2.3 Hardware based methods

Hardware-based fault methods do not deploy mathematical models of the system but use external information obtained from sensors and predefined anomalies to explicitly detect the fault.

Limit checking is a trivial technique in which we check whether the values are within limits. If the values are above defined limits, we can consider a system partially or entirely faulty. This approach is sufficient for trivial systems but if we want early fault detection this approach by itself is insufficient.

Hardware redundancy and voting techniques are methods that use redundant hardware, this is achieved by adding redundant sensors that are used for cross-checking. These approaches also lead to increased equipment and maintenance costs and are therefore not feasible in some cases.

Although approaches used in this method are simpler than model-based ones, they are still best suited for some less demanding systems. Of course, the approaches above are often combined for the purpose of yielding better results. Holbert et al. [7] combined fuzzy logic and hardware redundancy for fault detection of multiple pressure reactor sensors.

2.4 Methods based on historical data

Methods based on historical data generate a mathematical model of the observed system based on historical data. These methods use techniques such as expert systems, neural networks,

including self-organizing maps, statistical methods, and fuzzy logic, which in this division as an auxiliary task must generate a fuzzy model based on historical data without expert knowledge.

Due to the large number of techniques, it would be unreasonable to go through each approach and describe the advantages and disadvantages. The following are a brief description of some of the most used machine learning approaches.

Machine learning algorithms at the upmost level can be divided in three groups, supervised, unsupervised and reinforcement learning. Supervised learning algorithms use a known set of input and output data, unsupervised uses only input data with the goal of finding some patterns in the data that could group the data. Reinforcement learning uses terms such as “agent” and “environment”. The goal of reinforcement algorithms is to collect cumulative rewards (agent) through environmental search and the use of past knowledge. Supervised and unsupervised algorithms are the most popular approaches, but in cases where data is lacking, they are not of much use. Reinforcement learning, although still in development, has the potential to help in such cases.

Clustering is a technique that belongs to the branch of unsupervised learning, which can be a good choice if we work with a large set of data for which we do not have labelled outputs. Some of the better-known algorithms included here are K-means, C-means, and hierarchical clustering. The objective is to find naturally formed clusters of available data that will represent the behaviour of the whole system in such a cluster. The group with the given input and output data set can be used to generate a model that that can further be used in generating residual signal. This approach requires the optimal number of clusters that can be formed beforehand. The number of clusters can correlate with the number of different states in which the system can be (e.g., normal, faulty). The most well-known method for determining the number of clusters is the so-called elbow method.

Principal Component Analysis (PCA) is one of the most well-known algorithms for reducing data dimensionality. It can be used as a pre-processing step, but it can also be used to detect a fault as a technique by itself. The reason why dimensional reduction is useful in fault detection lies in the fact that if we do not choose the optimal data features from those available, the redundant ones will represent noise that will badly affect the accuracy of the final model [8]. Thus, PCA aims to filter only those most important features that carry the most information and include them in our final machine learning model.

Neural networks, in general, are considered the best technique for detecting the failure of highly nonlinear dynamical systems. The reason for this is their ability to "learn" the connection between input and output in a way that they form an associative memory that can return the appropriate output for a given input. One of the most important characteristics is the ability to successfully generalize the predictions on unseen data.

Methods based on historical data are a good choice if we are working with a complex system in which we cannot clearly define each component that could affect the failure and the way the components within the system are connected. Also, unlike methods based on physical models, methods based on historical data can learn from a dynamic environment and adapt to it. Neural networks are good for nonlinear dynamical systems but there they represent a black box for which we do not know how the conclusions and final output are reached. Another drawback is that we need to have a large number of data available. This data however can be obtained in the system design phase, we can also use data from the production phase or artificially generated failures presented in the form of added predefined noise or data generated by system simulation using simulation programs as seen in [8]–[10].

3 EDGE COMPUTING IN THE CONTEXT OF FAULT DETECTION

In the last few years, we have seen an increase in the number of IoT devices and we are slowly entering the post-cloud era, where much more data will be generated at the edges of the network where many information processing applications will be established [11]. Increasing the number of devices and applications inevitably leads to bigger bandwidth consumption. Computing at the edge of the network offers lower maintenance costs, lower power consumption, smaller network bandwidth, and overall improves computation efficiency [12].

3.1 General architecture of edge computing

Edge computing has few basic terms, the term edge node refers to a that is located between the end nodes that generate data and the cloud. An edge node can represent one device or more devices that work together as a whole (edge layer). Edge devices must be capable of data storage, communication, load balancing, caching, and calculations.

Figure 2 presents the general architecture of edge computing. The end devices on the right represent actuators and sensors which can be interconnected by different topologies and communicate with central nodes that further communicate with the edge device or devices. Edge devices mainly include central station, access points, routers, switches, gateways, and they must have the ability to store and analyse data. It is important to mention that edge computing does not necessary exclude cloud computing. Cloud computing (see Figure 2) has its own place, usually for computationally heavy tasks and communication with end users over internet.

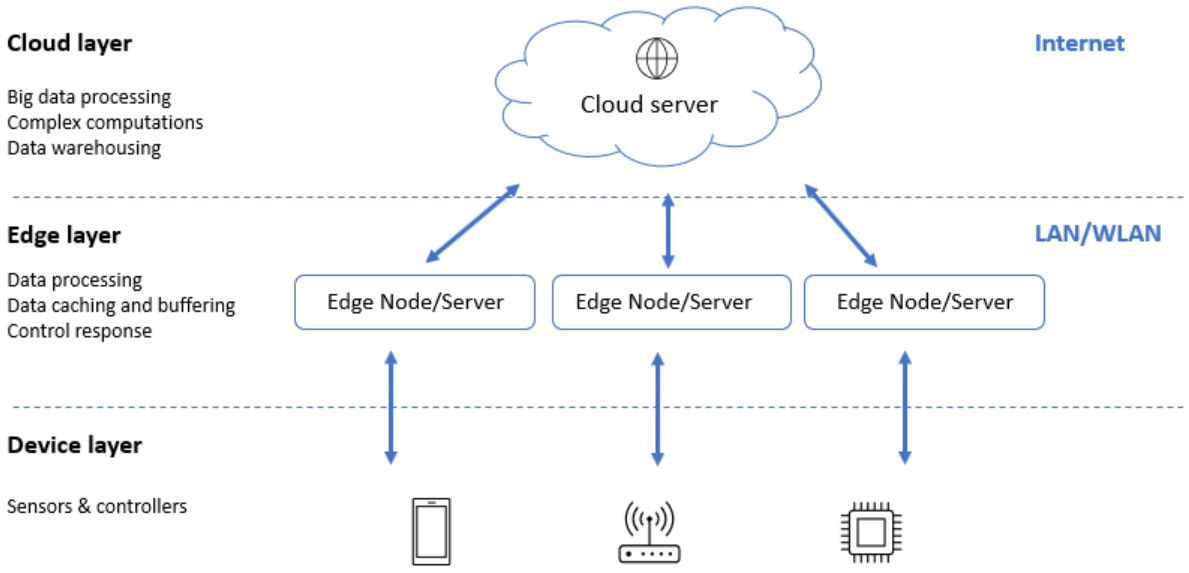

Figure 2: Edge computing architecture

3.2 Fault detection process at the edge of the network

Industrial end devices monitor various industrial parameters and deliver data to edge nodes that further process them. When data reaches the edge node, it processes and stores it without sending it to a distant cloud server. This approach speeds up processing and analysis which is especially important for early fault detection. Security of the data is also increased, by using edge computing instead of cloud, the data doesn't have to be sent to a remote server and thus be at risk of leakage.

Machine learning models deployed on edge computing devices very often need to be compressed and reduced due to limited computing power of said devices. Compressed model usually has lower accuracy than that of the original model. There are several methods of compression, quantization of parameters, distillation of knowledge and parameter pruning. In addition to compression, during model design it is possible to select and reduce number of features and parameters and thus design smaller models that take up less memory, have a faster response while maintaining high accuracy [13].

Some detection methods that use complex machine learning models such as convolutional networks for the model training process require much more resources than edge devices can offer. Sun, Liu, and Yue in their paper [1] propose an architecture that allows the use of larger and more complex models using transfer learning. The proposed architecture consists of three phases, the first phase begins with training the model on a remote server, i.e., cloud computing with a large set of data. Once trained, the model is forwarded by a downlink message to the edge node. The forwarded model has only key layers of the network as they are the first to learn the basic features. The edge node adds additional layers to the resulting model and finally trains over its data set for the specific task it oversees, and which is unique to it.

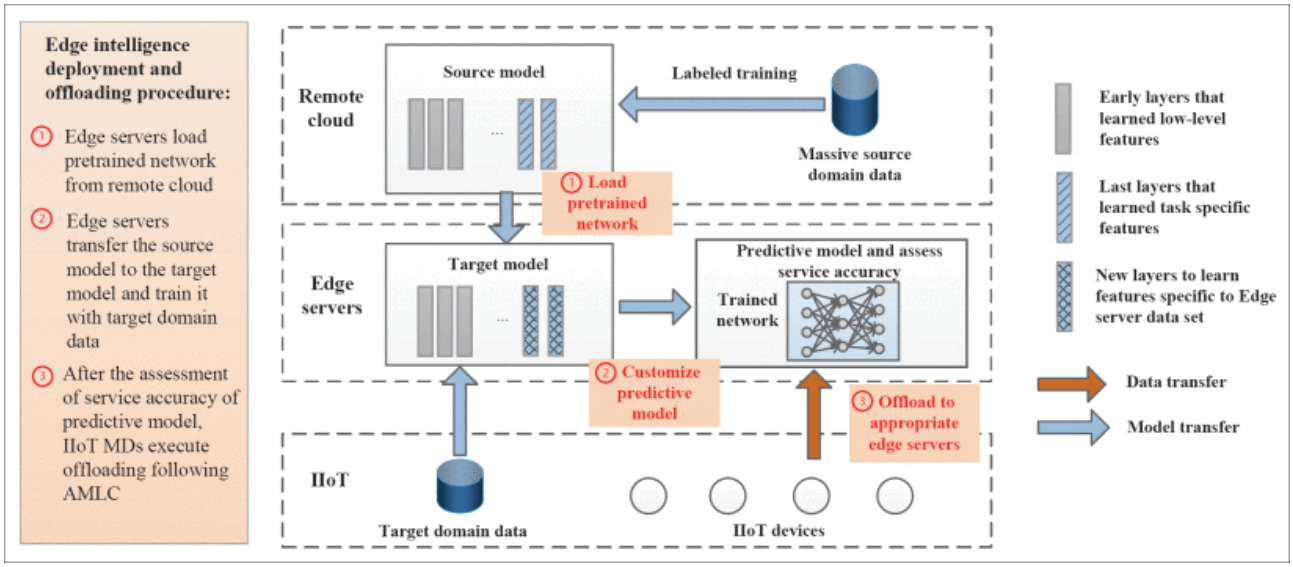

Figure 3: A case study in IIoT edge intelligence using transfer learning [1]. The edge server loads pretrained network from remote cloud, and then customizes its predictive model by replacing the last layers and train with the target domain data IIoT devices offload to the appropriate edge server after assessing the service accuracy following AMLC.

There are times when there is a need to train models over data from multiple remote subnets which also creates a need for synergy of edge computing and cloud computing.

Suppose we have couple identically configured systems s_i that perform an identical function but in different networks. Each system s_i has associated sensors and performs readings for the

purpose of monitoring and detecting early failure. Readings can represent parameters of the internal state of the system or some environmental parameters such as temperature, humidity, pressure, vibrations that affect the degradation of the system. All readings can contribute to the quality of fault detection and are therefore of great importance. The main idea would be to store data from all systems s_i via cloud computing which is the central (parent) node of all systems. The central node thus, with data from multiple sources, could generalize better which is important for the detection of unprecedented failures. This further means that if a fault occurs within the system s_1 after reading the set of parameters vector p_1, the central node located in the cloud that has not yet recorded this fault or combination of parameters p_1, has the opportunity to adjust model parameters (re-train model) with this new information. Once re-trained model will be able to recognize the parameters which led to the failure and be forwarded downlink to the edge nodes. Thus, the edge node of system s_2 will be able to detect a fault prematurely if a combination of parameters p_1 appears in the reading.

4 FAULT DETECTION IN IOT DOSIMETRY

Personal IoT dosimeters such as personal dosimeters are commonly based on silicon solid-state detectors, where ionizing radiation produces pairs of electron/holes that can be collected and converted to a dose reading in $\mu Sv\, h^{-1}$. The user can be informed about the dose rate during different work processes and can be warned against (unexpected) higher dose levels. Most dosimeters (including area dosimeters) are not built to have computational capabilities needed for predicting and detecting faults, furthermore, it is more energy efficient to move computation on edge nodes in the network where multiple dosimeters could be monitored. In this scenario the dosimeters would need to regularly send their measurements to the edge node with additional information of the different states of surrounding such as the temperature, pressure, humidity, and other environmental factors that could affect the integrity of the dosimeter.

To the best of our knowledge, very few work done on fault detection of IoT dosimetry. However, we believe that fault detection of IoT dosimetry could find its use in practice. Early fault detection can inform in time that maintenance is needed. This feature may not be necessary to all dosimetry instrumentation, but could be beneficial for long term high dose rate measurement scenarios.

4.1 Machine learning based fault detection

While designing an algorithm for fault detection purposes, we might choose model-based method such as fault trees if we already know which component is likely to cause a fault and the way the fault of said component would affect other components. Model based methods are in practice preferred for the explainability they possess. However, we may not know which component is likely to fail, and what could be external factors that contribute to the failure of the said component. For this reason, machine learning algorithms may be preferred. As we already stated previously, these algorithms may represent black box, but they also have the ability to generalize on yet unseen data while also learning from the dynamic environment.

Machine learning algorithms are data-hungry and different states od dosimeters needed for training the model is hard to obtain. For this reason, some researchers use fault injection techniques and make use of simulations in which the fault is artificially generated. Y. Liu et al. in their paper [9] use Geant4 simulation to study the radiation damage of the silicon drift detector (SDD) in a deep-space environment, which degrades the detector performance. L. Jiang et al. in [14] propose a method to capture dynamic characteristics of the sensor within the sensor readings and separate them from those of the monitored phenomenon in the monitored or controlled system, thus enabling one to track the key dynamic indicators of the sensor and statistically assess the significance of their changes, without explicit need for redundant sensing hardware.

Once the data for training the machine learning model is collected or artificially generated, we can swich between different algorithms and observe which one gives us better results. Long short-term memory (LSTM) neural networks might also be of great use for multivariate time-series data.

5 CONCLUSION

Computing at the edge of the network has become an inevitable component of the entire ecosystem in many plants, monitoring and fault detection is just one of the components that is of great importance for various institutions not only for industry. Early fault detection can extend the life of the device, increase the level of safety, and reduce the cost of production. When choosing a fault detection method, it is necessary to consider the type of possible faults, system structure and dynamics, complexity, amount and type of available data [15], but above all we must understand the system for which we are building a model. The simplest and most direct approach is to check that the readings are within the allowable range, which is sufficient for simple systems. If we are working with a complex dynamic system in which there are several points of possible failure, and the process structure itself is complex, more advanced methods that use machine learning are needed. In practice, we want the simplest possible models whose outputs, unlike deep neural networks, can be explained. Although edge computing is the solution to many of the challenges in IIoT, for some it still cannot give a full solution, the most obvious example is the need for more computing power that is needed for large, more complex models. Nodes at the edge of the network still have limited computing resources which is why such more complex computing must be done on the cloud. Due to the growing number of devices, the use of computing at the edge of the network is expected to increase in the future and thus it is possible that IoT dosimeters will follow this trend.

REFERENCES

[1] W. Sun, J. Liu, and Y. Yue, 'AI-Enhanced Offloading in Edge Computing: When Machine Learning Meets Industrial IoT', *IEEE Network*, vol. 33, no. 5, pp. 68–74, 2019, doi: 10.1109/MNET.001.1800510.

[2] R. Isermann, 'Supervision, fault-detection and fault-diagnosis methods — An introduction', *Control Engineering Practice*, vol. 5, no. 5, pp. 639–652, 1997, doi: https://doi.org/10.1016/S0967-0661(97)00046-4.

[3] A. Mouzakitis, 'Classification of Fault Diagnosis Methods for Control Systems', *Measurement and Control*, vol. 46, pp. 303–308, Dec. 2013, doi: 10.1177/0020294013510471.

[4] D. Kritzinger, '4 - Fault tree analysis', in *Aircraft System Safety*, D. Kritzinger, Ed. Woodhead Publishing, 2017, pp. 59–99. doi: https://doi.org/10.1016/B978-0-08-100889-8.00004-0.

[5] N. Hadroug, A. Hafaifa, B. Alili, A. Iratni, and X. Chen, 'Fuzzy Diagnostic Strategy Implementation for Gas Turbine Vibrations Faults Detection: Towards a Characterization of Symptom–fault Correlations', *Journal of Vibration Engineering & Technologies*, vol. 10, no. 1, pp. 225–251, Jan. 2022, doi: 10.1007/s42417-021-00373-z.

[6] M. A. Eissa, A. Sali, M. K. Hassan, A. M. Bassiuny, and R. R. Darwish, 'Observer-Based Fault Detection With Fuzzy Variable Gains and Its Application to Industrial Servo System', *IEEE Access*, vol. 8, pp. 131224–131238, 2020, doi: 10.1109/ACCESS.2020.3010125.

[7] K. E. Holbert and K. Lin, 'Nuclear Power Plant Instrumentation Fault Detection Using Fuzzy Logic', *Science and Technology of Nuclear Installations*, vol. 2012, p. 421070, Sep. 2012, doi: 10.1155/2012/421070.

[8] M. Talha, F. Asghar, and S. Kim, 'A Matlab and Simulink Based Three-Phase Inverter Fault Diagnosis Method Using Three-Dimensional Features', *The International Journal of Fuzzy Logic and Intelligent Systems*, vol. 16, pp. 173–180, Sep. 2016, doi: 10.5391/IJFIS.2016.16.3.173.

[9] Y. Liu, T. Zhu, J. Yao, and X. Ouyang, 'Simulation of Radiation Damage for Silicon Drift Detector', *Sensors (Basel)*, vol. 19, no. 8, p. 1767, Apr. 2019, doi: 10.3390/s19081767.

[10] M. Vathoopan, M. Johny, A. Zoitl, and A. Knoll, 'Modular Fault Ascription and Corrective Maintenance Using a Digital Twin', *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1041–1046, Jan. 2018, doi: 10.1016/j.ifacol.2018.08.470.

[11] W. Shi and S. Dustdar, 'The Promise of Edge Computing', *Computer*, vol. 49, no. 5, pp. 78–81, 2016, doi: 10.1109/MC.2016.145.

[12] K. Cao, Y. Liu, G. Meng, and Q. Sun, 'An Overview on Edge Computing Research', *IEEE Access*, vol. 8, pp. 85714–85728, 2020, doi: 10.1109/ACCESS.2020.2991734.

[13] J. Chen and X. Ran, 'Deep Learning With Edge Computing: A Review', *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019, doi: 10.1109/JPROC.2019.2921977.

[14] L. Jiang, D. Djurdjanovic, J. Ni, and J. Lee, 'Sensor Degradation Detection in Linear Systems', in *Engineering Asset Management*, London, 2006, pp. 1252–1260.

[15] D. Miljković, *Fault detection methods: A literature survey*. 2011, p. 755.