FIREMAN

WP2 Deliverable 2.4
Theoretical Bounds on Rare Events Detection

Project Title:	Framework for the Identification of Rare Events via Machine learning and IoT Networks

Title of Deliverable:	Theoretical Bounds on Rare Events Detection
Status-Version:	Final-1.0
Delivery Date:	31/1/2021

Contributors:	Ioannis T. Christou, Daniel Gutierrez-Rojas, Arun Nayaranan, Mehar Ullah, Arthur Sena, Pedro Nardelli, Charalampos Kalalas, Merim Dzaferagic, Irene Macaluso, Eslam Eldeeb, Hirley Alves
Reviewers:	Asst. Prof. Mateus Giesbrecht, University of Campinas, Brazil
Approved by:	All Partners

Document Revision History

Version	Date	Description
v0.1	7 January 2021	Table of Contents
v0.8	20 January 2021	All basic contents added
v0.9	22 January 2021	Final contents added and send to external reviewers
v1.0	29 January 2021	Changes indicated by reviewer implemented and deliverable approved by partners

Summary

This document presents different results that identify fundamental limits of the approach taken by FIREMAN project. Specifically, we have analyzed how pervasive edge computing paradigm – in contrast to cloud computing – may support the deployment of industrial IoT (IIoT), which is the basis of FIREMAN. In fact, we have identified the key bottlenecks that may indicate what is the best IIoT solution to be considered for the specific cases, including the test cases in Work Package (WP) 6 and providing a theoretical limitation of what can be done. In addition, we have presented a brief discussion about explainability of machine learning algorithms, which is important to any industrial solution; this is an ongoing work that will be further developed in WP5. We have also presented limitations of data imputation and alarm classification and their limitations, both related to the research carried out in WP3 and WP4. The research is performed within the context of the EU CHIST-ERA project FIREMAN (grant CHIST-ERA-17-BDSI-003).

Table of Contents

1 Introduction

1.1 Objective of the document

The goal of this document is to present the fundamental limitations of different solutions used by FIREMAN in different WPs. We have presented limitations from network architecture side, considering the advantages and promises of pervasive edge computing for industrial systems. In addition, we have indicated our initial studies of the fundamental limits of explainability of machine learning utilizing QARMA algorithm (studied in D2.3 and WP5). We have also indicating how data imputation can be used to help to recover samples lost in wireless communication transmissions and also in higher-level scenarios.

2 Limitations and trade-off of industrial pervasive edge computing based on IoT

The term edge computing has already been widely used by the research and industrial community alike. In this section, our aim is to explain why we denominate it as pervasive edge computing and its importance for industrial environments considering the fundamental trade-offs that determine its performance limits. These aspects are relevant to any specific solution to be developed by FIREMAN following the framework developed in WP2. The full analysis of pervasive edge computing is presented in [1].

2.1 What is pervasive edge computing?

The main idea behind the *edge computing* paradigm is to exploit the storage and computing capabilities of different devices at (or near) the network edge. A natural question that arises is what is an *edge*? Edge can be defined as any computing and networking resource, such as a smart phone or a 5G base station, that lies between the data source and the "cloud" (i.e., the core network). In other words, edge in this paper refers to the edge of the core network including all the devices related to the radio access network. Since these (potential) edge computing elements are widely spread, we denominate this architecture as *pervasive edge computing* (PEC), and it also incorporates the more restrictive (or fuzzy) concepts of mobile edge computing (MEC), edge servers, edge nodes, and fog network. Figure 1 illustrates the process of PEC (which is decentralized with computational tasks distributed among the nodes at the edge of the core network) and contrasts it with cloud computing (which is mainly centralized with computational tasks being performed in the servers at the core network and the Internet).

PEC is beneficial as it moves data processes away from centralized servers [2]. As a result, some IIoT applications do not need to send their data through the core network, avoiding congestion and potentially high delays. Moreover, PEC can also pre-process data by filtering during the acquisition phase, thereby improving the speed of data analysis and the decision-making processes [3–5]. Local processing provided by PEC also helps to protect sensitive data that are better processed on an edge device instead of sending to a cloud. Note that although PEC provides significant benefits to IIoT, cloud computing cannot be eliminated completely because having a centralized location for the data storage and analysis still has many benefits in different industrial applications. PEC is important for offloading some tasks from the core network and also to fulfill strict latency requirements, but the remaining data may still have to be sent to the cloud for processing because of its better processing capabilities. In the following section, we will describe some of the tradeoffs of edge and cloud computing architectures.

2.2 Cloud versus pervasive edge computing: the key tradeoffs

The PEC paradigm has emerged not to replace conventional local and cloud computing solutions, but to complement their capabilities. The truth is that PEC alone cannot provide a universal solution that can tackle all issues of future networks. Instead, it comes with various tradeoffs that need to be investigated and well understood. Cooperation among local, edge,

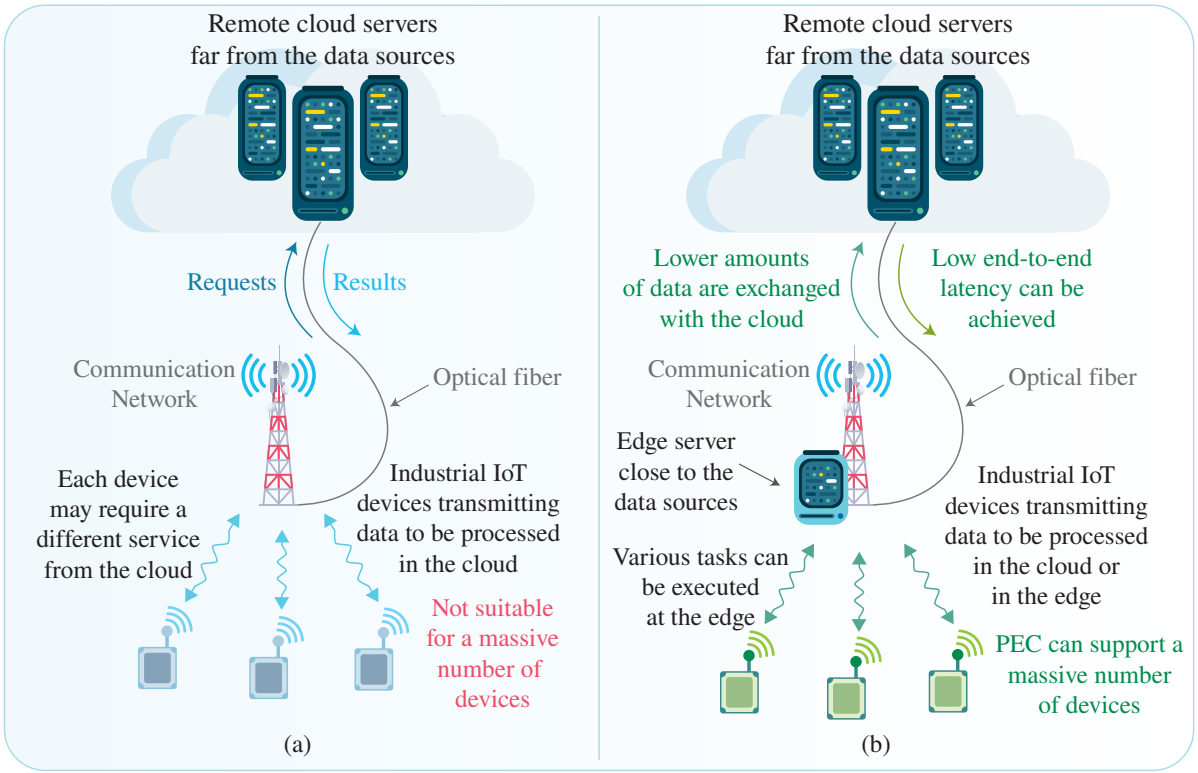

Figure 1: Cloud versus Pervasive Edge Computing. (a) Typical cloud computing network where the end devices, which sense their operation environments, forward the collected data to the cloud through the communication network for further processing and inference. (b) Typical pervasive edge computing network where the edge devices in the network proximity of end devices process the collected data, and the cloud acts as a complementary processing and storage unit.

and cloud computing will be essential to meet the diverse requirements of IIoT. Therefore, it is important to present a fundamental understanding of the benefits and drawbacks that can be potentially provided by the two architecture paradigms, edge and cloud computing, which are the basis of any data processing to be developed in FIREMAN. In what follows, we shed light on the main tradeoffs of cloud computing and PEC solutions.

Latency: When PEC comes into play, the first benefit that one can think of is the lower end-to-end latency that can be achieved. Indeed, as widely demonstrated [6–10], in contrast to cloud computing, the distances that data packets need to travel are, in most cases, shortened with edge servers installed closer to end-user applications; this can greatly contribute to reducing latency. However, there are scenarios in which such a benefit might not be attained. Latency does not depend only on the distance between the user and the processing server. It also depends on other factors such as the computational complexity of tasks, processing power of edge servers, and edge traffic. For instance, if the edge network is congested, or if the time spent to process the offloaded computational tasks is too high, it might be more advantageous to opt for some cloud computing solution. This tradeoff can be visualized in Figure 2 that shows the latency versus the central processing unit (CPU) cycles

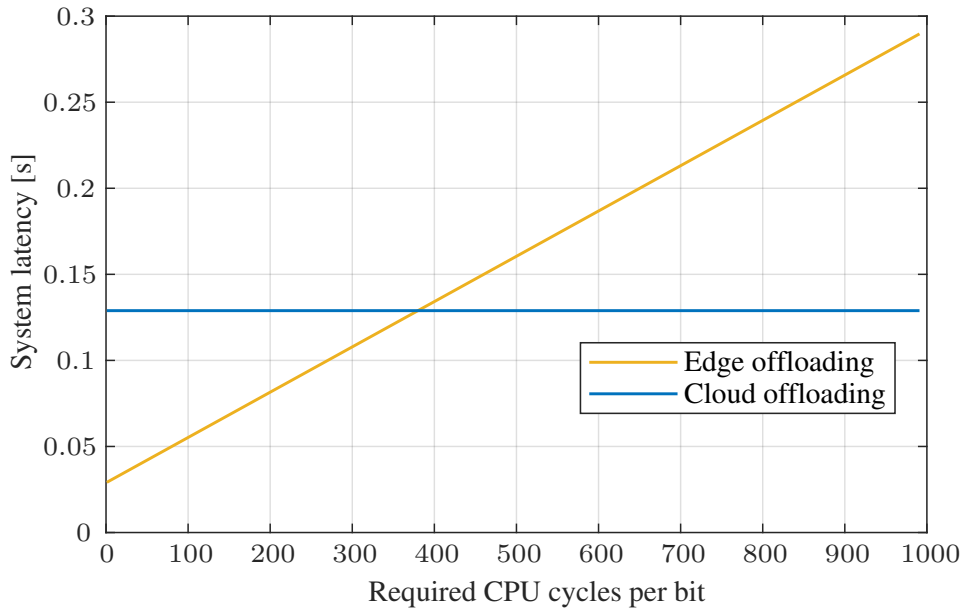

Figure 2: System latency versus required CPU cycles per bit in wireless systems assisted by edge and cloud computing ($b = 1$ Mbit, $B^{\text{Edge}} = 10$ MHz $B^{\text{Cloud}} = 10$ MHz, $\gamma^{\text{Edge}} = \gamma^{\text{Cloud}} = 10$ dB, $f^{\text{Edge}} = 6$ GHz, $f^{\text{Cloud}} = \infty$, $\tau^{\text{Cloud}} = 100$ ms).

required per bit offloaded by a single device in a wireless system assisted by either cloud or PEC. This simple example is generated based on the system models proposed in [6,7], in which the total edge computing latency can be represented by

$$T^{\text{Edge}} = \frac{b}{B^{\text{Edge}} \log_2(1+\gamma^{\text{Edge}})} + \frac{bC}{f^{\text{Edge}}}, \tag{1}$$

and the cloud computing latency by

$$T^{\text{Cloud}} = \frac{b}{B^{\text{Cloud}} \log_2(1+\gamma^{\text{Cloud}})} + \frac{bC}{f^{\text{Cloud}}} + \tau^{\text{Cloud}}, \tag{2}$$

where b is the total number of bits of the offloaded task; C is the number of CPU cycles required to compute one bit; B^{Edge}, B^{Cloud} and γ^{Edge}, γ^{Cloud} represent, respectively, the bandwidths and signal-to-noise ratios (SNR) of the uplink channels between device and edge server and device and cloud gateway; f^{Edge} and f^{Cloud} denote the CPU's clock frequency in the edge server and cloud server, respectively; and τ^{Cloud} is a fixed latency due to the long distance between the cloud gateway and the central cloud server. As shown in Figure 2, because cloud servers dispose of abundant computational resources, we assume that $f^{\text{Cloud}} \to \infty$. Consequently, the latency in the system assisted by cloud computing is not affected when the task complexity increases. On the other hand, the latency in the system assisted by PEC escalates with the increase in the number of CPU cycles, i.e., when the computational complexity of the task becomes high.

A dynamic mobile scenario is another example where it can be challenging to provide low end-to-end latency. This is because mobile IIoT devices might not always be able to fully use

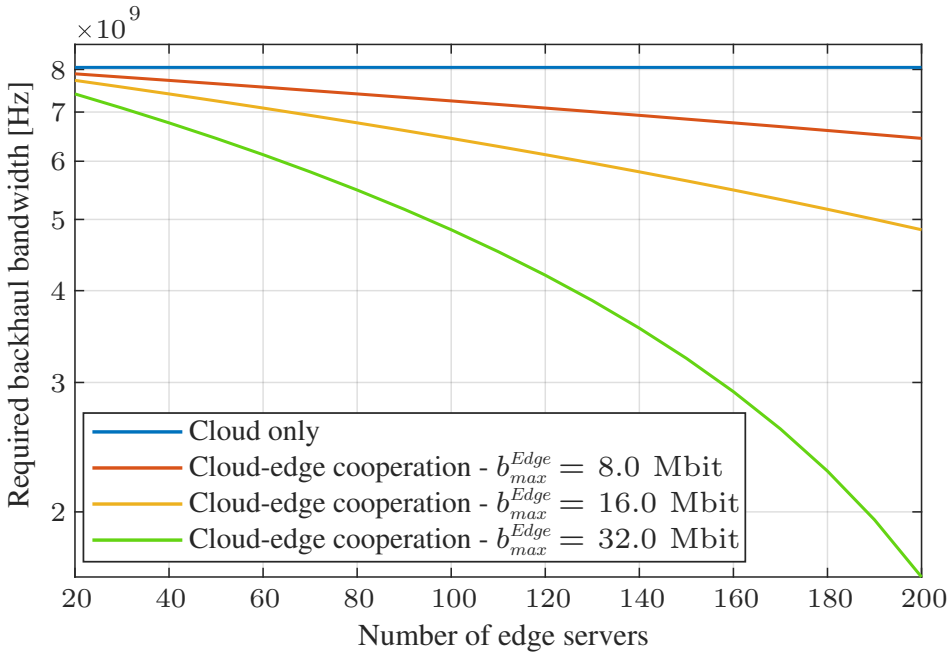

Figure 3: Required backhaul bandwidth versus number of edge servers for different values of b_{max}^{Edge} ($D = 500$ devices, $\gamma^{\text{BH}} = 30$ dB, $T^{\text{BH}} = 100$ ms).

edge servers (which have a fixed location) in their vicinity. As proposed in [11], a solution for this issue can be achieved by employing some multi-hop strategy so that the data can reach the nearest server. However, such an approach also comes at the cost of increased communication latency. In summary, to efficiently meet the latency requirements of different IIoT applications, factors such as task complexities, the processing power of servers, and the network topology must be carefully taken into consideration when designing a PEC-assisted network.

Backhaul bandwidth: To satisfactorily attend a massive number of connections via a centralized cloud architecture, stringent bandwidth requirements in the backhaul lines are needed. Otherwise, severe congestion and packet losses could occur, resulting in unstable and unreliable cloud service provisioning. To alleviate such an issue, cloud providers will have to make heavy investments in communication infrastructures, which can be financially unviable. Fortunately, PEC offers cheaper efficient alternatives for reducing backhaul data traffic. By distributing the computational workload among different edge servers, lower amounts of data are required to be exchanged with the cloud. Edge data caching, data cleansing, and compression are other efficient approaches for tackling the traffic issue [12–14]. A combination of all these strategies can effectively relax the backhaul bandwidth requirements and decrease the costs of communication infrastructure.

The simulation examples illustrated in Figure 3 show the potential benefits that the cooperation between cloud and PEC can provide to reduce the required backhaul bandwidth. In these examples, we consider a scenario with D devices and N edge servers, in which the ith device offloads b_i bits to be computed in a cooperative manner by cloud and edge servers. Specifically, bits are transmitted to the cloud server only when the sum of the tasks exceeds the

combined computational capacity of the edge servers, i.e., when $\sum_{i=1}^{D} b_i > \sum_{j=1}^{N} b_{j,max}^{Edge}$, where $b_{j,max}^{Edge}$ is the maximum number of bits that the jth edge server is able to process, in which, for illustrative purposes, we assume $b_{1,max}^{Edge} = b_{2,max}^{Edge} = \cdots = b_{N,max}^{Edge} = b_{max}^{Edge}$. With these assumptions, we compute the required backhaul bandwidth by averaging 1×10^6 realizations of the following formula

$$B^{\mathsf{BH}} = \begin{cases} \frac{\sum_{i=1}^{D} b_i - N b_{max}^{Edge}}{T^{\mathsf{BH}} \log_2(1+\gamma^{\mathsf{BH}})}, & \text{if} \quad \sum_{i=1}^{D} b_i > N b_{max}^{Edge} \\ 0, & \text{otherwise,} \end{cases}, \tag{3}$$

where the number of bits b_i is drawn from a uniform distribution between 100 kbit and 32 Mbit, and γ^{BH} and T^{BH} are, respectively, the backhaul SNR and the desired backhaul latency. These results highlight the attractive capabilities of PEC to alleviate backhaul requirements. As one can see, remarkable reductions in bandwidth are achieved when increasing the number of edge servers. These savings become even more prominent when the processing power of edge servers gets high, i.e., when b_{max}^{Edge} is increased.

Privacy and Security: Privacy and security issues are critical concerns that arise with PEC-assisted systems [15–23]. The pervasive deployment of edge servers, geographically distributed and closer to the end users, can introduce numerous vulnerabilities to the network; moreover, these vulnerabilities can be hard to track. Edge servers may have limited computational capabilities and might lack physical protection, which creates a favorable scenario for hacker invasions and eavesdropping [15]. On the other hand, the centralized architecture of the cloud computing paradigm together with the high computational power of its cloud servers enable the implementation of strong security measures, including sophisticated encryption techniques and very safe physical infrastructures. As a result, in general, it is more challenging to hack and to physically violate cloud servers [24].

Robustness to failures: One strong aspect of PEC is that it enables robustness to failures. Due to the branch architecture of PEC, it becomes very hard to shut down the entire network. For instance, if an electricity outage happens in a particular area of the grid, the PEC services of other areas will continue to operate normally, unaffected. In contrast, if a given IIoT network relies solely on centralized cloud computing, when the electricity supply fails due to any natural disaster happening in the cloud infrastructure, or the backhaul communication link becomes unstable, the whole network will fail [25, 26]. To exemplify the robustness of PEC, in Figure 4, we show the outage probability curves experienced in two wireless systems assisted by cloud and PEC. In the first system, we consider that one IIoT device is assisted by only cloud computing, in which the device transmits its data to a gateway, such as a base station, that communicates with a central server through a wireless backhaul link with bandwidth B^{BH}. In particular, we assume that the cloud-assisted IIoT device experiences outage if the data rate achieved in the backhaul link is less than its target rate, R^{target}. On the other hand, the second system is assisted by only PEC, where one IIoT device offloads its data to N edge servers through N wireless links, each with bandwidth B^{Edge}, such that $B^{\mathsf{Edge}} < B^{\mathsf{BH}}$. Differently from the first scenario, in this edge-assisted system, the IIoT device faces outage only if the data rates achieved in all N links are less than R^{target}. For illustrative purposes, the SNR observed in the backhaul link for the cloud-assisted system is defined by $\gamma^{\mathsf{BH}} = \frac{|h^{\mathsf{BH}}|^2}{\sigma_n}$, and for edge-assisted system by $\gamma_n^{\mathsf{Edge}} = \frac{|h_n^{\mathsf{Edge}}|^2}{\sigma_n}$, where σ_n represents the noise variance, and h^{BH} and h_n^{Edge} denote,

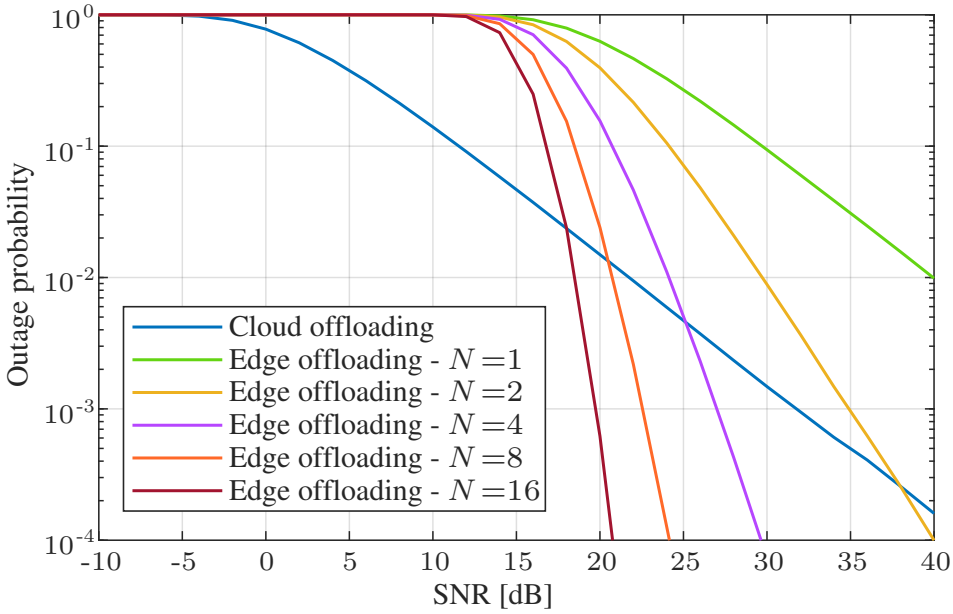

Figure 4: Outage probability curves for wireless systems assisted by cloud and pervasive edge computing considering different numbers of edge servers ($B^{\text{BH}} = 2$ MHz, $B^{\text{Edge}} = 50$ kHz, $R^{\text{target}} = 5$ Mbit/s).

respectively, the channel coefficients for backhaul link and for the nth edge link, which are modeled by complex Gaussian random variables with zero mean and unit variance. Under these definitions, the outage probability for the cloud-assisted system is calculated by

$$P^{\text{Cloud}} = P[B^{\text{BH}} \log_2(1 + \gamma^{\text{BH}}) < R^{\text{target}}], \tag{4}$$

and for the edge-assisted system by

$$P^{\text{Edge}} = \prod_{n=1}^{N} P[B^{\text{Edge}} \log_2(1 + \gamma_n^{\text{Edge}}) < R^{\text{target}}]. \tag{5}$$

As shown in Figure 4, even though the system assisted by cloud computing achieves the best performance when the SNR is low, the one assisted by PEC becomes more robust in the moderate to high SNR regime as the number of edge servers increases, outperforming the cloud-assisted counterpart. These results provide a clear example of the tolerance to failures of a PEC system. Nevertheless, despite the resiliency that PEC can provide to IIoT, systemic software failures can still happen, and this can result in a generalized network collapse, as reported in [25, 27, 28].

Monetary cost: Cloud computing providers usually charge the costumers for data transmissions, storage, and computation services [29]. Therefore, if the number of transmissions grows, if the amount of data that needs to be stored increases, or if computation tasks become more complex (which is likely to happen in IIoT), it can become excessively expensive to rely only on cloud services [30]. On the other hand, by computing and caching data and tasks at the edge, the PEC-assisted networks can decrease the traffic to cloud servers and effectively

reduce the monetary costs with cloud services. However, note that even though PEC can reduce the expenditure with cloud services, shifting to a pervasive decentralized architecture can also lead to an increase in costs for the installation of new hardware and additional energy consumption [31].

These tradeoffs show that the design of a PEC-assisted network should be optimized based on the characteristics and requirements of each specific application. At the same time, standardized flexible solutions, such as those offered by 5G (and beyond) communication systems, are crucial for guaranteeing the heterogeneous quality of services of the future IIoT, making 5G a key enabler of PEC. These aspects should be taken into account when selecting the most suitable IoT platform for the specific computational tasks to be considered, following the framework presented in D2.2 and D4.1.

2.3 Artificial Intelligence in PEC

IoT devices are normally heterogeneous devices and difficult to coordinate. As a result, a major challenge in PEC is the efficient resource co-ordination and communication between different types of heterogeneous edge devices, from computers equipped with graphical processing units (GPUs) to smart phones with mobile processors to devices with just small single-board computers like Raspberry Pi [32]. Secondly, edge devices have to coordinate with the cloud under dynamic network conditions and different settings to ensure satisfactory application-level performance. Several AI techniques have been proposed to meet these challenges. Among them, deep learning (DL) is very promising due to its potential to create hybrid approaches that combine cloud and PEC.

In this subsection, we first briefly present the recent status of DL research applied to PEC before moving to our major focus, the relatively new concept of federated learning (FL). FL is an extremely important technique that has a nearly symbiotic role with PEC for addressing privacy issues. Privacy is a major concern with smart devices because of the need to interact and share data with the cloud and third-party platforms. FL takes advantage of PEC to maintain user privacy while performing the required computations efficiently, which can be seen as advantageous for industrial scenarios.

2.3.1 Deep learning and applications to PEC

The application of DL at the network edge offers many improvements to PEC as DL and PEC together can support numerous applications, including computer vision, smart surveillance, natural language processing, network functions, virtual reality (VR) and augmented reality (AR). In a recent paper, Chen *et al.* surveyed research literature at the confluence of DL and PEC [32]. They highlighted that the researches so far have been based on three major architectures: (1) centralized edge server-based architectures, where data from end devices are sent to one or more edge servers for computation; (2) semi-distributed architectures in which the computation is shared among end devices, edge servers, and the cloud via a joint computation mechanism; and (3) distributed architectures based on on-device computations, where deep neural networks are executed on the end device itself.

In the future, edge intelligence is expected to move DL implementations from the cloud to the edge, forming the so-called edge DL [33, 34]. Research into the industrial applications of

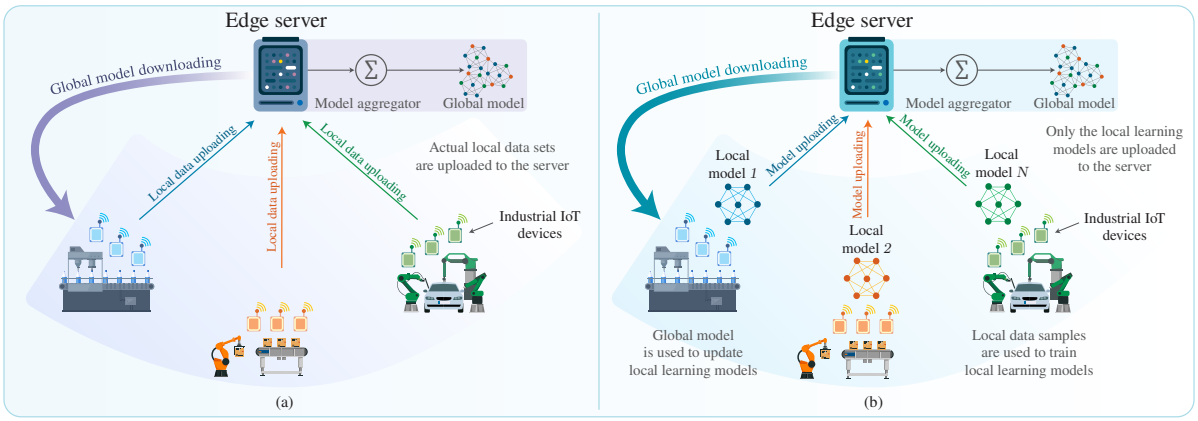

Figure 5: (a) Traditional deep learning: end devices send their local data to a global server that collects all data and performs the complete model training. (b) Federated learning: end devices use local data to cooperatively train a local model and then send it to the global server for aggregation. These steps are repeated in multiple rounds until a desirable accuracy is achieved.

this convergence of DL and the edge is still nascent but is expected to increase significantly, especially with the implementation of 5G. In [35], the authors proposed an edge computing-based DL model that migrates the DL process from the cloud to the edge in an IIoT network using edge computing concepts. The DL model is optimized so that computational power requirements are reduced. By deploying a testbed implementation with their proposed convolutional neural network model and real-world IIoT dataset, the authors showed that network traffic overheads are reduced without compromising the classification accuracy.

In another study, a DL-based method was used to detect hazardous conditions in supermarkets, such as spilled liquids or fallen items on floors [36]. They showed that their lightweight DL model can be deployed on edge devices for quick computations. Similar studies based on intelligent visual recognition using DL implemented at the edge have been applied to industrial electrical equipment [37], health monitoring [38], and flat surface texture inspection [39]. In these cases, the more intensive computations for training are performed at the cloud while the lighter classification tasks are distributed through the edges. This is the idea behind federated learning, a recent innovation that is particularly relevant to the PEC paradigm.

2.3.2 Federated learning and applications to PEC

FL is a relatively recent ML paradigm that was motivated by the need to protect user privacy. FL aims to train a centralized model using training data that are distributed over a large number of client devices that themselves are a part of the training [40]. In other words, FL differs from classical ML learning approaches such as DL in the fact that the model training does not use a single processing device [41, 42]. In FL, end devices use their local data to cooperatively train an ML model (using ML techniques such as DL, support vector machines (SVM), artificial neural networks (ANN), etc.) required by an FL server. They then send the model updates to the FL server for aggregation. These steps are repeated in multiple rounds until a desirable accuracy is achieved [43]. Figure 5 illustrates the difference between DL and

FL. In general, FL involves the training of statistical models directly on remote devices, and it is motivated by the need to maintain information privacy [44].

Such a decentralized approach facilitates collaborative complex ML techniques while guaranteeing that the data will remain in the personal devices, thereby preserving privacy. Note that there is usually an underlying assumption that the data owners are honest: they use their *real* private data to do the training and submit the true local models to the FL server. The main advantages of the FL approach are as follows:

- *Bandwidth efficiency*: Less data is required to be transmitted to the cloud.
- *Privacy*: Raw (local) data need not be sent to the cloud any more.
- *Low latency*: Real-time decisions can be made locally instead of being made in the cloud.

An important question in FL research is whether its performance is comparable to that of traditional DL-based approaches for resource coordination. In [45], the authors examined this question by employing FL for the joint allocation of communication and computing resources by guiding the training of deep reinforcement learning agents. The IoT devices in their model harvested energy units from edge nodes and stored them in their energy queue. The authors demonstrated that the fluctuation range of FL-based DL with respect to utility variation is bigger than that from centralized training. Their results confirm that the performance of FL-based DL training for computation offloading approaches the results from centralized DL training.

The efficient utilization of limited computation and communication resources to increase the optimal learning performance of different applications was examined in [46]. The authors considered a typical edge computing architecture where the edge nodes are interconnected with the remote cloud via network elements such as gateways and routers. The raw data was collected and stored at multiple edge nodes and FL learning was performed. In the FL approach in this work, the frequency of global aggregation was configurable; that is, it was possible to aggregate at an interval of one or multiple local updates. Each local update consumes computation resources and each global aggregation consumes communication resources of the network. The amount of consumed resources may vary over time, and there is a complex relationship among the frequency of global aggregation, the model training accuracy, and resource consumption. This is a tradeoff between the model resource optimization and precision. The authors in [46] analyzed the convergence bound of gradient-descent based FL from a theoretical perspective and proposed a control algorithm that learns data distribution. This algorithm was tested using real datasets both on a hardware prototype and simulated environment.

Based on the distributed topology that characterizes the PEC paradigm, different architectures are being considered to deploy FL techniques in order to improve the tradeoff between energy consumption, computation capacity, and training capabilities of distributed learning nodes connected by a customized communication network. For instance, in [47], a two-stage FL algorithm was proposed for a collaborative scenario between user equipment (UE), unmanned aerial vehicles (UAV)/ base stations (BS), and a heterogeneous computing platform (HCP) to predict content caching. Here, an asynchronous weight updating method was adopted to reduce the effects of redundant learning in FL.

FL in PEC also increases the reliability and security for some critical applications such as vehicular edge computing (VEC). VEC faces a major challenge that the accuracy of image quality can decrease during the model aggregation. To address this, the authors in [48] proposed a selective model aggregation exploiting a geometric model that illustrates the relationship between the object of interest and the camera in each vehicular agent. FL was used to train image classification and to tackle the asymmetry caused by it, and a model selection procedure was formulated as a two-dimensional contract theory problem. Then, the contract problem was transformed into a problem that can be tracked by relaxing and simplifying the complicated constraints. In [49], FL was applied to urban informatics tasks where the vehicular network consisted of a macro base station, a number of road-side units, and moving vehicles. The authors focused on three aspects: enhancing the privacy of the updated models in FL, development of a new asynchronous FL architecture by leveraging distributed peer-to-peer update schemes, and boosting the proposed FL process. The proposed FL method not only gave higher accuracy than the compared methods such as convolution networks, GraphStar, and text graph convolution networks, but it could also execute parallel local training; moreover, increasing the number of data providers did not affect the accuracy.

3 Explainability of Decisions of Other Algorithms by QARMA Extracted Rules in the Limit

In this section, we will present our initial results of how Quantitative Association Rule Mining Algorithm (QARMA; introduced in D.2.3) can be utilized to explain the decisions given by other algorithms. In particular, once the set of all "widest" rules that satisfy a minimum support and confidence threshold criterion have been extracted from a dataset, they can be used not only to form a rule-ensemble on their own right, but also to help explain the decisions made by other classifiers, regardless of their type; in that respect, the extracted rules can help make black-box type classifiers such as deep neural networks easier to explain, or at least interpret. The idea is that whenever a classifier makes a decision on a new data instance x, the rule set produced by QARMA can be searched to see if it contains any rules that "justify" that decision; if so, then among all rules that are triggered by the data x and have as consequent the same class $y = \omega$ that the black-box classifier made, the one with the highest confidence is proposed as the explanation for the classifier's decision (ties can be broken according to the support of the highest confidence rules.)

As an example, assume a 2-hidden layer neural network is trained on an imaginary 2-class (ω_1, ω_2) dataset with 10 numeric attributes $x_1, \ldots, x_{10}$. On a new data instance $d = (d_1, \ldots d_{10})$ the neural network replies with the answer ω_1. As long as QARMA was able to extract any rule of the form:

$$\bigwedge_{j \in S} x_j \in [l_j, h_j] \rightarrow y = \omega_1 \tag{6}$$

and it holds that $\forall j \in S : d_j \in [l_j, h_j]$ (i.e. the rule's antecedents are satisfied by the data instance d, the QARMA rule set "explains" the neural network's decision by the rule(s) (6).

An obvious question to ask then, is the following: *under what circumstances, can the QARMA produced rules guarantee to have an explanation for a decision made by another classifier when the decision of that other classifier is the correct one?* We answer this question in the limit case of the simplest classification problem possible: the case of a 1-dimensional, 2-class classification problem where the patterns come from 2 distinct Gaussian distributions $\mathcal{N}(\mu_1, \sigma^2)$, and $\mathcal{N}(\mu_2, \sigma^2)$ with the same variance and with corresponding probability density functions (pdfs) $\phi(x; \mu_i, \sigma^2)$ centered around distinct means (without loss of generality, we assume $\mu_1 < \mu_2$), and cumulative distribution functions (cdfs) $\phi_i(x), i = 1, 2$. See Figure 6.

For this simple problem, we derive the conditions under which the decision of an optimal Bayesian classifier (which assigns a data point X to the class for which $P[\omega_i|X]$ is maximal) will be explained by at least one rule extracted by QARMA. We assume that the dataset on which QARMA is applied, is created by equi-sampling from the two classes, so that the priors $P[\omega_i]$ are equal to 0.5. We also assume that the dataset was created without the introduction of any bias during the sampling process. From such a dataset, the two widest (and only) rules that should be extracted from the dataset are the following:

$$r_1 : x < c \rightarrow y = \omega_1, \tag{7}$$

$$r_2 : x \geq c \rightarrow y = \omega_2, \tag{8}$$

where c is the unique solution to the equation $\phi(c; \mu_1, \sigma^2) = \phi(c; \mu_2, \sigma^2)$.

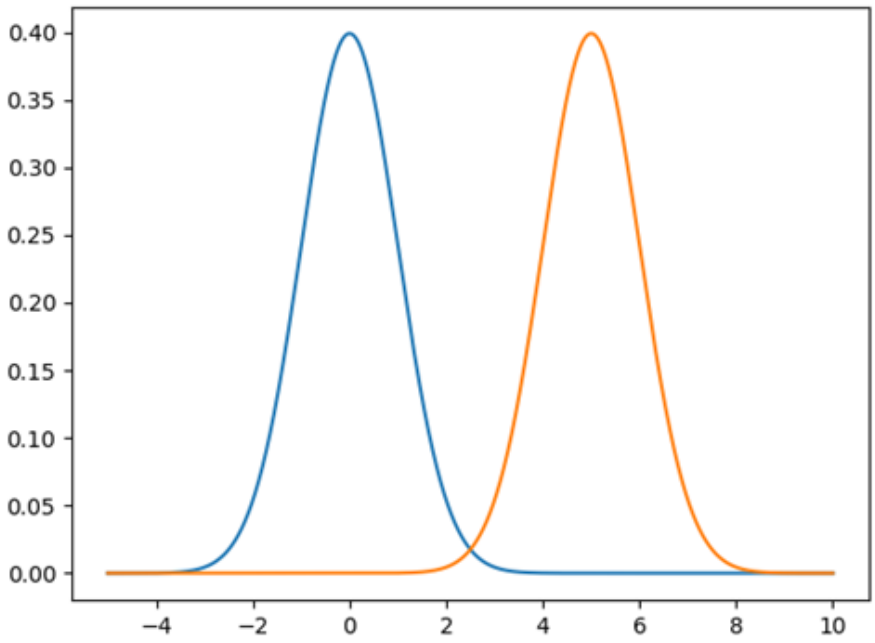

Figure 6: Simple 2-class Classification Problem in 1D. Class ω_1 pdf is the blue line, and class ω_2 pdf is the orange line.

Given that the QARMA guarantees that all and only the widest rules that hold on the dataset with the required minimum support and confidence selected will be extracted [50], the two rules in (7) and (8) will indeed be produced as long as the requested minimum support and confidence are less than or equal to the values they are expected to hold with. The expected support of the rules $r_i, i = 1, 2$ is simply $\text{supp}(r_1) = P[y = \omega_1 \cap X < c] = \phi_1(c)$ and $\text{supp}(r_2) = P[y = \omega_2 \cap X \geq c] = 1 - \phi_2(c)$.

Similarly, the expected confidence for rules r_1 and r_2 are respectively:

$$\text{conf}(r_1) = P[y = \omega_1 | X < c] = \frac{2\phi_1(c)}{\phi_1(c) + \phi_2(c)}, \tag{9}$$

$$\text{conf}(r_2) = P[y = \omega_2 | X \geq c] = \frac{2\bar{\phi}_2(c)}{\bar{\phi}_1(c) + \bar{\phi}_2(c)}, \tag{10}$$

where by $\bar{\phi}(\cdot)$ we denote the complementary CDF of the distribution $\phi(\cdot)$ so that $\bar{\phi}_1(\cdot) = 1 - \phi_i(\cdot)$.

From the above we have the following corollary.

Corollary: *Given a sufficiently large 1-dimensional dataset D with patterns drawn without any bias from two Gaussian distributions $\mathcal{N}(\mu_1, \sigma^2)$, and $\mathcal{N}(\mu_2, \sigma^2)$, the rule set derived by applying QARMA required to produce all widest rules that apply on D with minimum support threshold $s_m \leq \min_{i=1,2}\{\text{supp}(r_i)\}$ and minimum confidence threshold $c_m \leq \min_{i=1,2}\{\text{conf}(r_i)\}$ the resulting rules will always contain a rule that is satisfied by an instance t and provides as*

consequent the same class as the decision of any classifier whose decision is in agreement with that of the optimal Bayesian classifier on the dataset D.

4 Feasibility study of dynamical systems for data imputation tasks in industrial environments

4.1 Motivation

The fundamental problem of missing data reconstruction among sensor signals captured from a physical process has been extensively studied in the literature. The most intuitive case of imputation is scaffolded from interpolation techniques. Such an approach, albeit exploiting temporal smoothness pertaining to a single data stream and resonating computational efficiency, fails to incorporate information leveraged from correlations among different sensor streams [51]. This cogently justifies the utilization of approaches based on singular value decomposition (SVD) which aim to deduce linear correlations among measurement streams and, thus, reconstruct missing values in a data stream from the remaining ones. By means of iterative low-rank decomposition and data imputation, missing values can be inferred [52]. Nevertheless, such factorizations do not account for temporal smoothness. It is to be noted that SVD-based schemes are characterized by an inherent inability to effectively address the existence of invariances of low-rank embeddings in the data. Furthermore, perils which deteriorate their performance arise in the presence of transient content among the measurement streams [53].

4.2 Dynamical Systems

In this context, and motivated by the intrinsic spatiotemporal cross-correlations among measurement streams in industrial environments [51], we have studied the feasibility of interpretable dynamical systems as the formal machinery for addressing the sensor data imputation problem in the presence of non-linear and multi-scale spatiotemporal behavior of measurement trajectories [54]. In particular, we investigate the behavior of hidden variable identification, deduction of dynamics, and imputation of potentially missing values, by exploiting spatiotemporal correlations among sensor data streams.

The utilization of dynamical systems as an interpretable scheme for gaining insight into the underlying industrial data dynamics stems from their ability to extract knowledge from measurement data based on the spatiotemporal synergy among the respective trajectories. In this context, three variants of dynamical systems are studied: 1) linear dynamical systems (LDS), 2) switching linear dynamical systems (SLDS), and 3) recurrent switching linear dynamical systems (rSLDS). Each of the aforementioned systems can be graphically described by a Dynamic Bayesian Network (DBN), as illustrated in Figure 7.

In the case of LDS, the dynamics governing the handled data are captured by latent variables through linear mappings. The equations which describe LDS are shown in Eqs. (11) and (12), as follows

$$h_{t+1} = Fh_t + e_h,\ e_h \sim \mathcal{N}(0, \Lambda), \tag{11}$$

$$z_t = Gh_t + e_z,\ e_z \sim \mathcal{N}(0, \Sigma), \tag{12}$$

where the former expresses temporal correlations among latent variables of measurement streams and the latter captures spatial interactions among different measurements in the same

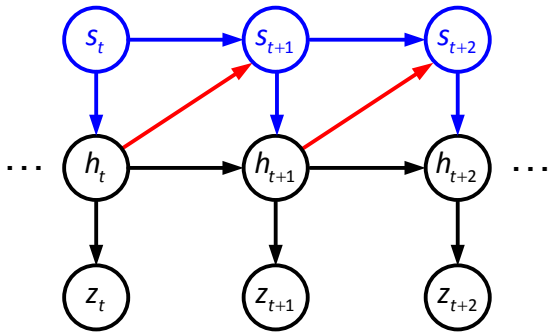

Figure 7: Dynamic Bayesian Network for LDS (black), SLDS (black, blue), and rSLDS (black, blue, red) [54].

time-step. Gaussian transition and observation noises are assumed, in order to account for the inherent variation in observed and latent variables caused by sundry factors which cannot be predicted from the deterministic transitions and emissions.

Based on the factorized form of Figure 7, the joint distribution of latent and observation variables is expressed as

$$P\left(\{h_t, z_t\}_{t=1}^{T}\right) = P\left(h_1\right) \prod_{t=1}^{T-1} P\left(h_{t+1} | \, h_t\right) \prod_{t=1}^{T} P\left(z_t | \, h_t\right). \tag{13}$$

The parameters which need to be estimated consist of the entries in linear mappings F and G, as well as noise covariances Λ and Σ. The latter are assumed to constitute diagonal matrices. Term $p\left(h_{t+1} | \, h_t\right)$ in Eq. (13) can be expressed as

$$\begin{aligned} p\left(h_{t+1} | \, h_t\right) \propto & \det\left(\Lambda\right)^{-1/2} \times \\ & \exp\left(-\frac{1}{2}(h_{t+1} - Fh_t)^T \Lambda^{-1} \left(h_{t+1} - Fh_t\right)\right). \end{aligned} \tag{14}$$

The remaining terms can be straightforwardly deduced. Learning parameters $\theta = \{F, \Lambda, G, \Sigma\}$ can be achieved by means of Expectation-Maximization algorithm [55] or in a Bayesian framework using blocked Gibbs sampling by setting conjugate prior distributions over all parameters [56]. Since it is plausible that measurement trajectories may not be sufficiently summarized by a single LDS due to temporal structural changes, the use of SLDS which switches among a predetermined number of LDS can be made [57]. Discrete latent variable s_t serves as the switching enabler for each respective LDS, which is now characterized by its unique parameters F_{s_t}, G_{s_t}, Λ_{s_t}, and Σ_{s_t}. It is common that parameter sharing pertaining to emission matrix G_{s_t} and covariance matrix Σ_{s_t} among the different LDS is imposed. Finally, SLDS is extended to rSLDS through the explicit incorporation of a dependence of discrete hidden variable s_{t+1} on continuous latent variable h_t, which introduces non-Gaussian potentials $\psi_{t,t+1}\left(h_t, s_{t+1}\right) = p\left(s_{t+1} | h_t\right)$ with further implications in the sampling of hidden variables and Bayesian parameter learning [58]. It should be stated that the increased expressive power of SLDS and rSLDS comes with the cost of elevated computational complexity which may pose a challenge for real-time applications.

The iterative scheme for missing measurement imputation employed in [54] is considered for the following simulation experiments.

4.3 Simulation Results

For the simulation setup, a power grid monitoring scenario is considered. A number of deployed Phasor Measurement Units (PMUs), able to monitor the state of various grid components, transmit their measurements to a Phasor Data Concentrator (PDC). Assuming an underlying cellular topology for the support of time-critical communication, each PDC is co-located with an associated base station (BS) and a closest BS association scheme is adopted.

Data collected by 6 PMUs as part of the EPFL Smart Grid project [59] are utilized as input. From each PMU, we retain 3 measurement streams which correspond to three-phase voltage magnitudes. For the simulation setup of the network topology, we consider a Poisson Point Process (PPP) for the BSs and their co-located PDCs with intensity λ_b, while the PMUs are randomly spread over the area until all PDCs are associated with at least one PMU, i.e., to achieve traffic saturation conditions. Regarding resource allocation, a dedicated set of uplink time-frequency slots has been considered for the PMU transmissions. The closed-form expression for the PMU outage probability which models the occlusions attributed to the imperfect sensor connectivity is derived in [60].

Two distinct simulation setups have been considered where missing data result from either the lossy PMU wireless transmissions (resulting in data decoding failures at the PDC) or the creation of synthetic occlusions due to a denial-of-service attack or hardware sensor failures. In the latter, the measurement channels exhibiting consecutive dropouts or uniformly dispersed missing data are selected at random. For all experiments, we have utilized the respective codes which are provided in [61] and [62] for LDS, SLDS, and rSLDS. Each experiment was repeated 20 times and we averaged the achieved performance. Reported results stem from the best average performance achieved by the union of the aforementioned dynamical systems in the examined case scenarios. To quantify the quality of model fitting and data imputation, the root mean squared error (RMSE) is employed as

$$\mathrm{RMSE}\left(\{z_{t,k}, \hat{z}_{t,k}\}\right) = \sqrt{\frac{1}{|\mathcal{O}|} \sum_{(t,k)\in\mathcal{O}} \left(z_{t,k} - \hat{z}_{t,k}\right)^2}. \tag{15}$$

where $z_{t,k}$ and $\hat{z}_{t,k}$ pose a dual role; in the case of examining model fitting, they correspond to the decoded measurements and predicted values by the dynamical model, respectively, at each time-step t where observed values exist, while for quantifying the quality of data imputation they pertain to the ground truth of occluded values and the imputed values, respectively, at each time-step t where missing values exist. Additional information related to hyperparameter tuning is provided in [54].

Figure 8 illustrates the RMSE performance with respect to the SINR threshold γ_{th}. It can be observed that the RMSE levels increase with increasing γ_{th}, as the aggregate interference leads to significant signal degradation and results in a high number of PMU links in SINR outage. Subsequently, the increased outage probability in the high γ_{th} regime, impacts the achieved RMSE and leads to degraded performance in the model fitting process. Figure 9 depicts the evolution of the RMSE performance for varying BS intensity λ_b. Note that each PDC is considered to be co-located with its respective BS, therefore λ_b also represents the PDC intensity. We observe that model fitting performance improves (i.e., the RMSE decreases) with increasing λ_b. This can be intuitively explained as follows. When the BSs/PDCs are dense

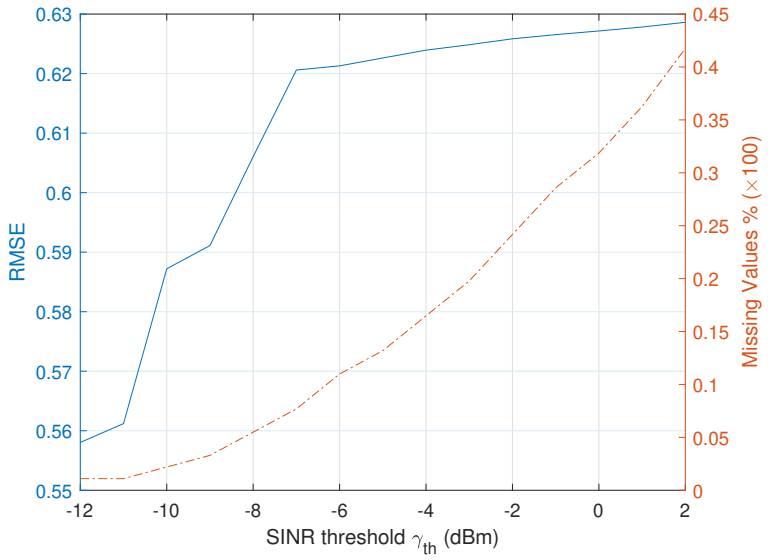

Figure 8: Model fitting performance for varying levels of SINR threshold γ_{th} (parameter settings: $\lambda_b = 2\text{BS/km}^2$, $\rho = -60\text{dBm}$, $\sigma^2 = -90\text{dBm}$) [54].

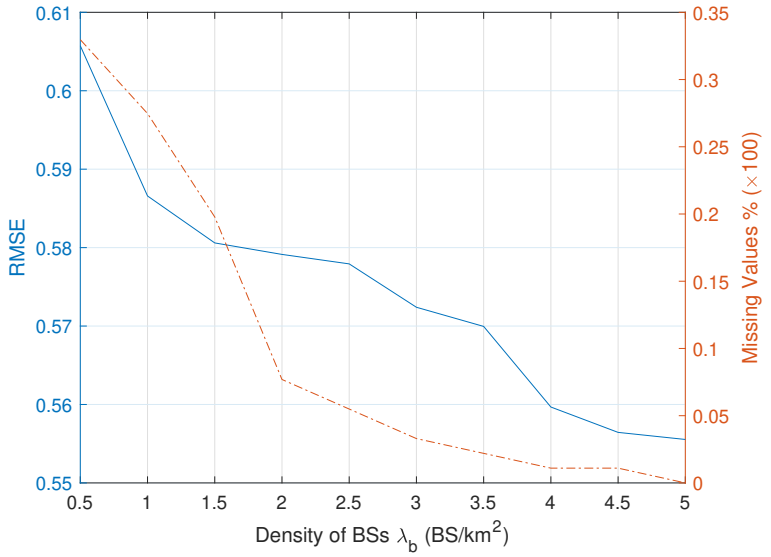

Figure 9: Model fitting performance for varying levels of BS intensity λ_b (parameter settings: $\gamma_{\text{th}} = -8\text{dBm}$, $\rho = -60\text{dBm}$, $\sigma^2 = -90\text{dBm}$) [54].

enough, the distance between a generic PMU and its corresponding serving BS/PDC decreases, resulting in higher quality of the received measurements and lower outage probability levels.

In contrast to the previous simulation setup, we now assume an ideal network configuration with high data-decoding accuracy for the PMU transmissions. Synthetic dropouts can be made by uniformly selecting space-time points for occlusion, which can be attributed to the effect of denial-of-service or PMU hardware failures.

Figure 10 has been devised by conducting two types of experiments, investigating primarily

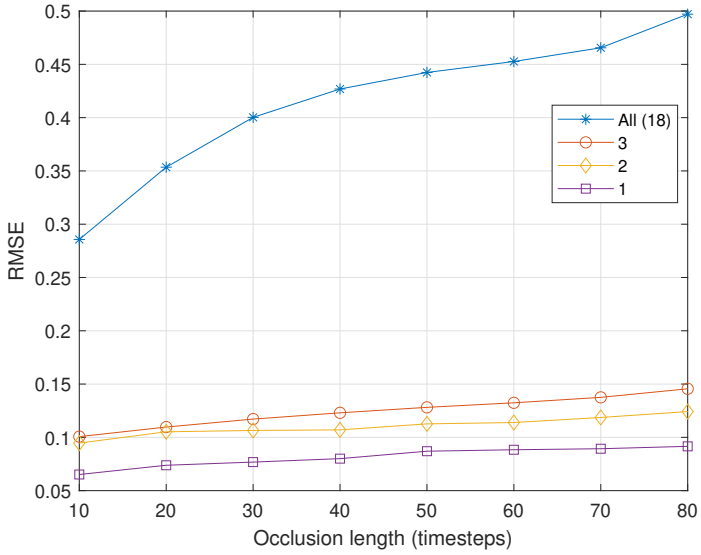

Figure 10: Imputation performance for varying occlusion length [54].

the effect of ensuing occlusions: (*a*) we randomly opted for 1, 2, and 3 measurement streams to exhibit consecutive missing values of varying length starting at a uniformly distributed common point in time, and (*b*) we opted for all streams to simultaneously report missing values. The visual assessment of Figure 10 suggests that reconstruction of missing values in the case of experiment (*a*) does not suffer from the limitations imposed by potentially unfavorable network configuration parameters. The imputation mechanism takes advantage of spatiotemporal correlations based on the perfectly decoded measurement streams which are consistent with the underlying physical process. Nevertheless, even under such ideal network conditions, the recovery of missing data in experiment (*b*) suffers from the fact that in the absence of any measurement stream for an extended period of time there is non-existent spatial information in each time-step for the dynamical model to exploit. Moreover, the temporal information is limited to the relatively distant time-steps with recorded measurement values. Hence, the predicted values are mainly influenced by the outcome of linear interpolation from the initial stage of data imputation process, showing significant mismatch with respect to the ground truth values.

Extended simulation results on the feasibility of dynamical systems for data imputation and detailed assessment under favorable and non-favorable network connectivity conditions are provided in [54]. It can be deduced that when network parameters are such that significant stream-wise distortions are present among the received measurement values, the utilized dynamical systems are not capable of capturing the underlying spatiotemporal interactions and, thus, the validity of estimating missing values is curtailed.

5 Alarm classification in Industrial IoT applications

In this section, we introduce how rare alarms can be detected within IoT applications. In D4.1, we introduced the Coupled Markov Modulated Poisson Process (CMMPP) traffic model, which describes the packets arrival efficiently in terms of accuracy and complexity [63]. It results in a set of devices, which can be active or silent at a time. As shown in Figure 11, active devices can be found in data or alarms states according to an unknown event.

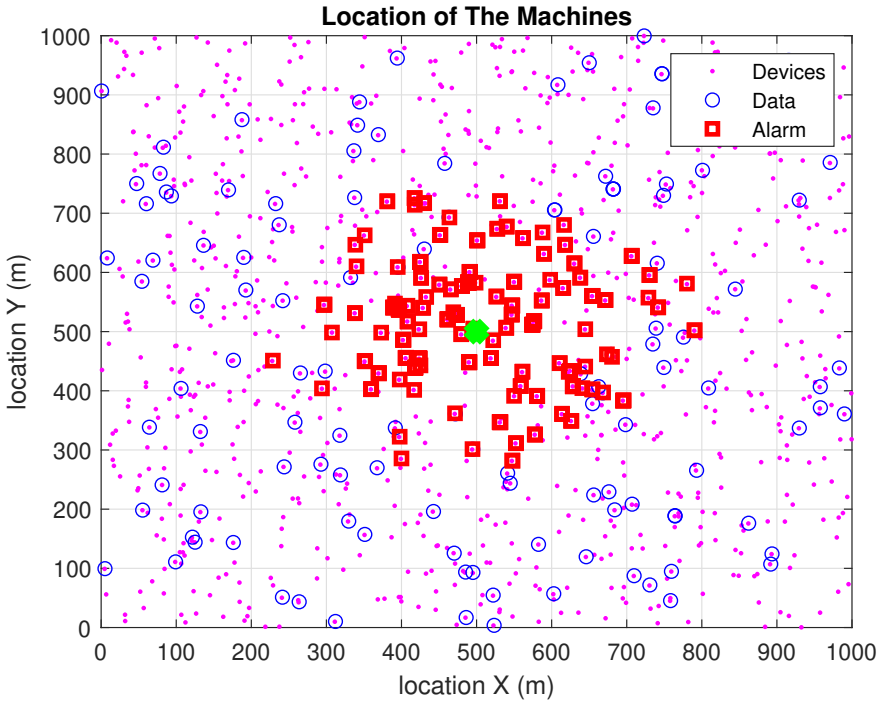

Figure 11: A set of devices, which can be active or silent. Active devices can be in either data or alarm state.

5.1 Device Classification via Support Vector Machines

Our classification problem is considered as a typical binary classification problem. Binary classification is a type of supervised learning, where a model is developed to classify some candidates among two classes. First step in device classification is collecting an appropriate dataset with desired features and labels, that reflects different scenarios of the application for a period of time. Features are the input data that the classifier should take into consideration and learn their pattern and how they are related to their corresponding labels. In MTDs classification, features are position coordinates of the devices, while labels are the data or alarm classes. Usually, binary classification problems are considered as finding the best decision boundary that separates the classes correctly. The optimum decision boundary can be found using SVM algorithm. Define training points $T = (\vec{x}_i, z_i)$, where $\vec{x}_i$ are the features, and z_i are the labels $(-1, 1)$. Then, define the decision hyperplane as b and a normal vector $\vec{w}$

perpendicular to the hyperplane. Finally, define a point $\vec{x}$ to be on the hyperplane, so that:

$$\vec{w}^T \vec{x} = -b. \tag{16}$$

The SVM aims to maximize the width between the nearest features from one class to the other (support vectors of each class) [64]. Fortunately, this optimization problem is convex, which can be solved using Lagrange multiplier theorem with any quadratic programming to find the solution:

$$L = \frac{1}{2}\|w\|^2 - \sum \alpha_i \left[z_i(\vec{w}.\vec{x}_i + b) - 1 \right], \tag{17}$$

where α_i is the Lagrange multiplier. Classify class 1 if:

$$\sum \alpha_i.y_i.\vec{x_i}.\vec{x} + b \leq 0. \tag{18}$$

By inserting features (device coordinates) and labels (class 1 or class 2) into the classifier, we find the optimal boundary between the two classes. However, this classifier is only applicable to feature points that are linearly separable. The CMMPP classes are non-linearly separable. Thus, to avoid this problem we use transformation kernels (ϕ) to map the feature points into higher dimensions, where they can be linearly separable:

$$\phi : \vec{x} \rightarrow \phi(\vec{x}). \tag{19}$$

The radial basis function (RBF) maps the data into an infinite dimensional Hilbert space [64]. It is very simple and fast to remap feature points using such kernel transformations; here, we employ the Gaussian RBF kernel given as:

$$K(\vec{x}, \vec{x'}) = e^{-\frac{(\vec{x}-\vec{x'})^2}{2\sigma^2}}, \tag{20}$$

where $\vec{x}$ and $\vec{x'}$ are two features. The RBF SVM can not only extract the pattern of devices efficiently, but also classify them according to their priorities.

5.2 Performance Evaluation Metrics

There are many appropriate evaluation metrics, which are suitable for skewed data applications (rare alarms) such as [65], [66]:

- **The confusion matrix** is the most important evaluation method, which illustrates the number of correct and wrong classifications in each class.
- **The classification accuracy** describes the overall performance of the classifier.
- **The precision (P) and recall (R).** The former describes the ratio of true predicted samples for each class to the total predicted samples of that class, while the later describes the ratio of true predicted samples for each class to the total actual samples of that class.
- **The f1-score ($f1s$)** combines the P and R measurements via harmonic mean, resulting in a percentage near to the minimum between them.

$$f1s = \frac{2\,R\,P}{R+P}. \tag{21}$$

5.3 Simulation Results

We initially collect the training set by running multiple CMMPP and M-CMMPP (multiple background events) simulations with uniformly distributed epicenters, variances, and intervals of background processes. Then, we pre-process the data by balancing, normalizing and then removing redundancies to ease the training phase and extract the correct features. Afterwards, we compare different classification algorithms to a new CMMPP/M-CMMPP model as shown in Table 1. We observe that a polynomial kernel (degree = 6) SVM provides good performance in CMMPP case, but fails in the network congestion case. An artificial neural network (ANN) architecture with 4 hidden layers (16, 32, 8, and 4 neurons) works very well in both cases in terms of data and alarm classification, where it introduced the lowest errors (only 12 alarm errors in case of CMMPP and 2 alarm errors in case of M-CMMPP) in classifying alarms compared to all classifiers. However, it has relatively large number of errors classifying data devices.

The radial basis function SVM, decision trees (DT), and random forests (RF) produce better results than ANN in classifying data devices and almost similar results as ANN in classifying alarm devices. The radial basis function SVM is the simplest algorithm among them and works extremely fast. It produces a recall of 0.87 and 0.88 for data and alarm classification, respectively in case of CMMPP traffic and a recall of 0.96 and 0.97 for data and alarm classification in case of M-CMMPP traffic. These small number of errors reflect the strength of the rbf SVM classifier and introduces low amount of errors will prioritizing the devices. Hence, rbf SVM is the best algorithm in terms of accuracy and complexity to classify the IoT devices.

Table 1: Confusion matrix, accuracy, precision, recall and f1-score for CMMPP device classification (support: 544 data and 113 alarm) and M-CMMPP device classification (support: 539 data and 153 alarm).

	Algorithm	Conf. Mat.	Acc.	$P\&R$	$f1-Sc.$
CMMPP	Poly. SVM	354 190 7 106	0.90	0.98 0.65 0.36 0.94	0.78 0.52
	RBF SVM	474 70 14 99	0.87	0.97 **0.87** 0.59 **0.88**	0.92 0.70
	ANN	462 82 12 101	0.86	0.97 0.85 0.55 0.89	0.91 0.68
	DT	478 66 17 96	0.87	0.97 0.88 0.59 0.85	0.92 0.70
	RF	473 71 15 98	0.87	0.97 0.87 0.58 0.87	0.92 0.70
M-CMMPP	Poly. SVM	539 0 153 0	0.85	0.78 1.00 0.00 0.00	0.88 0.00
	RBF SVM	518 21 5 148	0.97	0.99 **0.96** 0.88 **0.97**	0.97 0.92
	ANN	486 53 2 151	0.92	1.00 0.90 0.74 1.00	0.95 0.85
	DT	518 21 8 145	0.96	0.98 0.96 0.87 0.95	0.97 0.91
	RF	517 22 6 147	0.96	0.99 0.96 0.87 0.96	0.97 0.91

6 Machine Learning Limitations for Data Imputation in an Industrial Environment

Our approach to fault detection in an industrial environment combines data imputation techniques and a classifier to detect those faults. This allows us to build a system that is robust enough to cope with sensor failures in a dynamic environment. The imputation mechanism we investigate is based on the Generative Adversarial Networks framework, and in particular on the work in [67]. We extend this approach to address both temporary and persistent missing data and we evaluate its performance based on the impact on the anomaly detection classifier rather than using the RMSE. This classifier is trained without missing data using the Extended Tennessee Eastman dataset training files (https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/6C3JR1).

There are multiple factors that affect the quality of imputation. For example, the size of the system (e.g., number of sensors) and the number of possible anomalies that should be detected by the classifier affect the complexity of the generator, and hence, affect the required training time and training dataset size. Additionally, the dynamical nature of the system (e.g., the mean changing over time) will require a more complex model to adjust to the fluctuations over time. Therefore, we cannot build a generic model that fits multiple scenarios. Each system has to be studied independently by collecting a "large enough" dataset that will allow us to understand the correlation between multiple input variables for different scenarios.

Considering the above-mentioned properties and the fact that unexpected events can happen in an industrial setup (e.g., two machines crashing into each other, human intervention), the main limitation for machine learning algorithms is the availability of training data and the time needed to collect such a dataset. Therefore, the training process should be iterative, which would allow us to improve the models over time. This, however, means that we cannot really rely on the first iterations of the machine learning algorithm due to its limited knowledge. Nevertheless, a way to circumvent this issue is to collect data from a simulated industrial environment that mirrors the known behavior of the system. This approach allows us to build a more reliable initial model, which still has to go through multiple iterations after collecting real system data.

Additionally, some sensors might be more important for the classifier than others. For example small reconstruction errors in some sensors might not affect the classifier, while for other sensors they could have a large impact.

7 Conclusion

In this deliverable, we have presented the most relevant fundamental limits identified in different activities carried out during FIREMAN. In that respect, we have dealt with more practical questions about network architecture and fundamental trade-offs (related to WP6), indications of how QARMA can be used as a tool for the fundamental limits of explainability (WP5) and fundamental questions about data imputation considering wireless transmissions (WP3) and data fusion in industrial settings (WP4). These results were obtained to theoretically support such activities and provide a better understanding of the limitations of FIREMAN, which will guide the practical deployments in WP6.

References

[1] A. Narayanan, A. S. De Sena, D. Gutierrez-Rojas, D. C. Melgarejo, H. M. Hussain, M. Ullah, S. Bayhan, and P. H. Nardelli, "Key advances in pervasive edge computing for industrial internet of things in 5g and beyond," *IEEE Access*, vol. 8, pp. 206 734–206 754, 2020.

[2] Y. Miao, G. Wu, M. Li, A. Ghoneim, M. Al-Rakhami, and M. S. Hossain, "Intelligent task prediction and computation offloading based on mobile-edge cloud computing," *Future Generation Computer Systems*, vol. 102, pp. 925–931, 2020. [Online]. Available: https://doi.org/10.1016/j.future.2019.09.035

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[4] P. Popovski, O. Simeone, F. Boccardi, D. Gunduz, and O. Sahin, "Semantic-effectiveness filtering and control for post-5g wireless connectivity," *arXiv preprint arXiv:1907.02441*, 2019.

[5] D. Gutierrez-Rojas, M. Ullah, I. T. Christou, G. Almeida, P. H. Nardelli, D. Carrillo, J. M. Sant'Ana, H. Alves, M. Dzaferagic, A. Chiumento *et al.*, "Three-layer approach to detect anomalies in industrial environments based on machine learning," *arXiv preprint arXiv:2004.09097*, 2020.

[6] X. Hu, L. Wang, K. Wong, M. Tao, Y. Zhang, and Z. Zheng, "Edge and central cloud computing: A perfect pairing for high energy efficiency and low-latency," *IEEE Trans. Wireless Commun.*, vol. 19, no. 2, pp. 1070–1083, 2020.

[7] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, 2019.

[8] U. Saleem, Y. Liu, S. Jangsher, X. Tao, and Y. Li, "Latency minimization for d2d-enabled partial computation offloading in mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4472–4486, 2020.

[9] Y. Ai, M. Peng, and K. Zhang, "Edge computing technologies for Internet of Things: a primer," *Digital Communications and Networks*, vol. 4, no. 2, pp. 77–86, 2018.

[10] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," *MobiSys 2014 - Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 68–81, 2014.

[11] Z. Deng, Z. Cai, and M. Liang, "A multi-hop vanets-assisted offloading strategy in vehicular mobile edge computing," *IEEE Access*, vol. 8, pp. 53 062–53 071, 2020.

[12] T. Dang and M. Peng, "Joint radio communication, caching, and computing design for mobile virtual reality delivery in fog radio access networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 7, pp. 1594–1607, 2019.

[13] J. Kwak, Y. Kim, L. B. Le, and S. Chong, "Hybrid content caching in 5g wireless networks: Cloud versus edge caching," *IEEE Trans. Wireless Commun.*, vol. 17, no. 5, pp. 3030–3045, 2018.

[14] J. Ren, Y. Ruan, and G. Yu, "Data transmission in mobile edge networks: Whether and where to compress?" *IEEE Commun. Lett.*, vol. 23, no. 3, pp. 490–493, 2019.

[15] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, "Edge computing security: State of the art and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1608–1631, 2019.

[16] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, "Data security and privacy-preserving in edge computing paradigm: Survey and open issues," *IEEE Access*, vol. 6, pp. 18 209–18 237, 2018.

[17] M. Yahuza, M. Y. I. B. Idris, A. W. B. A. Wahab, A. T. Ho, S. Khan, S. N. B. Musa, and A. Z. B. Taha, "Systematic review on security and privacy requirements in edge computing: State of the art and future research opportunities," *IEEE Access*, 2020.

[18] D. Liu, Z. Yan, W. Ding, and M. Atiquzzaman, "A survey on secure data analytics in edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4946–4967, 2019.

[19] T. Wang, G. Zhang, A. Liu, M. Z. A. Bhuiyan, and Q. Jin, "A secure iot service architecture with an efficient balance dynamics based on cloud and edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4831–4843, 2018.

[20] R. Ullah, M. A. U. Rehman, and B.-S. Kim, "Design and implementation of an open source framework and prototype for named data networking-based edge cloud computing system," *IEEE Access*, vol. 7, pp. 57 741–57 759, 2019.

[21] R. W. Lucky, "Automatic equalization for digital communication," *Bell System Technical Journal*, vol. 44, no. 4, pp. 547–588, 1965.

[22] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[23] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[24] C. Esposito, A. Castiglione, F. Pop, and K. R. Choo, "Challenges of connecting edge and cloud computing: A security and forensic perspective," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 13–17, 2017.

[25] J. Zhang, H. Lee, and E. Modiano, "On the robustness of distributed computing networks," in *2019 15th International Conference on the Design of Reliable Communication Networks (DRCN)*, 2019, pp. 122–129.

[26] Y. Sharma, B. Javadi, W. Si, and D. Sun, "Reliability and energy efficiency in cloud computing systems: Survey and taxonomy," *Journal of Network and Computer Applications*, vol. 74, pp. 66 – 85, 2016.

[27] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, p. 350–361, Aug. 2011.

[28] H. Yang, W. Bai, A. Yu, Q. Yao, J. Zhang, Y. Lin, and Y. Lee, "Bandwidth compression protection against collapse in fog-based wireless and optical networks," *IEEE Access*, vol. 6, pp. 54 760–54 768, 2018.

[29] R. Makhlouf, "Cloudy transaction costs: a dive into cloud computing economics," *Journal of Cloud Computing*, vol. 9, no. 1, 2020.

[30] H. Yuan and M. Zhou, "Profit-maximized collaborative computation offloading and resource allocation in distributed cloud and edge computing systems," *IEEE Transactions on Automation Science and Engineering*, pp. 1–11, 2020.

[31] S. Wang, X. Zhang, Z. Yan, and W. Wenbo, "Cooperative edge computing with sleep control under nonuniform traffic in mobile edge networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4295–4306, 2019.

[32] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[33] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2020.

[34] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing," *IEEE network*, vol. 32, no. 1, pp. 96–101, 2018.

[35] F. Liang, W. Yu, X. Liu, D. Griffith, and N. Golmie, "Towards edge-based deep learning in industrial internet of things," *IEEE Internet of Things Journal*, 2020.

[36] M. Murshed, E. Verenich, J. J. Carroll, N. Khan, and F. Hussain, "Hazard detection in supermarkets using deep learning on the edge," *arXiv preprint arXiv:2003.04116*, 2020.

[37] C.-F. Lai, W.-C. Chien, L. T. Yang, and W. Qiang, "Lstm and edge computing for big data feature recognition of industrial electrical equipment," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2469–2477, 2019.

[38] S. Tuli, N. Basumatary, S. S. Gill, M. Kahani, R. C. Arya, G. S. Wander, and R. Buyya, "Healthfog: An ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated iot and fog computing environments," *Future Generation Computer Systems*, vol. 104, pp. 187–200, 2020.

[39] Y. Wang, M. Liu, P. Zheng, H. Yang, and J. Zou, "A smart surface inspection system using faster r-cnn in cloud-edge computing environment," *Advanced Engineering Informatics*, vol. 43, p. 101037, 2020.

[40] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[41] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[42] N. H. Tran, W. Bao, A. Zomaya, N. M. NH, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1387–1395.

[43] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys Tutorials*, p. 1, 2020.

[44] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[45] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang, "Federated learning-based computation offloading optimization in edge computing-supported internet of things," *IEEE Access*, vol. 7, pp. 69 194–69 201, 2019.

[46] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[47] Z. M. Fadlullah and N. Kato, "Hcp: Heterogeneous computing platform for federated learning based collaborative content caching towards 6g networks," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2020.

[48] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23 920–23 935, 2020.

[49] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Differentially private asynchronous federated learning for mobile edge computing in urban informatics," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2134–2143, 2020.

[50] I. T. Christou, "QARMA: Parallel Mining of Quantitative Association Rules for Maximal Coverage of Multidimensional Datasets," In review, 2021.

[51] L. Li, J. McCann, N. S. Pollard, and C. Faloutsos, "Dynammo: Mining and summarization of coevolving sequences with missing values," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 507–516. [Online]. Available: https://doi.org/10.1145/1557019.1557078

[52] N. Srebro and T. Jaakkola, "Weighted low-rank approximations," in *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ser. ICML'03. AAAI Press, 2003, p. 720–727.

[53] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019.

[54] T. A. Alexopoulos, C. Kalalas, and G. N. Korres, "On the imputation of power system measurement streams with imperfect wireless communication," in *2020 2nd Global Power, Energy and Communication Conference (GPECOM)*, 2020, pp. 302–307.

[55] Z. Ghahramani and G. E. Hinton, "Parameter estimation for linear dynamical systems," Tech. Rep., 1996.

[56] A. Wills, T. B. Schön, F. Lindsten, and B. Ninness, "Estimation of linear systems using a gibbs sampler*," *IFAC Proceedings Volumes*, vol. 45, no. 16, pp. 203 – 208, 2012, 16th IFAC Symposium on System Identification. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667015379520

[57] V. Pavlovic, J. M. Rehg, and J. MacCormick, "Learning switching linear models of human motion," in *Advances in Neural Information Processing Systems*, T. Leen, T. Dietterich, and V. Tresp, Eds., vol. 13. MIT Press, 2001, pp. 981–987. [Online]. Available: https://proceedings.neurips.cc/paper/2000/file/ca460332316d6da84b08b9bcf39b687b-Paper.pdf

[58] S. Linderman, M. Johnson, A. Miller, R. Adams, D. Blei, and L. Paninski, "Bayesian Learning and Inference in Recurrent Switching Linear Dynamical Systems," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. Fort Lauderdale, FL, USA: PMLR, 20–22 Apr 2017, pp. 914–922. [Online]. Available: http://proceedings.mlr.press/v54/linderman17a.html

[59] "EPFL Smart Grid. [Online]," http://nanotera-stg2.epfl.ch/data/, Accessed: 18 January 2021.

[60] C. Kalalas and J. Alonso-Zarate, "Sensor data reconstruction in industrial environments with cellular connectivity," in *2020 IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (IEEE PIMRC '20)*, August 2020.

[61] "PyLDS: Bayesian inference for linear dynamical systems. [Online]. Available:," https://github.com/mattjj/pylds, Accessed: 18 January 2021.

[62] "SSM: Bayesian learning and inference for state space models. [Online]. Available:," https://github.com/slinderman/ssm, Accessed: 18 January 2021.

[63] M. Laner, P. Svoboda, N. Nikaein, and M. Rupp, "Traffic models for machine type communications," in *ISWCS 2013; The Tenth International Symposium on Wireless Communication Systems*, 2013, pp. 1–5.

[64] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. USA: Cambridge University Press, 2008.

[65] A. Ng, *Machine learning yearning*, 2017.

[66] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427 – 437, 2009.

[67] J. Yoon, J. Jordon, and M. Van Der Schaar, "Gain: Missing data imputation using generative adversarial nets," *arXiv preprint arXiv:1806.02920*, 2018.