UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Smart and Efficient Sensor Networks Operation for 5G and Beyond Ecosystems

Ahmad Mohammad El Sayed

Universitat Politècnica de Catalunya

Optical Communications Group

Smart and Efficient Sensor Networks Operation for 5G and Beyond Ecosystems

Ahmad Mohammad El Sayed

ahmad.mohammad@upc.edu

A thesis presented in partial fulfillment of the
requirements for the degree of

Philosophy Doctor

Advisor: Dr. Marc Ruiz
Co-advisor: Dr. Hassan Harb

September 2023

Advanced Broadband Communications Center (CCABA)

Universitat Politècnica de Catalunya (UPC)

C/ Jordi Girona, 1-3

Campus Nord, D4-213

08034 Barcelona, Spain

Acknowledgements

Firstly, I would like to thank my supervisors, Prof. Marc Ruiz and Prof. Hassan Harb, for their vital advice, continuous support, and patience during these three long years. I would also like to thank Prof. Luis Velasco, for his valuable input in many of the technical parts.

Secondly, I would like to thank my parents, Mohammad and Basima, for their support, and my sister Hiba for her encouragement, and all my friends. Most importantly, I would like to thank my wife, who endured with me the difficulties that accompanied this journey with unwavering support and love, without her I may have been able to achieve this. Thank you all, my achievement is as yours as it is mine.

To my wife, Mariam.

Abstract

Sensor Networks (SN) will play an integral role in Beyond 5G (B5G) ecosystems, especially for highly-distributed use cases and services such as Digital Twins (DT). Thus, the underlying transport network needs to provide connectivity between the highly dense and distributed SNs and the DT manager, that typically runs far from sensor data sources, i.e., in a centralized server. In view of this, critical requirements such as high data throughput, latency sensitivity communication, and data veracity and integrity assurance are essential to be provided by B5G networks to support DT services.

In order to meet such requirements, statistical and Artificial Intelligence (AI)-based SN data collection and analysis can be implemented to provide smart and efficient data transmission. By means of those procedures, SN data can be compressed and analyzed locally in order to reduce the total data volume to be conveyed in the centralized server. In addition, the inherent nature of the compression and analysis algorithms add privacy and security to the transmitted data without affecting integrity. The use of these kind of AI-based techniques opens the opportunity to perform Knowledge Transfer (KT) between DTs operating under the same tenant infrastructures. Since sharing raw data poses a privacy breach, AI-based methods allow interchanging relevant information while obfuscating critical details, thus enabling coordinated operation across differentiated segments.

This Ph.D. thesis aims at enhancing the operation of dense SNs, which are supported by an underlying transport infrastructure that includes edge/fog computing capabilities distributed among nodes. Through the application of statistical and AI-based methods and procedures, the proposed methods will target several objectives, such as reducing the volume of data transported through the network while keeping privacy and integrity, detecting anomalies or events in the collected data to provide early alarms and notifications, and facilitating the operation of services across several SN domains.

In more detail, the first objective is to develop methods to reduce the volume of collected sensor data through statistical and AI-based methods for data compression

and sampling rate manipulation. We proposed both statistical-based and Autoencoder (AE)-based approaches for compression, as wells as sampling rate adaptation method that works with either. Simulations of implemented algorithms on real world datasets showed a significant ability to reduce the volume of the data, reaching 1% of its original size in some cases, and leading the reduced energy consumption in the factor of one-tenth in case of sensors with limited energy availability.

The second objective is to develop methods for maintaining data veracity through employing AI-based anomaly detection methods at multiple levels of the network. Anomalies may arise due to a range of factors from faulty sensors to malicious attacks and detecting them can facilitate timely actions to avoid or mitigate their effect. We proposed two AE-based methods: one operating at the sensor level, and the other on the network level. Simulations of implemented algorithms on real world datasets showed more than 90% of accuracy in detecting anomalies in single sensor data analysis. Moreover, prompt detection of subtle anomalies spanning multiple sensors that could not be detected by single sensor data analysis was achieved.

Finally, the third objective is to investigate methods to improve multi-domain DT systems management and coordination through KT while preserving the privacy of each individual DT. We proposed an AE-based knowledge extraction method that extracts codified information about the state of the sharing DT and sends it to the target DT. The method showed that the target DT is able to use the codified and private information about the state of the sharing DT before the changes are apparent through their effect on its system.

Resumen

Las redes de sensores (SN) desempeñarán un papel esencial en los ecosistemas de redes 5G y más allá (B5G), especialmente para casos de uso y servicios altamente distribuidos como Gemelos Digitales (DT). Así pues, la red de transporte subyacente debe proporcionar conectividad entre las redes de sensores distribuidas y el sistema de gestión del DT, que suele estar lejos de las fuentes de datos de los sensores (típicamente, en un servidor centralizado). En vista de ello, es esencial que las redes B5G proporcionen requisitos críticos como alta capacidad de transmisión, comunicación sensible a la latencia y garantía de veracidad e integridad de los datos, todo ello para permitir servicios de DT.

Para cumplir con estos requisitos, se puede recurrir a la monitorización y el análisis de datos de SN basados en estadística e Inteligencia Artificial (IA) con el fin de proporcionar una transmisión de datos inteligente y eficiente. Mediante estos procedimientos, los datos generados por las SNs se pueden comprimir y analizar localmente para reducir el volumen total de datos que debe transmitirse al servidor centralizado. Además, la naturaleza inherente de los algoritmos de compresión y análisis añade privacidad y seguridad a los datos transmitidos sin afectar a su integridad. El uso de este tipo de técnicas basadas en IA abre la oportunidad de realizar Transferencia de Conocimiento (KT) entre DTs que operan sobre una infraestructura compartida. Dado que compartir datos sin procesar puede vulnerar la privacidad, los métodos basados en IA permiten intercambiar información relevante a la vez que se ofuscan detalles críticos, permitiendo así un funcionamiento coordinado y seguro entre segmentos diferenciados.

Esta tesis doctoral tiene como objetivo mejorar el funcionamiento de las SNs densas, que se necesitan de una infraestructura de transporte subyacente que incluya capacidades de computación entre los nodos. Mediante la aplicación de procedimientos estadísticos y basados en IA, los métodos propuestos perseguirán varios objetivos, como són: reducir el volumen de datos transportados a través de la red manteniendo la privacidad y la integridad; detectar anomalías o eventos en los datos recogidos para proporcionar alarmas y notificaciones tempranas; y facilitar la operación de servicios a través de varios dominios de SN.

Entrando en detalle, el primer objetivo consiste en desarrollar métodos para reducir el volumen de datos recogidos por los sensores mediante métodos estadísticos y basados en IA para la compresión de datos y la manipulación de la frecuencia de muestreo. Se han propuesto enfoques de compresión basados tanto en estadística como en el uso de autocodificadores (AE), así como un método de adaptación de la frecuencia de muestreo que funciona con ambos. Mediante simulación con conjuntos de datos reales, se demuestra una capacidad significativa para reducir el volumen de los datos, alcanzando el 1% de su tamaño original en algunos casos, y conduciendo a la reducción del consumo de energía a una décima parte en el caso de sensores con disponibilidad limitada de energía.

Finalmente, el tercer objetivo consiste en investigar métodos para mejorar la gestión y coordinación de sistemas de DT multidominio mediante KT, preservando al mismo tiempo la privacidad de cada DT individual. Se ha propuesto un método de extracción de conocimiento basado en AEs que extrae información codificada sobre el estado del DT que comparte y la envía al DT de destino. Resultados numéricos demuestran que el DT de destino es capaz de utilizar la información codificada y privada sobre el estado del DT que comparte antes de que un evento correlacionado entre ambos DTs afecte al DT de destino.

Table of Contents

List of Figures

List of Acronyms

The following list contains those acronyms that are extensively used in this document and become relevant for the correct understanding of the contributions of this Ph. D. thesis.

AAC	Adaptive Autoencoder-based Compression
AD	Anomaly Detection
AI	Artificial Intelligence
AE	Autoencoder
B5G	Beyond 5G
CH	Cluster Head
DT	Digital Twin
IBE	Index-Bit-Encoding
LS	Latent Space
ML	Machine Learning
MS-AGD	Multiple Sensor Anomaly Group Detection
PEKT	Privacy-Ensuring Knowledge Transfer
SRA	Sampling Rate Adjustment
SN	Sensor Networks
SS-AD	Single Sensor Anomaly Detection
TN	Transport Network
WSN	Wireless Sensor Network
ZIZO	Zoom-In Zoom-Out

Chapter 1

Introduction

1.1 Motivation

Sensor Networks (SN) is a technology paradigm consisting of miniaturized sensor devices with the ability to self-organize into an interconnected collective. These devices are becoming increasingly integral components within Beyond 5G (B5G) and Digital Twin (DT) systems, where one of their primary use cases lies. Within a DT deployment, the multitude of sensors continuously generate huge volumes of telemetry data at high velocity, in order to maintain an up-to-date digital facsimile of the physical system under observation. In addition, the underlying Transport Network (TN) provides the required connectivity between those highly dense and distributed sensors and the DT that typically runs in a centralized server, i.e., in the cloud. Such required connectivity amongst network elements and the DT poses critical challenges such as high bitrate and latency-sensitive communications. Moreover, other aspects such as reducing data redundancy amidst high throughput, avoiding bandwidth exhaustion due to data transmission, and performing anomaly detection within the gathered data to facilitate fault handling in a localized and real-time manner are foreseen as key features of intelligent data collection.

This intelligent data collection requires the convergence of resources as well as the utilization of Artificial Intelligence (AI) and Machine Learning (ML) techniques. AI and ML can be leveraged to intelligently govern network and computational resources, as well as the energy utilization of the sensor networks. Implementation can commence at the sensor level, integrated in software Sensor Agents (SA) enabling sensors to derive meaningful insights from raw data. SA outputs can then scale up to Cluster Agents (CA) with the ability to make decisions regarding sensors operating parameters using aggregate sensor data, and ultimately reaching the sink node i.e., the DT Manager, where knowledge can be extracted. Here, AI and ML have

the potential to optimize resource utilization and Anomaly Detection (AD) within the immense volumes of telemetry data generated by the multitude of sensors deployed within DT systems.

Moreover, within the same tenant infrastructure, there may exist combinations of DTs managing sub-physical systems of different natures, such as multiple water distribution systems or the power grid. While these DTs may have different operators and clients, relevant information about the operation of some may be valuable to others - for example, the power consumption profile of a water distribution system may be critical for the DT of the power grid for its own peak demand forecasts. This poses a privacy challenge, necessitating solutions to share essential information between DTs without violating privacy. Thus, Knowledge Transfer (KT) paradigms can be leveraged to facilitate this coordination, enabling a DT to disseminate key operating information of interest to others while obfuscating critical details like system infrastructure. In this way, DT and the underlying physical system can fully exploit resources for intelligent and coordinated operation across segments, vital for realizing many B5G use cases, while preserving privacy.

1.2 Goals of the Thesis

This Ph.D. thesis aims at enhancing the operation of dense SN, which are supported by a TN infrastructure that includes edge/fog computing capabilities distributed among SN and TN nodes. Through the application of statistical and AI-based methods and procedures, the proposed methods will target several objectives, such as reducing the volume of data transported through the network, detecting anomalies in the collected data to provide early alarms to problems, and facilitating the operation of services across several SN domains.

Specifically, this thesis has been organized in different goals:

G.1 – Develop methods to reduce the volume of collected sensor data

This is an important goal as SNs generate large volumes of data, which can be challenging to transport and process them efficiently. By reducing the volume of collected sensor data, the amount of network bandwidth required for data transport can be decreased, which in turn can lead to a reduction in energy consumption and improved network efficiency. This goal can be achieved through the application of statistical and AI-based methods and procedures and utilizing ML, which would enable intelligent management of sensor data at various levels of the network hierarchy.

G.2 – Develop anomaly detection methods in the collected sensor data

The ability to detect anomalies in real-time is crucial for providing early alarms to potential problems in SN. Anomalies may arise due to a range of factors, such as

environmental changes, faulty sensors, or malicious attacks. Detecting such anomalies can facilitate timely actions to avoid or mitigate the impact of these problems on the network. The use of AI-based and ML methods and procedures can enable efficient AD in SN. By training the ML models to detect patterns in the collected data, the network can identify and flag potential anomalies in real-time, leading to improved network reliability and performance.

G.3 – Develop methods to improve multi-domain network management and coordination

Another objective of the Ph.D. thesis is to improve global network management and coordination. Using KT-based methods, different DTs can coordinate their efforts through sharing information (data/models) among each other on a need-to-know basis, which helps to ensure client privacy. ML is used in order to extract from each system only the information that other systems need to know in order to perform coordinated actions successfully. In addition to privacy, this method also reduces the required bandwidth for such exchange of knowledge without reducing the effectiveness of coordination between multi-domain networks.

1.3 Methodology

The methodology employed in this thesis involves the implementation of several scenarios to assess the developed methods. To this end, real-life datasets were utilized, which were collected under diverse conditions and gathering sensor data from many sensors, frequently on a periodic basis. Datasets that contain such data, with labels and timestamps, were used. To simulate nodes generating the data generated by each sensor at the intervals indicated by the timestamp, Python code was written to create synthetic sensors and their agents (both SA and CA). Additionally, SN and TN architecture and topology were implemented, and each method was rigorously evaluated to ensure its effectiveness. By using real-life datasets and simulating network conditions, the developed algorithms were evaluated under various scenarios, leading to robust and reliable performance.

1.4 Thesis Outline

The remainder of this thesis is organized as follows.

Chapter 2 provides the needed background for an easier understanding of the research conducted throughout this PhD thesis. Specifically, background on SN, autoencoders (AE), compression, anomaly detection, and KT is provided. Chapter 3 reviews the state-of-the-art associated with the defined thesis goals. Chapter 4

details information about the DT use case, the reference SN and TN architecture, and the datasets used in the simulations. Chapter 5 is related to G.1 and explains how statistical-based methods on the level of the sensor and the network level were developed and evaluated to efficiently and intelligently reduce the size of sensor data through compression. This chapter is related with the work published in [IEESJ21]. Chapter 6 also relates to G.1 and describes how AE-based methods were developed for efficient data compression at the level of the sensor. It partly covers the work published in [Sensors23] and [ICTON23]. Chapter 7 deals with G.2 showing how ML-based methods were used to perform AD on sensor telemetry data on multiple levels of the network. Main results and contributions are also included in [Sensors23]. Chapter 8 focuses on goal G.3 and provides effective coordination between different DTs through KT that ensures privacy for the sharing party. Finally, Chapter 9 present the main concluding remarks and lessons learned.

1.5 Contributions and References from the Literature

For the sake of clarity and readability, references contributing to this PhD thesis are labelled using the following criteria: [<conference/journal acronyms Year(*by*) [. autonum]>], e.g., [IEESJ21] or [ICTON23]; in case of more than one contribution with the same label, a sequence number is added. The rest of the references to journal articles and conference papers are labelled with the initials of the first author's surname and year of publication, e.g., [Abd21] or [Aza20]. Additionally, references to books are labelled with the initials of the book title and year of publication, e.g., [EON16]. Finally, references to norms or standards are labeled with its identification, e.g., [ITU-T G.709].

Chapter 2

Background

The goal of this chapter is to give a brief overview of some of the main technologies that are involved in several parts of this PhD thesis. In section 2.1, we give a small background about SN and their development in recent years. Section 2.2 introduces the DT and its applications. Section 2.3 describes the basic concept of ML and its role in developing many technologies. Finally, Section 2.4 describes the main concepts related to KT.

2.1 Sensor Networks (SN)

In this section we provide a background about SN, their definition, architecture, applications, and how they evolved to become an integral part of many of today's technological infrastructures.

An SN consists of spatially distributed autonomous sensors that collaboratively monitor physical or environmental conditions. These sensors can measure parameters such as temperature, humidity, light, sound, motion, pressure, voltage, and various other variables depending on the application, and can have wired or wireless connections to the network [Yic08]. SN are extensively used for monitoring air quality, water quality, weather conditions, and natural disasters. Real-time data collected by these networks enables effective environmental management and early warning systems [Doi20].

An SN architecture typically comprises numerous sensors, a gateway or base station, and data processing units. Sensor nodes are individual devices equipped with sensors, processing units, and communication capabilities. They collect data from the environment and transmit it to the base station. Base stations act as the central entity in the network, responsible for collecting and aggregating data from multiple

sensor nodes. The base station communicates with external systems for data storage, analysis, and decision-making.

The development of the capabilities of sensor-based data collection has been truly remarkable. In the early stages, data collection was primarily performed manually. Researchers and field workers would physically collect data by making observations, taking measurements, and recording them manually. This approach was time-consuming, prone to human errors, and limited in terms of data volume and frequency. Later the process was automated by the invention of wired sensors, which improved scalability and enabled real-time data transmission, and centralized monitoring and management. However, it was constrained by the limitations of wired connections, such as installation challenges and limited coverage [Tub03]. The later stage of development was the Wireless Sensor Networks (WSN), which revolutionized data collection capabilities. It enabled wider deployments and offered wider flexibility in sensor placement [Sta08]. This enabled real-time collection from remote or inaccessible locations, and opened new possibilities for environmental monitoring, industrial applications, and smart infrastructure. These capabilities only increased with the emergence of the Internet of Things (IoT) [Mad15]. IoT integrated SN with internet connectivity and cloud-based platforms. This integration allowed for seamless data transmission, storage, and analysis on a large scale. IoT-enabled devices, equipped with sensors, became able to collect and transmit data autonomously, enabling real-time monitoring, predictive analytics, and data-driven decision-making across various domains [Gul22]. More recently, and with the rise of edge computing, data processing capabilities have been pushed closer to the data source. Edge sensors, embedded with computing power, gained the ability to perform data preprocessing and filtering tasks locally before transmitting relevant information to centralized systems or the cloud. This approach reduced latency, conserved network bandwidth, and enabled faster response times for critical applications [Wan19]. Finally, the integration of AI and ML algorithms has revolutionized sensor data collection capabilities. These technologies enabled automated data analysis, AD, and pattern recognition, allowing for real-time insights and proactive decision-making based on sensor data. AI and ML also enabled predictive and prescriptive analytics, empowering systems to optimize operations and anticipate future events [Abu14].

The applications of SN are numerous, and they are a key part of many future technologies such as *i)* Smart Cities: SNs enable the development of smart cities by monitoring traffic flow, parking availability, waste management, and energy consumption. This data helps optimize resource allocation and improve the quality of life for citizens [Sha21]. *ii)* Healthcare: SNs play a crucial role in remote patient monitoring, wearable devices, and telemedicine. They enable continuous monitoring of vital signs, medication adherence, and early detection of health issues [Abd20]. *iii)* Industrial Automation: SN are employed in industrial settings to monitor and

control manufacturing processes, optimize energy consumption, and ensure worker safety [Maj22].

Overall, the evolution of SN and their data collection capabilities has paved the way for a more connected, intelligent, and data-driven world, enabling enhanced monitoring, control, and decision-making across a wide range of applications.

2.2 Digital Twin (DT)

In this section, we will talk about the DT, its concept, key components, applications, the transformative impact it has on various industrial sectors, and how the contributions proposed in this PhD can be utilized in its scope.

A DT is a cutting-edge concept that has gained significant prominence in recent years, particularly within the field of industrial simulation and optimization, especially as current TN are being pushed toward the B5G era [Int20]. As industries strive to enhance productivity, efficiency, and decision-making processes, the integration of physical systems with their digital replicas has emerged as a powerful tool [Bar19].

A DT refers to a virtual replica or simulation of a physical object, system, or process. It is composed of three primary components: the physical entity or asset, the virtual counterpart, and the connection that enables real-time communication between the two. The physical entity captures data through sensors, which are then transmitted to the virtual counterpart, where sophisticated algorithms analyze and model the data. This integration allows for the exchange of information and enables decision-making based on real-time insights [Sin21].

SNs play a pivotal role in the implementation of DT technology, providing a vital link between the physical asset and its virtual counterpart. These networks consist of a multitude of sensors strategically deployed to capture real-time data from the physical environment. Whether utilizing wired or wireless connections, SNs enable the continuous acquisition of valuable information about the asset's condition, performance, and operating parameters. This data serves as the foundation for creating an accurate digital representation of the asset within the DT framework [Fla09] [Sha14].

Wired SNs offer reliable and robust communication channels, ensuring high-quality data transmission with minimal latency. They are particularly suitable for critical applications where data integrity and security are of utmost importance. Wired connections provide a stable and dedicated link, minimizing the risk of signal interference and offering precise data synchronization. These networks often rely on standardized protocols such as Ethernet, ensuring compatibility and ease of integration with existing infrastructure [Fla09].

On the other hand, WSNs offer unparalleled flexibility and scalability, making them well-suited for dynamic and distributed environments. They eliminate the need for extensive wiring, allowing for easy installation and deployment of sensors in remote or challenging locations. Wireless connections enable real-time data transmission over long distances, facilitating the monitoring of assets spread across vast areas. Additionally, wireless networks offer the advantage of mobility, allowing sensors to be easily repositioned or added as per changing operational requirements [Sha14].

Regardless of the chosen network type, SN are essential for collecting a diverse range of data, including temperature, pressure, vibration, humidity, and more. These data streams provide valuable insights into the asset's performance, behavior, and environmental conditions. The integration of SN with DTs empowers organizations to monitor asset health, identify anomalies, and optimize operations in real-time. The continuous feedback loop established through SN ensures that the virtual model remains synchronized and accurate, enabling effective decision-making and predictive analysis.

Basically, SN, whether wired or wireless, are a fundamental component of DT technology. They serve as the bridge between the physical asset and its digital counterpart, enabling the collection of real-time data for accurate modeling, simulation, and analysis. The choice of network type depends on the specific requirements of the application, balancing factors such as reliability, scalability, flexibility, and cost-effectiveness. With SN as the backbone, DTs unlock the potential for improved operational efficiency, predictive maintenance, and data-driven decision-making in various industries.

DT technology comprises a combination of advanced technologies, including IoT, AI, ML, cloud computing, and data analytics. The architecture typically involves SN, data acquisition systems, communication protocols, data storage infrastructure, and visualization interfaces. The integration of these technologies facilitates data collection, analysis, and feedback between the physical asset and its digital representation [Rat21].

DT has found applications across various industries, including manufacturing, energy [Pet20], healthcare [Ero20], transportation [Gao21], water distribution [Fue20] and smart cities [Iva20]. In manufacturing, DTs enable real-time monitoring and optimization of production processes, predictive maintenance, and quality control [QiQ18]. In the energy sector, DTs facilitate efficient management of power grids, renewable energy systems, and predictive maintenance of equipment. In healthcare, they aid in patient monitoring, personalized medicine, and simulation of surgical procedures. Transportation industries utilize DTs for predictive maintenance of vehicles, traffic optimization, and autonomous vehicle testing. Smart cities leverage DTs to optimize urban planning, energy consumption, and resource management.

DT technology offers several significant benefits. Firstly, it allows for enhanced operational efficiency through real-time monitoring and optimization of systems. By simulating and analyzing the behavior of physical assets, organizations can identify bottlenecks, reduce downtime, and improve overall productivity. Secondly, DTs enable predictive maintenance, allowing for proactive identification of equipment failures and minimizing unplanned downtime. Thirdly, DTs facilitate improved decision-making by providing data-driven insights and the ability to simulate different scenarios. This capability enhances strategic planning and risk mitigation. Finally, DTs contribute to sustainability efforts by optimizing resource consumption, reducing waste, and supporting energy-efficient [Wes18].

DT technology is poised to play a pivotal role in the advancement of 5G and beyond applications. As the next generation of wireless communication, 5G promises ultra-low latency, high bandwidth, and massive connectivity, creating a fertile ground for the integration of DTs. By leveraging the power of 5G networks, DTs can seamlessly exchange vast amounts of real-time data between physical assets and their virtual representations. This enables highly responsive and interactive simulations, enabling precise control, monitoring, and optimization of complex systems. Furthermore, the combination of 5G and DT technology opens doors to transformative applications, such as remote operation of autonomous vehicles, real-time monitoring of critical infrastructure, and immersive virtual experiences. The high-speed and reliable connectivity provided by 5G networks enhances the accuracy, scalability, and effectiveness of DTs, fueling innovation and unlocking new possibilities across industries and sectors [Ngu21].

The DT concept represents a groundbreaking approach to merge physical and virtual worlds, revolutionizing industrial simulation and optimization. By harnessing the power of advanced technologies, DTs enable real-time monitoring, predictive maintenance, and data-driven decision-making across various sectors. As this paradigm continues to evolve, it holds the potential to transform industries, drive innovation, and shape the future of how we design, operate, and optimize physical systems.

2.3 Machine Learning (ML)

In this section we will talk about ML-based methods that are considered in this PhD thesis. Specifically, in section 2.2.1 we will discuss the general concept of Deep Learning, and in section 2.2.2 we will go into the details of AE, a type of feed-forward, non-recurrent ANN, which learn by unsupervised learning, and have an inherent capability to learn a compact representation of data.

2.3.1 Deep Learning

Deep Learning is a subset of Artificial Neural Networks (ANNs) that focuses on architectures with multiple layers. It aims to emulate the hierarchical organization of the human brain, allowing computers to learn and extract intricate patterns from complex data. By leveraging deep neural networks, Deep Learning has revolutionized the field of ML and achieved remarkable breakthroughs in various domains [Goo16].

The key feature of Deep Learning is its ability to automatically learn representations from raw data. Each layer in a deep neural network learns increasingly abstract and high-level features by building upon the representations learned in the previous layers. This hierarchical learning process enables the network to capture complex relationships and extract meaningful insights from large datasets.

Training deep neural networks poses challenges due to the vanishing or exploding gradient problem. As the gradients used for weight updates are propagated backward through multiple layers, they can either become extremely small or explode, hindering the learning process. To address this issue, techniques such as the rectified linear unit (ReLU) activation function and batch normalization have been introduced, enhancing the stability and convergence of deep networks [Shr19].

Deep Learning has achieved groundbreaking results in various fields. In computer vision, deep neural networks have demonstrated superior performance in image classification, object detection, and facial recognition tasks. Natural Language Processing (NLP) benefits from deep learning architectures for tasks such as machine translation, sentiment analysis, and language generation [Ott21]. Deep Learning has also made significant contributions to healthcare, autonomous driving, finance, and many other domains.

The success of Deep Learning is attributed to the availability of massive amounts of data, advances in computing power, and algorithmic innovations. The use of graphics processing units (GPUs) and distributed computing frameworks has accelerated the training of deep neural networks on large-scale datasets.

However, Deep Learning also faces challenges, including the need for substantial computational resources, the requirement of extensive labeled data for training, and the potential for overfitting when dealing with small datasets. Researchers continue to explore techniques to address these challenges, such as transfer learning, semi-supervised learning, and generative adversarial networks [Sha19].

2.3.2 Autoencoders (AE)

AEs are a type of artificial neural network architecture that aim to learn efficient representations of input data by reconstructing the input itself. They belong to the field of unsupervised learning, as they do not require labeled data for training

[Hin06]. AE have gained significant attention in the field of deep learning due to their ability to learn meaningful representations and their numerous applications in various domains such as image recognition, AD, and data compression [Tsc18].

The basic structure of an AE consists of an encoder and a decoder. The encoder takes an input and maps it to a lower-dimensional representation called the *Encoding* or the *Latent Space* (LS). The decoder then attempts to reconstruct the original input from this latent representation. The overall goal of the AE is to minimize the difference between the input and the reconstructed output, encouraging the model to learn a compressed representation that captures the most key features of the input [Hin06].

The architecture of an AE can vary depending on the specific application and the nature of the data. However, the most common type is the "fully connected" or "dense" AE, where each neuron in one layer is connected to every neuron in the subsequent layer. The number of neurons in the input and output layers is equal, while the number of neurons in the intermediate or hidden layer is typically smaller, resulting in dimensionality reduction.

Training an AE involves two main steps: the forward pass and the backward pass. During the forward pass, the input is fed through the encoder, and the latent representation is obtained. Then, the latent representation is passed through the decoder to generate the reconstructed output. In the backward pass, the difference between the input and the output is measured using a loss function, such as mean squared error (MSE) or the mean absolute error (MAE), and the weights of the AE are adjusted using backpropagation and gradient descent to minimize this loss. By iteratively updating the weights, the AE learns to reconstruct the input as accurately as possible.

One of the key benefits of AE is their ability to capture useful representations of data. The hidden layer, or the LS, acts as a compressed and informative representation of the input. This property makes AE useful for tasks such as dimensionality reduction, where the high-dimensional input is mapped to a lower-dimensional space while preserving relevant information. By discarding the decoder part of the AE, the compressed LS can also be used for data visualization.

Another important application of AE is AD. Since AEs learn to reconstruct normal data patterns, they can be used to identify anomalies or outliers in new data [Yoa10]. By comparing the reconstruction error between the input and the output, anomalies can be detected as instances where the reconstruction error exceeds a certain threshold [Vel19].

AE can also be used for generative purposes. By training an AE on a specific dataset, the decoder can be used to generate new data samples that resemble the training data. Variations of AE, such as variational AE (VAEs) [Kus17] and generative

adversarial networks (GANs) [Cre18], have been developed to enhance the generative capabilities of the model.

In summary, AEs are a type of neural network architecture that learns efficient representations of input data by reconstructing the input itself. They are widely used for dimensionality reduction, AD, data compression, and data generation tasks. With their ability to capture meaningful representations, AEs have become an essential tool in the field of deep learning.

2.4 Knowledge Transfer (KT)

Knowledge transfer plays a crucial role in the advancement of AI and ML systems. It involves the process of extracting, abstracting, and transferring knowledge from a source domain to a target domain, enabling the target domain to benefit from the expertise of the source domain. ML algorithms heavily rely on data and patterns to make accurate predictions or decisions. However, in many real-world scenarios, labeled data may be scarce or expensive to obtain. Knowledge transfer offers a solution by leveraging knowledge acquired from related or similar domains to improve the learning performance in the target domain [Pan10].

Knowledge transfer enables ML models to generalize better, adapt faster, and achieve higher performance with limited labeled data. It helps to overcome the data sparsity problem and reduce the need for extensive training in the target domain. By transferring knowledge, ML systems can benefit from pre-existing knowledge, models, or feature representations, saving time and resources. Despite its potential benefits, knowledge transfer in ML faces several challenges. These include the identification of relevant source domains, the alignment of feature spaces, the mitigation of negative transfer effects, and the trade-off between transferring too much or too little knowledge. Domain dissimilarity, distribution shift, and the presence of noisy or irrelevant information are additional hurdles to be addressed [Zhu20].

Various transfer learning approaches have been developed to facilitate knowledge transfer in ML. These include: *i)* Pre-training and Fine-tuning: Pre-training a model on a large-scale dataset from a source domain and fine-tuning it on a smaller dataset from the target domain. *ii)* Domain Adaptation: Modifying the ML model to reduce the discrepancy between the source and target domains, aligning their feature distributions. *iii)* Instance Transfer: Transferring instances or examples from the source domain to the target domain, either directly or by generating synthetic instances. *iv)* Multi-Task Learning: Training a model to perform multiple tasks simultaneously, where the knowledge gained from one task can improve performance on another [Wei20].

Assessing the effectiveness of knowledge transfer methods requires appropriate evaluation metrics. Commonly used metrics include accuracy, precision, recall, F1-score, and area under the curve (AUC). Comparative evaluation against baseline models or other transfer learning techniques is essential to validate the improvements achieved [Sha18].

Knowledge transfer using ML is a valuable technique for improving performance in target domains with limited labeled data. It enables ML models to leverage knowledge from related domains, significantly reducing the need for extensive training. By addressing challenges and utilizing various transfer learning approaches, researchers can unlock the potential of knowledge transfer and advance the field of AI and ML.

Chapter 3

Review of the State-of-the-Art

This chapter aims to review the state-of-the-art related with the technologies investigated in this PhD thesis. Section 3.1 reviews the state-of-the-art regarding the sensor data rate reduction. In Section 3.2, the state-of-the-art about AD is detailed. Section 3.3 focuses on the state-of-the-art of KT. Finally, Section 3.4 concludes the revision of the state-of-the-art

3.1 Time Series Data Rate Reduction

In this section, we will present a state of the art about the data rate reduction for time series data, and it will be divided into three sub-sections. The first sub-section is about data aggregation methods, where algorithms are used to combine similar values together to reduce the size of the sent data. The second sub-section is about sampling rate adaptation methods, which reduce the amount of transmitted data by increasing the time between collected samples. The third is about compression techniques.

3.1.1 Data Aggregation

In [Che19], the authors propose a layered adaptive compression design for efficient data collection (LACD-EDC) in industrial WSN. LACD-ED on the clustering data scheme and it aims to search the spatio-temporal correlation within (e.g., intra) and among (e.g., inter) clusters. Then, a compression method is proposed at the sensor level followed by a recover technique at the sink to regenerate the raw data and achieve an approximation of original data. The authors of [Jan19] propose a sequence statistical code-based data compression method to reduce the packet size and

improve the energy efficiency of sensors. The compression process is achieved using first order static code (FOST) and sequence code (SDC) algorithms which give better compression ratio compared to arithmetic coding. In [Lia14], the authors propose a Sequential Lossless Entropy Compression (S-LEC) which organizes the alphabet of integer residues obtained from differential predictor into increased size groups. S-LEC code-word consists of two parts: the entropy code specifying the group and the binary code representing the index in the group.

3.1.2 Sampling Rate Adaptation

The authors of [Har17] propose three mechanisms that allow sensor to adapt its sampling rate to the variation of monitored environment. The proposed mechanisms are respectively based on similarity functions, distance functions, and analysis of variance with statistical tests. The proposed techniques work in rounds, where each round consists of a set period. The sensor adapts its sampling frequency at the end of each round. In [Yan16], the adaptation of sampling rate of the sensor node is based on system-context and application context levels. On one hand, the availability of harvesting energy represents the system-context to identify the maximum rate of sampling to be assigned to the sensor node. On the other hand, the user request represents the application context where feedback from a system executing specific rules of user or field scientists is used to set the rates of sensor node sampling in an optimal way. The authors of [Baş19] propose two sampling rate adaptation techniques: exponential double smoothing adaptive sampling (EDSAS) and Wiener filter based adaptive sampling (WFAS). Both algorithms search the correlation between current and previous collected data and aim to minimize the sensor sampling rate while a high level of data accuracy. In [Bah14], the authors propose a prefix frequency filtering (PFF) technique based on clustering architecture of the network. Further to local processing at the sensor node level, PFF uses Jaccard similarity function to allow aggregator nodes to identify similarities between near sensor nodes at each period and integrate their sensed data into one record.

3.1.3 Learning Compression

Authors in [Eic14]-[Hol17] tackled the problem of lossy compression in time series data using different techniques. In [Eic14] a piecewise regression technique is used to compress time series data from the smart grid. The approach depends on three regression algorithms, each specializing in a class of polynomial functions, which are applied incrementally. The final compression factor depends on the user defined maximum tolerable deviation between the original time series and the reconstructed one. The authors of [Fin11] proposed a method of lossy compression that depends on extrema (minimum and maximum) extracted from the data. Different definitions and different importance levels for extrema are applied in several pass algorithms.

The authors in [Hol17] performed an evaluation of five data compression algorithms and five change detection algorithms on several datasets. Their approach focused on finding out how these different techniques perform under different datasets with different characteristics, and how best to choose the parameters under which these algorithms will properly work.

Regarding time series compression using AEs, the authors in [Cha19] developed an algorithm called LFZip (*Lossy Floating-point Zip*), which compresses time series by using an encoder and a decoder that is based on the prediction-quantization-entropy coder framework, with works under the mean absolute error metric that has a maximum allowable error that is defined by the user. Another variant of the AE, the Convolutional AE, was used by the authors in [Aza18] to compress and decompress electroencephalogram signals to reduce the data size, thus conserving energy of the edge devices reading and transmitting these signals. Another medical application used AEs to compress data collected from wearable IoT data that have a limited energy source [Sun20], using three parameters: compression ratio, reconstruction error, and energy consumption to optimize the learning process. Also, for IoT applications with limited processing memory, the authors in [Bla18] developed a low memory, low latency algorithm for time series compression that allows to decompress later at speeds up to 3GB/s, by using a high-speed forecasting algorithm. A Recurrent Neural Network Based (RNN) AE was used in [Hsu17], combined with data segmentation and aggregation into segments of variable length but with similar total variation. Similarly, RNN AEs were used in [Won18] to partially reconstruct multi-dimensional time series data effectively, allowing insight into the operating state of some of the sensors in the system without the need for full reconstruction.

3.2 Time Series Anomaly Detection

A remarkable list of use cases and algorithms for AD in time series can be found in literature [Red16] [Coo20]. In [Red16], the authors used deep AEs trained with raw time series data from flight sensors collected under nominal operating conditions and examined the reconstruction error to detect faults with up to 97% accuracy and identify two types of faults with no false positives. Similarly, deep AEs inspired by robust principal component analysis were developed to detect outliers and perform de-noising even without access to clean data [Zha17]. The method proposed in [Guo18], called GGM-VAE, uses Gated Recurrent Unit (GRU) and is used to discover the correlation in multi-dimensional time series data. Another approach which tackles AD in multi-variate time series is the method described in [Rus20], which describes the usage of 1D convolutional neural networks, where the convolutions are performed over the inputs across the temporal axis of the data, to detect anomalies in sewer processing monitoring data, by checking if the reconstruction error in the decoding stage is above a certain value.

Also, in the scope of AD, the authors in [LiL21] used a technique they called smoothness inducing sequential variational AEs (SISVAE), which is based in Variational AEs (VAE) but has a backbone in RNNs. Their method uses the mean and variance of each sample as parameters, which means the compression process is not rigid and is flexible to the variations in the data. Moreover, to compensate for the susceptibility to anomalies that this approach generates, a smoothness inducing prior over estimations is used, thus penalizing non smooth estimations. The authors in [Suh16] used the Echo-State Network, which is a method used to train RNN where only parameters for output are learned to train VAEs to detect anomalies in multivariate time series, making use of the temporal dependence in the data. A hybrid approach for AD was used in [Ghr20], where Long-Short Term Memory (LSTM)-based AEs trained on normal samples are used to extract features from both normal samples and ones containing anomalies where an SVM classifier is used for detection purposes. A squeezed Convolutional VAE (SCVAE) is modeled to detect anomalies in edge devices of IoT as described in [Kim18], and reconstruction probability, which is a probabilistic measure that takes into account the variability of the distribution of variables, was used to tune VAEs to detect anomalies in [AnJ15]. Finally, the authors in [Coo20] conducted a survey of the AD methods for time series across a variety of domains and concluded that the main challenges remain real time processing, online adaptive learning, multivariate data, the shortage of labels anomaly data and the difficulty in obtaining it, and the lack of a generalized approach which works in all cases.

3.3 Privacy Preserving Knowledge Transfer

There is scarce literature on KT where the privacy of the sharing party is taken into consideration. The authors of [You23] proposed a method of sharing natural language processing (NLP) datasets with disadvantaged parties that do not have the ability to produce high quality dataset with reduced privacy risks. They did this through partial sharing of the parameters of and NLP model already trained on the dataset that they intend to share with the aid of a proxy dataset.

Two other works focusing on the sharing of smart meter data while ensuring the privacy of the users are [Ton16] and [Tra22]. The authors of [Ton16] proposed a low overhead method to obfuscate meter data based on 802.11s wireless mesh network that is able to protect the privacy of user data while still enabling the utility company to perform distribution state estimation. On the other hand, the authors of [Tra22] addressed the problem of the lower quality of utility estimation performed on smart meter data that have been partly obfuscated by introducing perturbations to it. They proposed a two-phase approach, one of noise generation where a distributed perturbation method provides privacy for consumers while retaining the accuracy of energy services like regional load forecasting. The second phase is a private noise

distribution protocol for called nn-PND, securely distributes n noise elements generated by an energy distribution operator to n SMs in a semi-honest adversarial model.

3.4 Conclusions

As can be seen as to the best of our knowledge, we can conclude that: *i*) none of them perform all operations (data rate reduction + compression + AD) at the same time; *ii*) some of them require a lot of real time processing power at the level of the agent performing the compression or the AD; and *iii*) some of them are only deployable after extensive training using data from the targeted systems, which may delay the deployment process.

Hence, our proposed novel system outperforms the methods in the literature in the following terms: *i*) performs data rate reduction, compression and AD at the same time using the same models; *ii*) using pre-trained models requiring very little processing power in the agent; *iii*) allowing immediate deployment by training the models using general purpose data, which allows performing at acceptable levels of compression and reconstruction errors until enough data is collected to make the models system-specific.

Regarding KT, and to the best of our knowledge, there are no literature about sharing information between DTs managing physical systems operating under a shared TN, and the literature found is about obfuscating datasets for privacy ensuring sharing or introducing noise to consumer data within a single domain to protect them from eavesdropping.

Chapter 4

Preliminaries

4.1 Reference Use Case

The reference scenario is sketched in Figure 1 (a), where a physical system (in the example, a water distribution infrastructure) contains a plethora of different sensors that generate heterogeneous telemetry data that need to be gathered and analyzed for several purposes such as smart autonomous operation. Without loss of generality, let us assume that sensors generate data periodically, with a fixed time interval (that can be different among sensors). Therefore, each single sensor is a source of one or several time series telemetry data streams. All these data flows need to be transported from their sources to the centralized location where the DT is running. A typical DT architecture consists of three essential components: *i*) a Data Lake, where the collected, pre-processed and post-processed data is stored; ii) the Sandbox Domain, containing the different models and algorithms that emulate the different components of the physical system; and iii) the Digital Twin Manager (DTM) that oversees several actions including the management of the models in the sandbox domain. Moreover, the DTM interfaces the Application Manager in charge both the physical and DT systems. Note that the Application Manager use the DT to analyze the current and future state of the physical system, which can be done by combining the collected data available in the Data Lake and the models and algorithms in the sandbox domain. The result of such analysis can lead to specific actions to be executed in the physical system. Moreover, the Application Manager can configure rules and policies to the DTM, so that the latter can perform tasks such as intelligent data aggregation and anomaly detection in an autonomous way.

Figure 1 (b) provides a deeper insight of the hierarchical architecture needed to run the proposed telemetry data compression and analysis. The first level is at the sensors layer where data is generated periodically. For the sake of simplicity, let us

assume that sensors are those physical elements that can monitor one specific metric, e.g., temperature, pressure, etc. Then, a number of these sensors are integrated in a monitoring *device*, which provides the support (computing, power) to those sensors, as well as contains the needed transceivers and interfaces (wired or wireless) required to eject the data out of the device. Since most multi-purpose monitoring devices are built on top of powerful boards such as Arduino or Raspberry Pi [Fer14] [Vuj14] a software-based *Device Agent* (DA) is deployed in the device for several purposes including telemetry data processing and device control and management. Specifically, in the context of our work, we consider that the DA contains the AEs necessary to compress the collected telemetry data and perform AD. Then, the DA sends the compressed data to the DTM that is hosted in the remote location. Along with the compressed data, three types of metadata are sent: *i*) the device/sensors identification data, including location; *ii*) the compression method metadata including aspects such as the AE id that is required to decompress the data, as well as the expected reconstruction error; and *iii*) the AD diagnosis, in case that some anomaly affecting one or multiple sensors is detected.

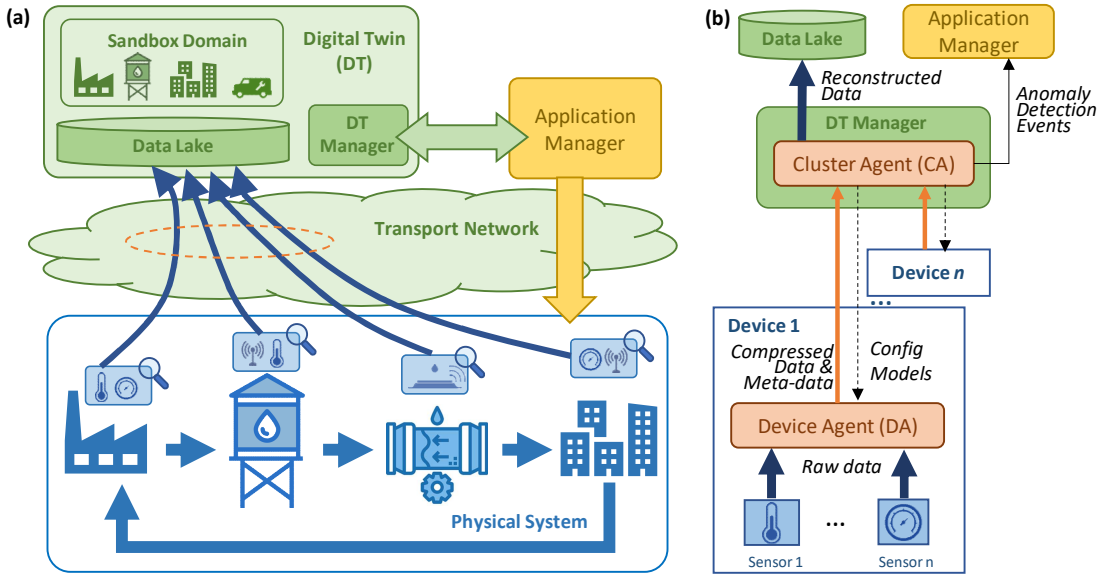


Figure 1: Reference scenario (a); overall architecture (b)

The second element in the proposed hierarchical architecture is the (CA) that runs as one of the processes in DTM, and aggregates the inputs received from several devices that form a group (cluster). The meaning of a cluster is open: can represent any subset of monitoring devices in a physical subsystem. Without loss of generality, we assume that the creation of clusters is part of the design of both the physical system and DT, which is out of the scope of this paper. Each CA oversees decompressing the data received from its nested DAs and store such decompressed data in the Data Lake. Moreover, it is also in charge of training AEs as soon as new relevant data is collected and uploading new models to the DAs in an automatized

manner. Finally, it processes the AD diagnosis reports received from DAs, perform multiple AD if needed, and notifies the application manager in case of some anomaly event has been detected.

4.2 Sensor Network Architecture

Network architecture is one of the most important aspects of the deployment of SN. It strongly affects the performance of any proposed technique. In this thesis, we interested in the cluster-based network architecture due to three main reasons: 1) It supports high network scalability regardless of the number of deployed sensor nodes. 2) It reduces the overhead communication among nodes in the network. 3) It facilitates handling node failure. Figure 2 presents the cluster scheme that is considered the main architecture of the systems that this thesis covers.

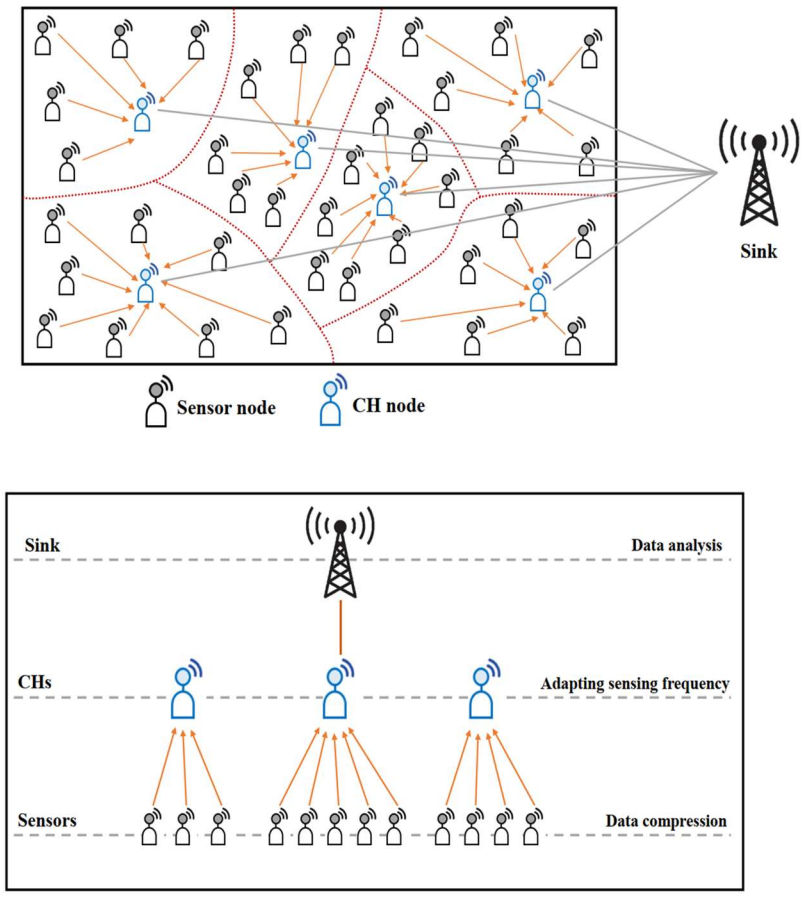


Figure 2: Network Design Based on Cluster Theme

Typically, the cluster scheme divides the nodes into a set of clusters and a CA is assigned to each. The CA receives data from the member nodes inside the cluster, performs a certain function with it if needed, then forwards it to the sink node. The CA can be the same type of mote as the nodes, or it can be a more advanced type, better equipped to perform the extra functions needed from it. Another approach would be the dynamic selection of the CH, in which case it would be subject to several metrics, for example in the case of a WSN the remaining energy of the mote, or the distance to the sink, etc.

4.3 Datasets

Two readily available datasets and were used to run simulations and experiments.

4.3.1 Intel Berkley Lab Data

Contains data collected from 54 sensors deployed in the Intel Berkeley Research lab between February 28th and April 5th, 2004 (Figure 3). Mica2Dot sensors with weather boards collected time-stamped topology information, along with humidity, temperature, light and voltage values once every 31 seconds. Data was collected using the TinyDB in-network query processing system, built on the TinyOS platform [Bod04]

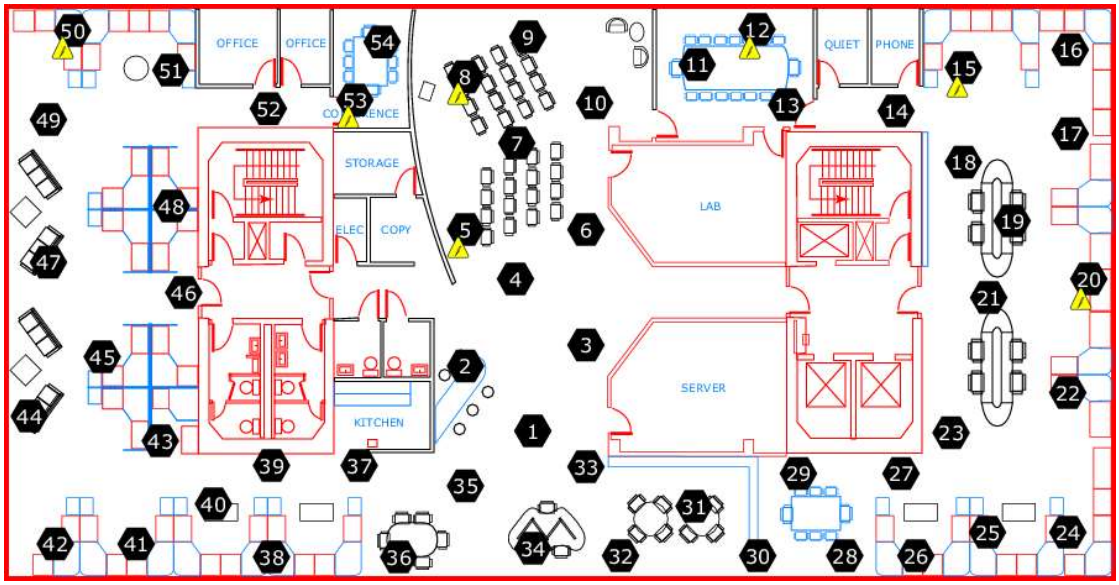


Figure 3: Distribution map of sensors in the Intel Lab

4.3.2 WADI Testbed Data

The WADI dataset contains experimental sensor data measured in a water distribution testbed under different conditions, including normal operation and operation in the presence of system perturbations (see example in Figure 4). The testbed comprises several water tanks as well as chemical dosing systems, booster pumps, valves, instrumentation, and analyzers, thus forming a complete and appropriate physical system for the performance evaluation of the proposed methods [iTr19].

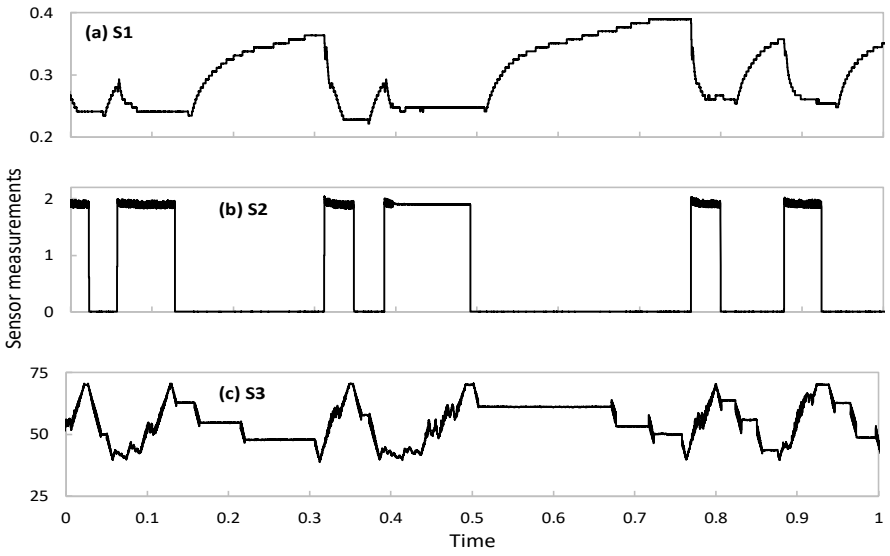


Figure 4: Example of sensor data time series in the WADI dataset

4.3.3 Datasets Summary

Table 1 contains a summary of information about the datasets we talked about earlier.

Table 1: Dataset Details Summary

Dataset	Intel Berkley Lab	WADI Testbed Data
Collected by us	No	No
Number of Sensors	54	83
Samples Per Sensor	36000	768000
Period Between Samples	31 seconds	10 second
Data Type	Light, Humidity, Temperature	Voltage, Pressure, Temperature…

4.4 General Notation

Table 2 contains the general notation used throughout the thesis.

Table 2: General Notation

S	Set of sensors
G	Set of groups. A group comprises a set of sensors that can be the sensors in each device or the sensors of the same type in a given cluster
$S_g \subset S$	Subset of sensors belonging to group g⊂ G
w	Monitoring interval duration, in time units
x_{st}	Raw telemetry data vector from sensor s at time interval t
y_{st}	Compressed telemetry data vector from sensor s at time interval t
x'_{st}	Reconstructed telemetry data vector from sensor s at time interval t

4.5 Performance Evaluation Metrics

Table 3 describes the main performance evaluation metrics that will be used throughout the document.

Table 3: Performance Evaluation Metrics

Metric	Definition	Example
Compression Ratio	The ratio of the compressed data size of the original data size	Uncompressed data: 10 Compressed Data 2 Ratio = 2 / 10 = 0.2
Compression Factor	1 / Compression Ratio	Uncompressed data: 10 Compressed Data 1 Factor = 1 / 0.2 = 5
Energy Consumption Ratio Percentage	The ratio of the energy consumption due to the transmission of compressed data over the energy consumption of uncompressed raw data	Uncompressed data energy consumption : 10 Compressed data energy consumption : 3 Ratio Percentage = 3 * 100 / 10 = 30%

Chapter 5

Statistically Based Compression and Sampling Rate Adaptation

In this chapter, we focus on automatic SN management and propose an energy-efficient Zoom-In Zoom-Out (ZIZO) mechanism based on the cluster network architecture and dedicated to periodic WSN applications. It utilizes two techniques which are applied at sensor level and the CA respectively. The first, called Index-Bit-Encoding (IBE), is a data compression method that exploits the similarity in readings collected by the sensor to reduce the size of data that it transmits. The second, called Sampling Rate Adjustment (SRA), studies correlation among sensor node data then adapts the sampling frequencies (SR) of the sensors in the cluster. Figure 5 sketches the main blocks of ZIZO within the architecture in Figure 1.

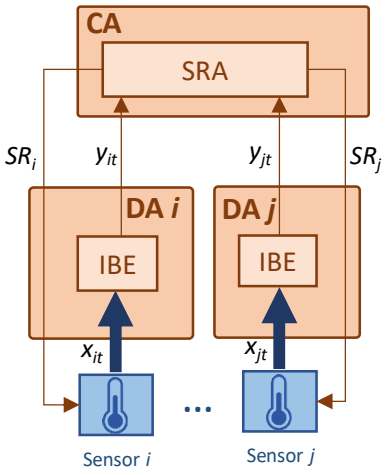


Figure 5: Diagram of ZIZO in the proposed architecture

The rest of the chapter is organized as follows: Section 5.1 describes the operation of ZIZO at sensor level i.e., the IBE method. Section 5.2 describes the operation of ZIZO at CA level i.e., the SRA method. The results obtained are evaluated in section 5.3. Finally, section 5.4 concludes the chapter.

Table 4 summarizes the special notation to be used in this chapter along with the general notation introduced earlier.

5.1 ZIZO at Sensor Level

In periodic SN, the similarity in collected readings is inversely correlated to the rate of variation in the conditions of the monitored target. i.e., The bigger the rate of variation in the conditions is, the lower the level of similarity, and vice versa. In other words when the conditions are stable, the rate of variation would be very small and the level of similarity will be high, which also means there will be a high level of redundancy in the data. ZIZO aims at reducing the size of raw data sent by using IBE, which allows to aggregate similar readings in each period without compromising the temporal information of the collected data.

Table 4: Special Notation

ε_{sim}	Similarity Threshold
u_{st}	Set of unique values extracted from a set of readings x_{st} in an interval w
c	Unique value code
$wgt(c)$	Weight of the code c
ε_{prob}	False-rejection probability for the T-test
RR_w	Redundancy Rate for a period w
C_r	Application Criticality

5.1.1 Definitions

Let us now define the similarity function (Δ) that each sensor uses to search for similarity between the readings it collected during a period.

Definition 1: (Similarity function, $\Delta\ x_i,\ x_j$). Assume x_i and x_j are two readings collecting by a sensor node S. We consider x_i and x_j similar if and only if the difference between them is less than a defined threshold ε_{sim} as follows:

$$\Delta\ x_i,\ x_j = \left|x_i\ -\ x_j\right| \leq \varepsilon_{sim}$$

where ε_{sim} is a threshold determined by the application parameters. By applying the similarity function to the readings of x_{st} collected at the period w sequentially, we get a set of unique readings u_{st} where each reading in x_{st} is similar to one and only one of the values in u_{st}. Then for each reading in u_{st} its weight is calculated using IBE which is explained in the next section. The combination of unique values and their weights form the compressed set of readings y_{st} which is sent to the CH.

5.1.2 Index-Bit-Encoding (IBE) Method

IBE method enables the compression of the set of readings x_{st} while preserving the temporal information of the data. It does this by encoding the indices of the readings in u_{st} in the weight of the corresponding reading. This way, upon decompression, a very close estimate of the original set of readings is generated, with the mean difference between the original set and the decompressed set is less than ε_{sim}. The generation of the weight passes through the intermediate process of generating the code of reading defined in the following:

Definition 2: (Code of a unique value $u \in u_{st}$, c). c is a binary code with a size equal to the number of readings in x_{st}. Thus, for a reading u in u_{st} , for each x in x_{st} , if u and x are similar corresponding bit in c is set to 1, otherwise it set to 0.

Definition 3: (Code weight, *wgt(c)*). The code weight *wgt(c)* of a code c is the integer value corresponding to it. The final form of the set of readings x_{st} would be: y_{st}= [(u_1, *wgt* (c_1)), (u_2, *wgt* (c_2)), . . ., (u_n, *wgt* (c_n))].

The size of the set of readings would be optimally set to multiples of 8, as this would correspond to one byte. Given that each decimal value in the set of readings needs 4 bytes to be encoded and depending on the size of the set of readings x_{st} , which is subject to adapt under ZIZO, the size of the compressed set of readings could be much smaller than the original set.

Figure 6 shows an illustrative example of the compression and decompression of the set of measurements through IBE using similarity function with threshold ε_{sim} = 0.8.

5.1.3 Sensor Level Algorithm

Algorithm 1 shows the sequential application of the similarity function on the set of readings of a sensor in period x_{st} to generate the compressed vector y_{st}, with a period size w and similarity threshold ε_{sim}.

x_{st}	[	10.0,	12.1,	12.2,	10.6,	11.4,	14.4,	13.9,	9.2]
▼						***Similarity Function***			
ε_{sim} =0.8	[	10.0,	12.1,	12.2,	10.6,	11.4,	14.4,	13.9,	9.2]
▼						***Code of Reading***			
8-bit	[	(10.0, 10010001)				(12.1, 01101000)		(14.4, 00000110)	]
▼						***Code Weight***			
y_{st}	[	(10.0, 145)				(12.1, 104)		(14.4, 6)	]
						Decompression			
x'_{st}	[	10.0,	12.1,	12.1,	10.0,	11.4,	14.4,	14.4,	10.0]

Figure 6: Illustrative example of data compression and decompression in ZIZO

Algorithm 1: ZIZO sensor level algorithm

```
    Input: period size: w; similarity threshold: ε_sim; set of
    readings of a sensor in period: x_st
    Output: compressed readings set of sensors in period: y_st.
1.  u_st ← ∅,
2.  for each value x_i ∈ x_st where i ∈ [1, w] do
3.      SimIndex ← indexOf (u ∈ u_st where Δ x_i,u ≤ ε_sim)
4.      if SimIndex == -1 then
5.          c = {0,0, 0, …, 0,}
6.          u_st ← u_st ∪{x_i, c}
7.          SimIndex ← i
8.      end if
9.      for each reading u_j ∈ u_st where j ∈ [1, Count (u_st)] do
10.         if SimIndex == j then
11.             c_j ← c_j ∪{1}
12.         end if
13.     end for
14. end for
15. y_st ←∅
16. for reading u_j ∈ u_st where j ∈ [1, Count (u_st)] do
17.     wgt(c_j) ←CalculateCodeWeight(c_j)
18.     y_st ← y_st ∪{u_j, wgt(c_j)}
19. end for
20. return y_st
```

The algorithm can be executed in two ways. Either the x_{st} is collected and at the end of the period the sensor proceeds to compress the set of readings, like what is demonstrated in Algorithm 1. The algorithm starts at index 1 and proceeds until the end of the period i.e., when w readings are obtained. Any possible sequence can be followed to obtain the best compression, as the low complexity of the algorithm permits this. The other way is to start the compression at the beginning of the period, with each new reading taken being immediately added to the set of compressed

values y_{st} appropriately by applying the similarity function and adjusting the code of an existing reading or adding a new one. This is even less complex and easier to implement. The set of unique readings u_{st} is started empty (line 1). Then for each reading x_i taken from x_{st} a check is done to see if there a reading in u_{st} similar to it and get the index of this values, or -1 in case none is found (line 3). If a similar value is not found, x_i is added to u_{st} with a code that indicates that there are no readings similar to it in the previous indices (line $4-7$). Then the code c of the unique value in u_{st} that is similar to x_i is changed to 1 and the rest of the unique values remain 0 (lines $9-13$). Finally, for each unique value in u_{st} the code weight *wgt(c)* is calculated from c, and they are added to the compressed set y_{st} (lines (14–19)).

5.2 ZIZO at CA Level

Each period w is divided into T equal slots of time, e.g., $w = [s_1, s_2, \ldots, s_T]$. At the end of each period, the CA will receive the sets of readings coming from its member nodes in the cluster. Indeed, data collected by the sensors in the same cluster contain a certain level of redundancy due to the spatial-temporal correlation between them. Thus, in order to reduce the data transmission from the member nodes, the CA would adapt the sensing frequencies of the sensors to minimize the redundancy. Let assume that the CA receives the compressed set of readings y_{st} from a sensor node s. The first step is to decompress y_{st}, using the process of reverse IBE, index-bit-decoding (IBD). This allows the generation the set x'_{st}, which is a very close estimation of x_{st}. Then, all the decompressed reading sets are vertically stacked to form a matrix M_w at each period w in the shape in Figure 7 where each row R^t for $t \in T$ consists of a set readings of n sensors at one instant of time t. Our objective is to study the redundancy existing in M_w.

	s_1	s_2	…	s_n
R^1	r_{11}	r_{12}	…	r_{1n}
R^2	r_{21}	r_{22}	…	r_{2n}
…	…	…	…	…
R^T	r_{1T}	r_{2T}	…	r_{nT}

Figure 7: Example of the matrix M_w

Then, based on it, to adapt the sensing frequency of all the sensors in the cluster for the next period *w+1* in order to try to minimize the collection of redundant data. In the next sections, we show how the CA calculates the redundancy level in the data matrix based on the *T-test* statistic and how it adapts the sensing frequencies of the sensors.

5.2.1 Redundancy Rate based on T-test.

In our mechanism, we incorporate the temporal information between the collected data in the calculation of the redundancy rate. The CA determines if each pair of sets of readings collected at successive slot times t_i and t_{i+1} are similar, and based on the number of such sets, it calculates the Redundancy Rate for the period RR_w. The paired *T-test* is fitting for this measurement because each pair of successive sets is a set of paired observations by statistical definition. The null hypothesis of the T-test is that the true mean difference between paired observations is zero. The alternative hypothesis being that there is a significant difference. Let's assume the set of readings collected by all the sensors during a slot time $t \in T$ as $R^t = [r_1, r_2, \ldots, r_n]$, where r_i is collected by the sensor S_i and the number of sensors, i.e. the number of readings in R^t is n. Then, the value of the *T-test* for the pair R^t and R^{t+1} collected at successive time slots t and $t + 1$ is calculated as follows:

$$T - test(R^t, R^{t+1}) = \frac{\bar{X}^{t,t+1} - \mu}{\frac{\sigma^{t,t+1}}{\sqrt{n}}}$$

where:

$\bar{X}^{t,t+1}$: the mean of the difference set $\{ R^{t+1} - R^t \}$

μ: is the population mean of the whole matrix M_w

$\sigma^{t,t+1}$: the standard deviation of the difference set $\{ R^{t+1} - R^t \}$

Thus, two successive sets of readings R^t and R^{t+1} are considered as having a significant difference if the value of the T-test between them is less than the defined threshold ε_{prob} as follows:

$$T - test(R^t, R^{t+1}) \leq \varepsilon_{prob}$$

where ε_{prob} is the *T-test* value corresponding to some desired false-rejection probability $\alpha \in [0, 1]$. Sets of readings that do not have a significant difference between them are considered similar.

Finally, the redundancy rate during the period w, i.e., RR_w, can be calculated as the number of similar successive pairs of sets of readings over the total number of possible successive pairs of sets of readings *(T −1)* during the period as follows:

$$RR_w = \frac{\sum_{i=1}^{T-1} T - test(R^t, R^{t+1}) \leq \varepsilon_{prob}}{T-1}$$

Thus, RR_wwill be a number between 0 and 1; with 0 meaning no redundancy and 1 meaning maximum redundancy. The value of a greatly influences the redundancy rate; the higher it is, the greater the tolerance is to the difference between sets. That is why the choice of a is a parameter set depending on the application.

5.2.2 Sensing Rate Adjustment Algorithm

After calculating the redundancy rate at each period, the CA adapts the sensing frequencies of the sensors in the cluster uniformly to try to minimize the collection of redundant data in the next period, within an upper and lower bound. A big level of similarity of the readings of one sensor can be offset by a big level of difference in the readings of another in the cluster, leading to a sampling rate that is neither too high nor too low. In addition to redundancy, several metrics that are determined depending on the application can be used, and they can be a predefined set of rules, or dynamic based on the changing nature of the data. The preservation of the temporal data and the fact the a very close estimate of the original set of measurement can be created for each sensor means that trend analysis and time series analysis could be used by the sink, which may have additional resources, and by using these tools a smart decision making system for sensing rate adaption can be implemented, and the sink would then direct the CHs to change the sensing frequencies in their respective clusters accordingly.

5.3 Performance Evaluation

The performance of our mechanism is evaluated using simulations and experiments. The first is done using data collected by the Intel Berkeley Research Lab and they are available online. The second is using data gathered through real experiments using telosB nodes. In the next sections, we detail and discuss the results of the simulation. The parameters used in our setup for simulations are shown in the Table 5.

For our application, we considered there are three levels of redundancy as shown in Table 6, Redundancy Rate Table (RRT). Also, for SRA, we introduced a parameter that we used along the redundancy rate called application criticality. We considered the application to be either of low or high criticality. The following Table 7 Sampling Rate Adjustment Table (SRAT) was created and used a decision mechanism by the CH.

Table 5: Simulation Application Parameters

Parameter	Description	Values
w	Period Size	32 and 64
ε_{sim}	Similarity Threshold	Temperature: 0.07, 0.1,0.2 Humidity: 0.2, 0.5, 0.1 Light: 10, 15, 25
C_r	Application Criticality	*low* and *high*
ε_{prob}	False-rejection probability risk value	0.2

Table 6: Redundancy Rate Table (RRT)

Redundancy Rate	Description
$0 \leq RR_w \leq 0.4$	Low Redundancy Rate
$0.4 \leq RR_w \leq 0.7$	Medium Redundancy Rate
$0.7 \leq RR_w \leq 1$	High Redundancy Rate

Table 7: Sensing Rate Adjustment Table (SRAT)

RR_w	C_r low	C_r high
low	60% of T	T
medium	40% of T	60% of T
high	20% of T	40% of T

5.3.1 Simulation Results

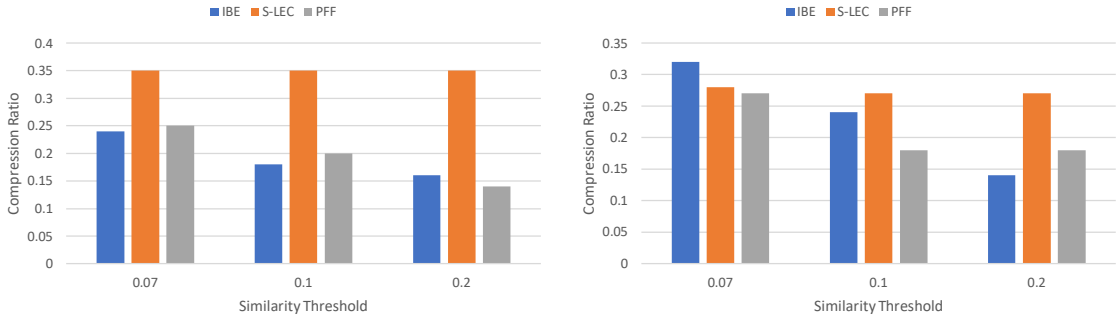
For our setup, we assumed that all nodes send their data to a common CA placed at the center of the lab. Figure 3 shows the distribution of the sensors in the Intel lab. For comparison purposes, we selected those techniques among all available in the literature that closer fit with the main characteristics of our proposed mechanism, i.e., distributed operation and energy minimization target by means of sensors data collection reduction. Specifically, we selected:

1) S-LEC in [Lia14] as it is a lossless compression technique that compresses data efficiently and robustly but pays no attention to the redundancy in the data.

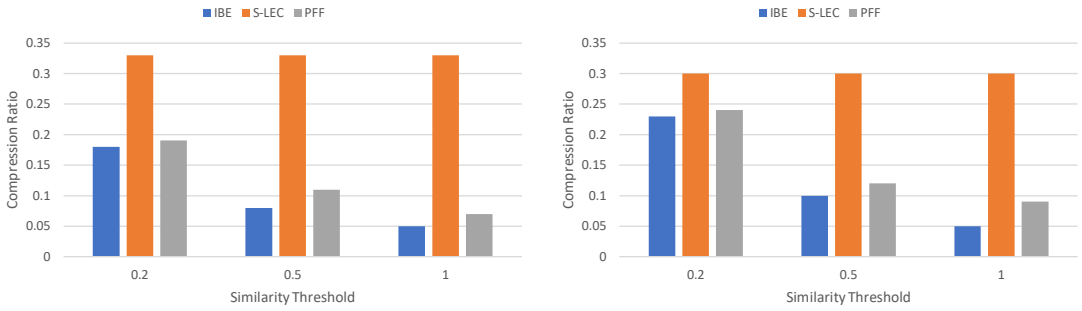
2) PFF technique in [Bah14] which is similar to our technique in terms of using aggregation on the part of the sensor to compress data and finding similarity in the data on the part of the CH. However, the aggregation method of the technique does not preserve temporal information, and only uses similarity in the data to further remove redundant data before sending the data to the sink.

1) Compression Ratio Study: First, we study the performance of the IBE method proposed in our mechanism compared to S-LEC and PFF at the sensor level in terms of data compression ratio only i.e., the size of data sent after compression relative to the size of the data vie that naive approach. Indeed, the obtained result of IBE is highly related to the period size T and the Similarity Threshold value ε_{sim} (Figure 8). We show that the compression ratio using IBE is less than those obtained using S-LEC in all cases, and less than those obtained by PFF in almost all cases, giving a compression ratio of 0.05 in the best case and 0.32 in the worst case. This is due to the simplicity in which data compression is encoded compared to the other techniques. In addition, the following can be observed: 1) The compression ratio of IBE improves with the decreasing of the period size (Figure 8 (a) and Figure 8 (b)) or (Figure 8 (c) and Figure 8 (d)) or (Figure 8 (e) and Figure 8 (f)). This is because when T decreases, the number of bytes needed to record the temporal information decreases, which is why period dividing becomes an advantage. 2) The compression ratio of IBE improves with increasing the Similarity Threshold Figure 8 (a) to Figure 8 (f)). This is because when the value of ε_{sim} increases, the greater the difference that can exist between two readings while still considering them similar and aggregating them together. This is why the Similarity Threshold is a crucial factor in determining both the Compression Ratio and Data Accuracy. 3) IBE gives much better results than S-LEC in the cases where there is high redundancy, such as with the humidity condition as shown in Figure 8 (c) and Figure 8 (d)). This is because SLEC does not aim to reduce redundancy but compresses the data as is, which allows IBE to be more effective in these cases.

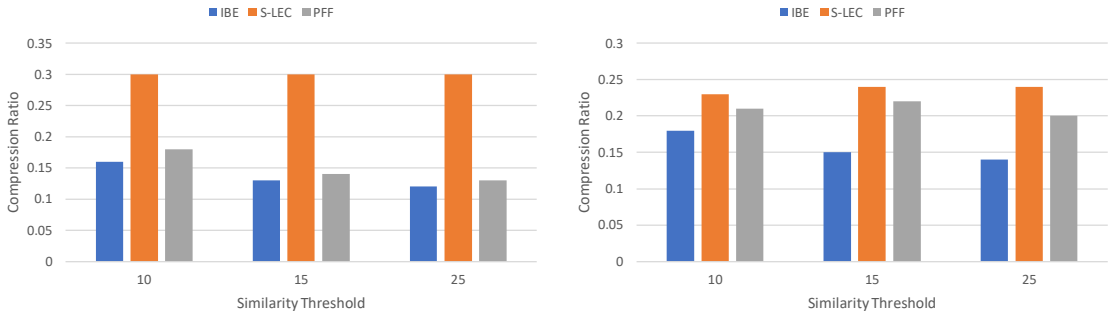
2) Data Redundancy Study: The aim of this section is to show how SRA affected the value of the redundancy rate resulted in the CA at the end of each period. For this aim we fixed three parameters: the period size to 64 readings, the similarity threshold to 0.1 and T-test false-rejection probability risk to 0.2. We also used only temperature data and ran the simulation under three conditions: Without SRA, with SRA and low criticality, with SRA and high criticality. Figure 9 demonstrates how the redundancy level (calculated in equation 5) varied over the course of 20 periods. The results show how the RR_w leads to SRA, which leads to decreasing RR_w in the next round, which is clear in the case of SRA with low criticality as compared to No SRA. They also show the effect of the SRA decision parameters, as setting strict SRA decision parameters such as for high criticality applications lead to redundancy rates that sometimes matched the condition where no SRA is used.



(a) Temperature, *T=32* (b) Temperature, *T=64*



(c) Humidity, *T=32* (d) Humidity, *T=64*



(e) Luminosity, *T=32* (f) Luminosity, *T=64*

Figure 8: Data compression ratio for as a function of similarity threshold under different conditions

Figure 9: Redundancy rate variation as a function of period number for temperature data $T=64$, $\varepsilon_{sim} = 0.1$

3) Adapting Sampling Rate Study: This section uses the same simulation used in the previous one but aims to highlight how exactly the sampling frequency was adjusted over the course of the periods under the aforementioned conditions. Figure 10 shows this variation and demonstrates how strict SRA decision parameters such as with high criticality led to the sampling frequency remaining equal to period size in much of the simulation, which caused the high levels of RR_w as shown in Figure 9.

While with low criticality the parameters allowed for a bigger decrease in sampling frequency, and lead to lower levels of RR_w. This shows the importance of choosing adequate parameter values in deciding SRA to lower redundancy without affecting data accuracy by using a very high Similarity Threshold ε_{sim}.

4) Energy Consumption Study: Reducing the energy consumption in the network is one of the most important challenges in WSN. In our simulations, we used the Heinzelman model [Hei00], [Hei02] which is the most popular model for energy consumption estimation to evaluate the energy consumption in the network under different scenarios. In such a model, energy consumption is highly dependent on the data transmission and reception in the network while negligent the other factors (sensing, processing, etc..). For each scenario, we calculated the overall energy consumption by all the sensors and the CA over the course of the whole simulation.

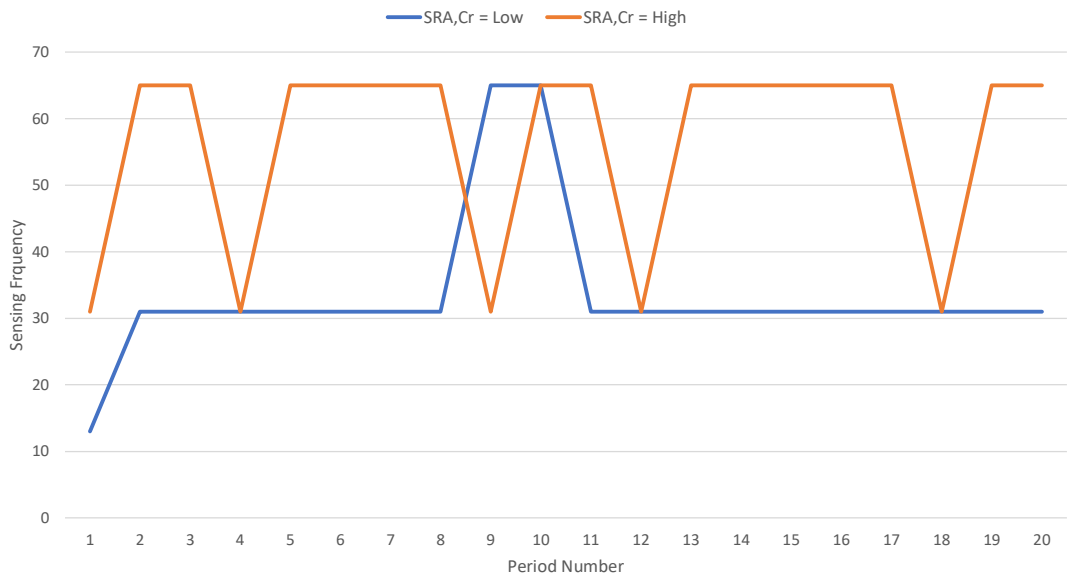
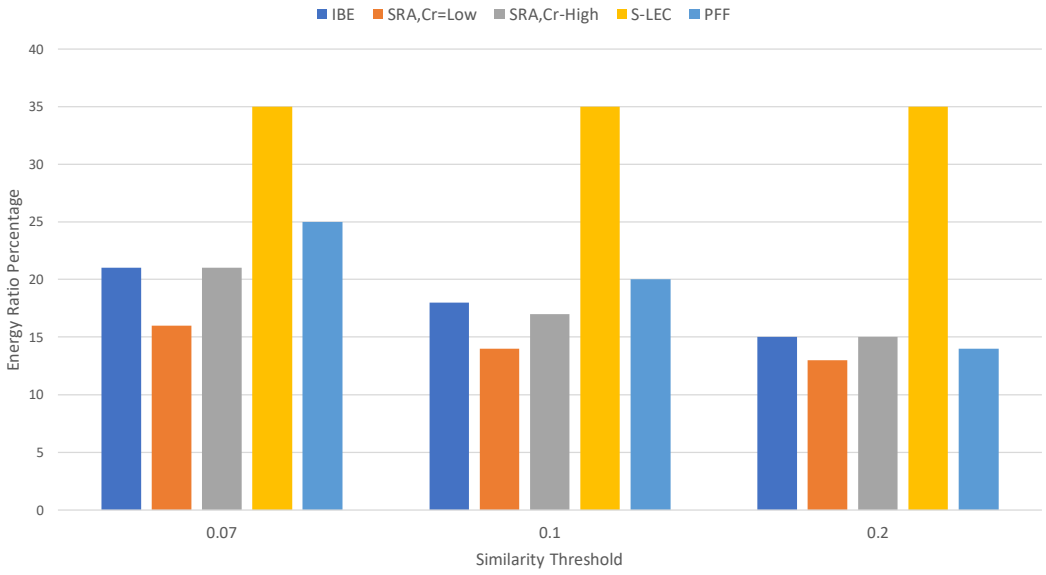


Figure 10: Variation of sensor sampling frequency as a function of period number for temperature data $T=64$, $\varepsilon_{sim} = 0.1$

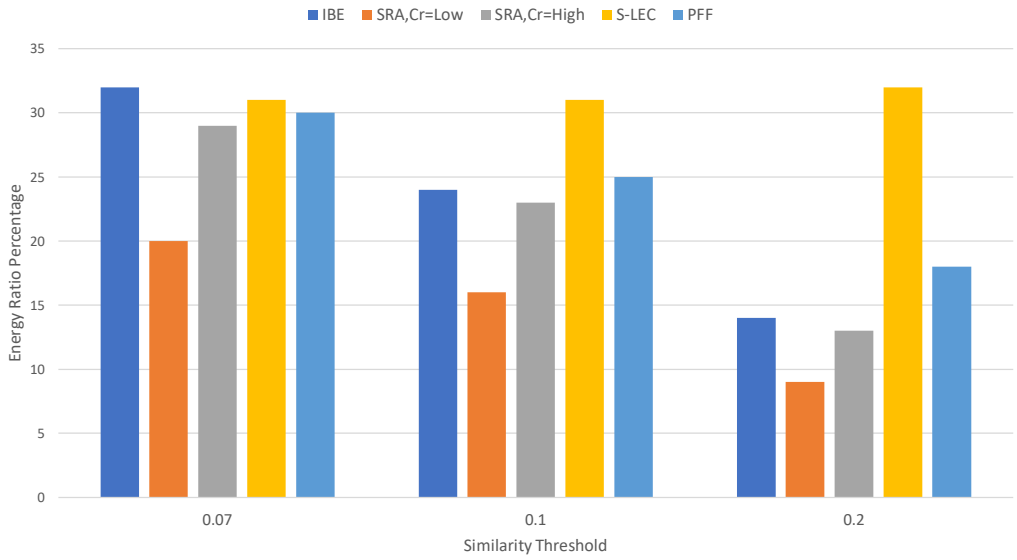
The scenarios were: using IBE alone, using IBE with SRA with low criticality, using IBE with SRA with high criticality, using S-LEC and PFF, while varying the period size and the Similarity Threshold, along the naive approach. Figure 11 displays the energy ratio of these scenarios i.e., the energy consumption in each relative to the energy consumption of the naive approach. It shows that our technique can significantly reduce energy consumption, as using IBE alone was more energy efficient in all but one case, reducing energy consumption ratio percentage down to 10% compared to 35% and 14% for S-LEC and PFF respectively. This is consistent with the results in Figure 8 which are similar in terms of compression ratio. Also, we observe that the energy consumption using our mechanism is lower with smaller period size (Figure 11 (a) and Figure 11 (b)) and higher Similarity threshold (Figure 11 (a) or Figure 11 (b)), which shows a trade-off between energy efficiency and data accuracy. Finally, we see the effect of SRA, as it led to lowering the energy consumption ratio to 8% in the best case.

5.3.2 Analytical and Complexity Study

In this section, our objective is to give further considerations of our mechanism by studying its complexity at both sensor and CA levels. Indeed, the complexity is an important metric that must be considered in WSN due to the limited sensor resources and the impact of processing to the data delivery delay when sending the data to the sink [FuX19], [FuX20].



(a) *T=32*



(b) *T=64*

Figure 11: Energy consumption ratio percentages for the whole network for some algorithms as a function of similarity threshold under different period sizes

From one hand, each sensor node S will form a set x_{st} of T readings in each period. Due to the IBE, the size of this set will be reduced from T to $|y_{st}|$. Therefore, our mechanism has at most $O(|y_{st}|^2)$ as a computation complexity at the sensor and it will save at most $(2 \times |y_{st}|)$ readings at each period in its memory. These complexities

are suitable for the case of sensor node since the collected readings are usually redundant thus, makes $|y_{st}|$ small even in the worst case. This fact is shown clearly in simulation (Figure 8) where the data collected by each sensor in each period is significantly reduced. On the other hand, the complexity of our mechanism at the CA level is dependent on the dimensions of the matrix M_w which is $T \times T$ in the worst case. Thus, after enumerating and comparing every successive reading set based on the T-test, the complexity of our mechanism will be in order of O (T^2). Such complexity can be more reduced in case we adapted the scaling down strategy proposed for large period size (cf. section IV.C) or minimized the value of T.

5.4 Conclusions

In this chapter, we have proposed a ZIZO mechanism to minimize the data transmission in SN to extend its lifetime. ZIZO is based on the cluster network architecture and works on the two levels of a WSN: a low complexity, energy efficient data compression technique called IBE at the level of the sensor, and a sampling frequency adaptation technique based on statistical similarity study called SRA at the level of the CH. Through both simulations and experiments, we evaluated the performance of our mechanism in terms of minimizing data transmission and energy consumption while comparing them to other techniques and showed that we can achieve a compression ratio of down to 0.05, and an energy consumption ratio percentage of down to 8%.

In terms of future work, ZIZO can be improved in several directions. First, we seek to improve the compression ratio of IBE. Second, at the CH, we aim to reduce the difference between the original set of readings and the decompressed set of readings by applying different filters on the latter, increasing the level of accuracy without sacrificing compression ratio. Third, we aim to use the data collected by the sensors at the sink, to implement dynamic clustering, i.e., change the distribution of sensors to the clusters, considering factors such as battery levels, and implement a scheduling strategy to switch correlated nodes into sleep/active modes.

Chapter 6

Autoencoder-based Telemetry Data Compression

This chapter describes the novel method for lossy compression of time series data using deep AEs and its possible application within the architecture of a DT. For the compression, and instead of compressing the input data using a single AE, a pool of AEs with different number of latent features is used. Thus, the *Adaptive AE-based Compression* (AAC) method is presented as an autonomous process that can choose the best AE in the pool i.e., the one that reaches a target reconstruction error with the minimum latent space (LS) size. The variability of the number of the latent features means that the size of the compressed data is not fixed, which draws similarities between AE-based compression and conventional compression methods in which the characteristics of the input data play an important role in the compression ratio. It also means that the compression is adaptive to the variations in the data and hence, compression size is indeed a variable that can be analyzed as additional and extended information of collected monitoring data. It is worth adding that AEs are trained using data from the specific sensor/s that they operate. However, since this may not be available from the beginning of sensor operation, generic AEs with moderated compression rates trained for heterogeneous sensor data are used until enough data is collected to train the specific AEs.

6.1 Main Components

Figure 12 details the architecture previously sketched in Figure 1(b), showing the key building blocks and their relationship. The figure focus on the processes related to telemetry data compression. For the sake of simplicity, the processes of training and updating AEs are not depicted in the figure. Let us assume that the DA

implements a telemetry database (DB) that temporarily stores the data injected by each of the sensors in the device. We can assume that this data collection is done at a very narrow *telemetry interval* e.g., one measurement per second and device. Then, a larger *monitoring interval* e.g., every minute, is configured to retrieve data from telemetry DB and compress them. Thus, let us denote x_{st} as the telemetry measurements collected during monitoring interval t by sensor s. This data is then fed to the *compressor* module that is responsible for running the AAC process. By means of the AE pool, adaptive and effective compression is achieved. The compressed telemetry data (denoted as y_{st}) as well as the identifier of the AE selected by AAC for compression (denoted as id_{st}) are sent to the CA. Without loss of generality, we assume that CA process the received compressed data immediately upon their reception, calling a simple *de-compressor* process that uses the decoder of selected AE to reconstruct the original telemetry stream (denoted as x'_{st}) and inject it into the data lake.

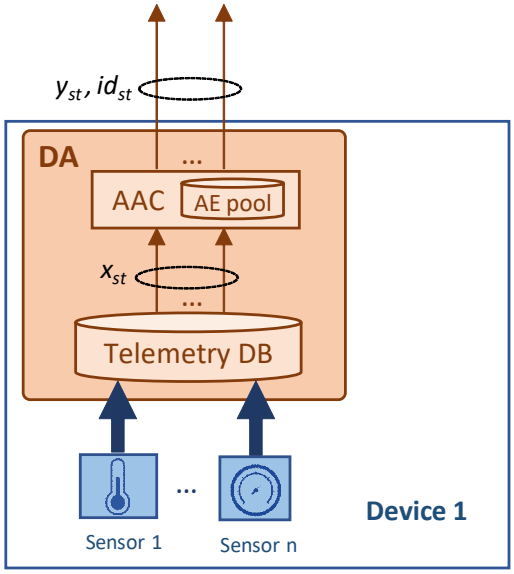


Figure 12: Components involved in AAC.

6.2 Algorithms

6.2.1 AAC Special Notation

Table 8 provides the main notation that is consistently used in the following algorithms, in addition to the general notation.

Table 8: AAC Special Notation

Z	Set of allowable sizes for the LS
Ψ_s	Pool of AEs for compressing telemetry data from sensor s
id_{st}	Id of the AE used to compress data from sensor s at time interval t
r_{st}	Reconstruction error vector from sensor s at time interval t
DB	Telemetry Database for training and testing purposes
ε_{comp}	Target average reconstruction error for compression

6.2.2 AAC

Algorithm 2 details the pseudo-code of the AAC process, which runs for each sensor s in a device and is executed every interval t a new telemetry stream is available. As introduced in the previous section, it receives the raw telemetry data stream x_{st} containing a number w of measurements, the pool of AEs of the sensor Ψ_s, and the reconstruction error threshold ε_{comp} to determine whether a given compressed stream y_{st} produces an accurate enough reconstructed telemetry stream when decoded. Besides y_{st}, the algorithm also returns the identifier of the selected AE in the pool id_{st}, as well as the reconstruction error vector r_{st}.

After initializing output variables (line 1 of Algorithm 2), the set of AEs in the pool Ψ_s are sorted in ascendant order of size of LS (line 2). Thus, they are going to be sequentially evaluated in a loop from highest to lowest compression ratio (line 3). Given an AE Ψ , the input data is normalized with the min-max values stored as model coefficients (line 4). Then, the normalized input x is propagated through the encoder part and the compressed stream y is obtained (line 5). At this point, the decoder is used to compute the reconstructed data stream x', which is used to compute the reconstruction error r that Ψ produces (lines 6-7). Note that if the average reconstruction error is below threshold ε_{comp}, then an accurate compression is found, and AE pool search is interrupted (lines 8-10). Finally, the resultant output is returned (line 12). Note that this output can be either a compressed telemetry stream if an AE producing average reconstruction error below ε_{comp} is found or the original input if no accurate compression can be effectively done.

Recall that we consider that generic AEs with moderated compression rates trained from heterogeneous generic sensor data are used until enough sensor-specific data is collected to train ad-hoc AEs that better compress the data of a given sensor. Without loss of generality, we can assume that this procedure can run periodically as soon as telemetry data from sensors is available. Algorithm 3 details the proposed

Algorithm 2: AAC method

```
INPUT: x_st, Ψ_s, ε_comp
OUTPUT: y_st, id_st, r_st
1.  y_st ← x_st; id_st ← ∅; r_st ← zeros(w)
2.  sort (Ψ_s, |Ψ .latent|, "ascending")
3.  for each Ψ ∈ Ψ_s do
4.      x ← normalize (x_st, Ψ .minmax)
5.      y ← Ψ .encoder.propagate (x)
6.      x' ← Ψ .decoder.propagate (y)
7.      r ← computeReconstructionError (x, x')
8.      if avg (r) ≤ ε_comp:
9.          y_st ← y; id_st ← Ψ .id; r_st ← r
10.         break
11. return y_st, id_st, r_st
```

Algorithm 3: AE pool update

```
INPUT: Ψ_s, Z, DB
OUTPUT: Ψ_s
1.  DB_train, DB_test ← split (DB)
2.  for each z ∈ Z:
3.      ψ_new ← trainAE (DB_train, z)
4.      ψ_cur ← select (Ψ_s, |Ψ .latent|= z)
5.      if ψ_cur = ∅ then
6.          Ψ_s.add(ψ_new)
7.      else
8.          Y ← ψ_new.encoder.propagate (DB_test)
9.          X' ← ψ_new.decoder.propagate (Y)
10.         r_new ← computeReconstructionError (DB_test, X')
11.         Y ← ψ_cur.encoder.propagate (DB_test)
12.         X' ← ψ_cur.decoder.propagate (Y)
13.         r_cur ← computeReconstructionError (DB_test, X')
14.         if avg(r_new) < avg(r_cur) and max(r_new) < max(r_cur) then
15.             Ψ_s.replace(ψ_cur, ψ_new)
16.  return Ψ_s
```

procedure to train and update the AEs in a pool. Thus, given an AE pool Ψ (that could be initially empty) and a database DB containing telemetry measurements (that can be either generic or sensor-specific), the algorithm trains a set of AEs with LS sizes defined in set Z to find new models that improve existing ones.

The procedure starts by splitting the data in DB in both training and testing datasets e.g., following a typical 80%-20% split [Bis06] (line 1 in Algorithm 3). Then, each LS size z in Z is selected and a new AE ψ_{new} is trained for such LS size (lines 2-3). This new AE needs to be compared against the current one in the pool with the same LS

size (denoted as ψ_{cur}) and therefore, it is retrieved from the pool (line 4). Note that ψ_{new} is directly added to the pool if there is no currently available AE with such size z (lines 5-6). Otherwise, the testing dataset is used to evaluate the reconstruction error in both ψ_{new} and ψ_{cur} (lines 8-13). Thus, the current AE is replaced by the new one if both average and maximum reconstruction errors are reduced by the new AE (lines 14-15). Eventually, the updated AE pool is returned.

6.3 Performance Evaluation

In this section, we first introduce the simulation environment developed to evaluate the methods and algorithm presented in previous sections. Then, we analyze the performance of AAC, using telemetry data from a real physical system. Finally, we analyze the impact of the proposed methods on a network case study where TN capacity savings are shown.

6.3.1 Simulation Environment

For numerical evaluation purposes, we implemented a Python-based simulator reproducing the main blocks of the architecture presented in Figure 17, as well as the algorithm described in the chapter. In particular, a CA with three DAs was configured, where every DA processes data from one single sensor. Sensors were implemented as time series data generators injecting real measurements (one per second) from the Water Distribution (WADI) dataset [iTr19]. The WADI dataset contains experimental sensor data measured in a water distribution system under different conditions including normal operation and operation in the presence of system perturbations. Among all available data in WADI, we selected three time series from three different sensor types (hereafter, referred as *S1*, *S2*, and *S3*) with different behaviors and patterns. Figure 4 shows an example of each sensor time series data under normal operation. As can be observed they are different in terms of time patterns, as well as in magnitude and range of the telemetry data.

For the sake of simplicity, we assume that AEs in the pool of CA and DAs are trained using Algorithm 2 after a period of raw data collection to populate the initial database *DB*. Without loss of generality, we assume that the measurements collected during this period belong to the normal operation of the physical system. Then, fixing interval w to 256 seconds, we obtained 7.68e5 samples for training, as well as 9.6e4 samples for testing. Regarding AE pool configuration, we considered 4 different AEs with $Z = \{4, 8, 16, 32\}$ LS sizes. In all the cases, we considered 2 hidden layers, with 128 and 64 hidden neurons each. We used *keras* library for AE training and testing, as well as *pandas* and *NumPy* to load and manipulate the datasets. AEs were trained

during 100 epochs using *Adam* optimizer and mean absolute errors as loss function, which results in reconstruction accuracy values around 99%.

6.3.2 AAC Results

The first numerical study is focused on evaluating the performance of AAC procedure in Algorithm 2 once the AE pool of every DA has been trained with telemetry data of its specific sensor. The metric used was the ratio of the number of windows compressed by any AE in the pool to total amount of windows (compressed to total data ratio), so 0 means that AAC cannot compress any measurement below the target ε_{comp} and 1 means that all measurements are compressed with the AE with lowest LS size (in our case, 4). Figure 13 shows the CTR as a function of target reconstruction error ε_{comp} for both normal operation Figure 13 (a) and operation with perturbations (Figure 13 (b) after 9 hours of simulated time (~32000 monitoring samples per sensor).. As can be observed, AAC shows the desired adaptability, sharply increasing compressed to total data ratio when ε_{comp} is relaxed. Interestingly, we can observe that different time series produce different compression performance, even when AEs were specifically trained for that data. However, maximum compression is always achieved with low reconstruction error (0.05) under normal system operation, as well as remains very high when perturbations appear in the system, which validates the applicability of the proposed method in systems subject to changes in the telemetry data generated.

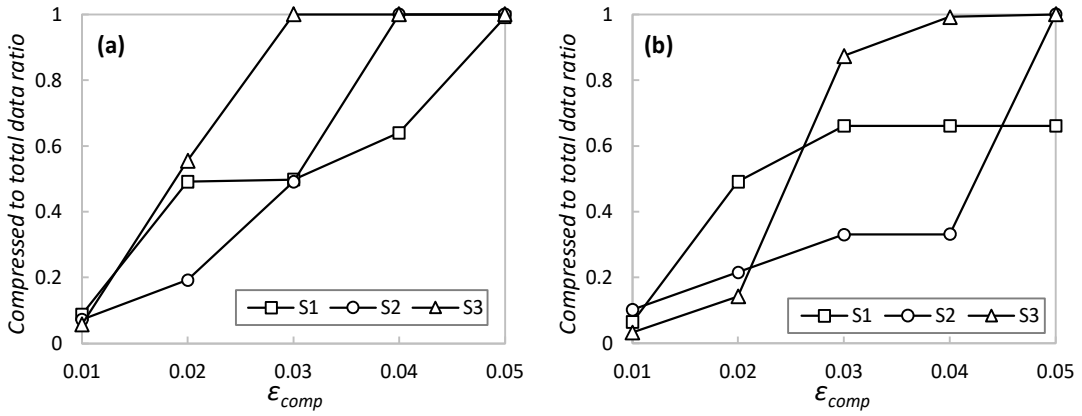


Figure 13: AAC performance under normal operation (a) and operation with perturbations (b)

Once the AAC has been presented as an adaptive and polyvalent method, let us now focus on evaluating its performance compared against two benchmarking methods. First, Figure 14 (a) compares AAC against a simplest method consisting in the single AE that better works for a given ε_{comp}, i.e., the one with the smallest LS size that always achieves reconstruction error not greater than ε_{comp}. Note that this

benchmarking method is easy to deploy in our system, provided that the required ε_{comp} does not (often) change in time, because every requirement variation could entail a new AE re-training to adjust LS size. The figure shows the compression factor, which is simply the inverse of the compression ratio, as well as the relative gain of AAC with respect to using the best AE in each case. In light of the results, we can conclude that AAC produces larger compression ratio than using a single AE, reaching remarkable relative gain above 60% for stringent reconstruction errors around 0.01. Recall that AAC can adapt to changes in ε_{comp}without the need of retraining AEs that combined with its high performance makes AAC as the best option for AE-based compression.

In order to have a second benchmarking evaluation, Figure 14 (b) compares the achieved compression ratio for two selected ε_{comp} values against the compression method presented in [Cha19], called LFZip. Similar to AAC, LFZip is a lossy compression method using fully connected neural network decoders that achieves good compression ratios. In [Cha19], the authors provide the achieved compression ratio for the selected target reconstruction errors using different time series data. The figure compares the performance of AAC (averaging all sensors under normal operation) and LFZip (results from [Cha19]), where large benefits of AAC can be observed. However, since we were not able to reproduce neither LFZip method with our sensor data nor AAC with the data in [Cha19] (due to the lack of algorithm details and data availability), the conclusions of such comparison are mild. For this reason, we included the relative gain of each method when ε_{comp} is relaxed from 0.01 to 0.05. In view of the values, we can state that AAC clearly outperforms LFZip in terms of adaptability to variable requirements and relative compression gain, as the compression factor of LFZip increased from 8 (equivalent to a 0.125 compression ratio) to 18 (equivalent to a 0.05), while AAC increased from a compression factor of 5 (0.2 compression ratio) to a compression factor of 60 (0.01 compression ratio).

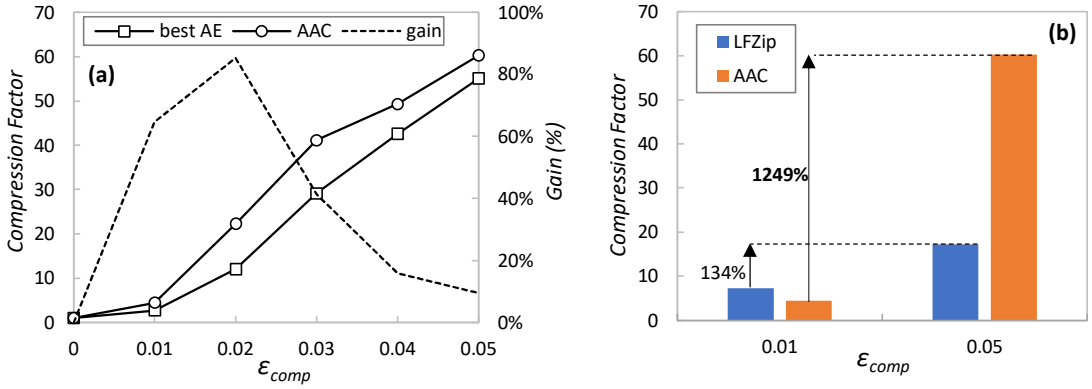


Figure 14: AAC benchmarking against best single AE (a) and LFZIP (b) methods.

It is worth noting that the outstanding AAC performance illustrated so far requires the availability of a pool of AEs specifically trained for each of the sensors. Once a

new sensor is installed in the physical system and telemetry is started being collected and processed by a new or existing DA, such specific AEs are not available until enough data have been collected. This is the reason why, as introduced in previous sections, our approach proposes initializing AAC with a pool of generic AEs trained with heterogeneous data, i.e., mix of data from other sensors available in the data lake. Figure 15 (a) compares the percentage of compressed samples using generic AEs trained with a mix of telemetry measurements of all sensors and specific AEs for each of the sensors individually. In both cases AAC has been configured with a stringent ε_{comp} = 0.01. As can be seen, generic AEs produce an overall good performance (around 50% of samples can be effectively compressed), although provides negligible benefit for sensors that behaves very different to considered generic data. This occurs in S2 data, which shows a clearly on-off period (recall the example in Figure 4) that vastly differs from generic data used for training. As soon as specific AEs can be trained, then both individual and overall compression increases (around 80% of samples can be compressed).

Finally, Figure 15 (b) details how many times every AE in the pool is used, for both generic and specific AE pools. Results show the average performance for all DAs and ε_{comp} = 0.01. Note that the smallest LS size is frequently selected; however, sometimes is needed a smaller compression (larger LS) to guarantee the target reconstruction error, which adds value to the proposed AAC method. Moreover, the use of larger AEs is reduced when specific AEs are trained. For this very reason, we can conclude that the use of generic AEs is useful to provide compression from the beginning of sensor operation but needs to be substituted by specific AEs to reach maximum performance.

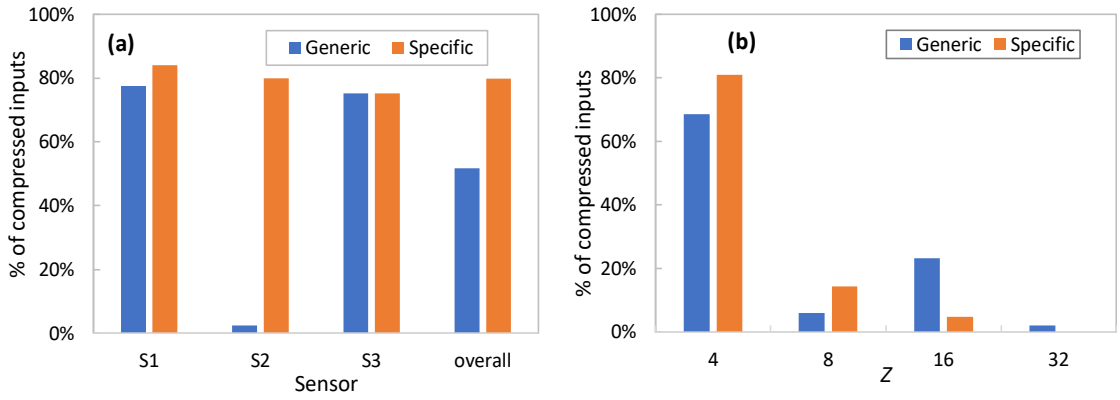


Figure 15: Generic and Specific AAC performance vs sensor type (a) and LS size (b)

6.3.3 Case Study

Eventually, we conducted a numerical case study in order to evaluate the impact of the proposed methodology assuming a larger network scenario like the one sketched in Figure 2 (a). Thus, we assume that a physical system containing hundreds to

thousands of sensors is geographically distributed among a number of locations, where the DAs are locally deployed. For the sake of simplicity, let us assume that the overall telemetry data generated by all the sensors in the system, i.e., the total volume that needs to be gathered by the DT, is fixed to 400 Gb/s. Moreover, let us assume an optical TN that allows transparent connectivity between the remote physical locations and the location where the DT is deployed, e.g., a datacenter. To support the transport of such telemetry data, optical connections taking advantage of digital subcarrier multiplexing technology can be deployed [Vel21]. This means that optical connections can be established with a fine granularity of 25 Gb/s each.

Figure 16 (a) shows the amount of data injected as a function of the number of locations, assuming an even split among locations of the total amount of telemetry data. Two cases are shown: no compression and using AAC. For the latter, we consider ε_{comp}= 0.01 and, according to Figure 14 and considering that sensors behave similarly to the ones used before, the average compression factor is around 12.5. Assuming this compression performance, the figure shows great savings in the total amount of data generated by every location distributed in the network. Nevertheless, the impact on the true amount of data that needs to be conveyed in the TN will depend on the number of optical connections needed to carry out such data, which is shown in Figure 16 (b) as a function of number of locations, as well as the capacity savings of using AAC with respect to no compression scenario. For instance, 50% of optical capacity savings are achieved when 400 Gb/s of raw data are generated among 8 different locations. In this case, every location is generating around 50 Gb/s of raw telemetry data, which requires 2 optical connections between the location and the centralized DT. On the contrary, the proposed AAC method reduces conveyed data to 4 Gb/s, which can be served with only one optical connection per location.

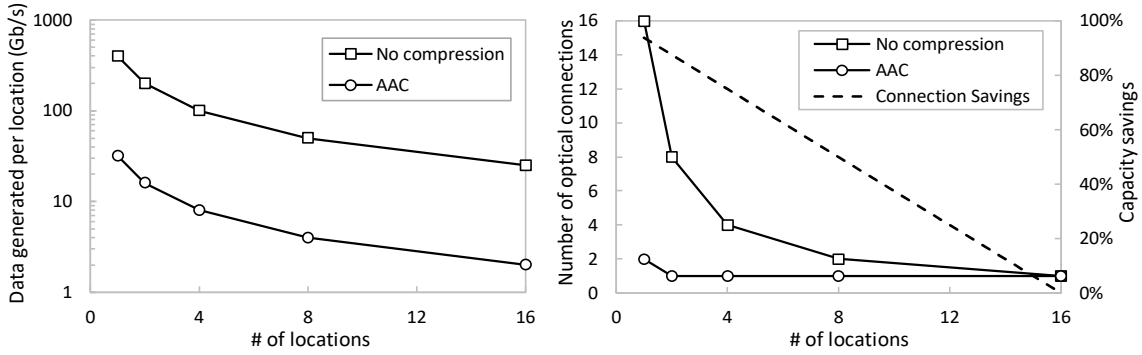


Figure 16: Network study analysis

Hence, we can definitely conclude that the proposed adaptive telemetry compression mechanism allows a large reduction on the number of optical connections and true data to be conveyed through the TN.

6.3.4 Comparison to IBE

Table 9 compares the main parameters of IBE algorithm (component of ZIZO) described in Chapter 5 and ACC presented in this chapter. While the IBE algorithm is a similarity-based, low complexity algorithm that can be used on sensors with small computing resources, AAC is a more complex method that relies on ML to produced more efficient results, however it requires more computing resources and can be deployed on more advanced sensors. is a comparison of the parameters of the simulation performed using both methods.

Table 9: Operation parameters of IBE and AAC

IBE (ZIZO)		**AAC**	
Period Size	32 and 64	*Input Size*	256
Application Criticality	High and Low	ε_{comp}	[0.01, 0.05]
Similarity Threshold	temperature: 0.07, 0.1, 0.2 humidity: 0.2, 0.5, 1 light: 10, 15, 25	*Latent Space Size*	4, 8, 16, 32

While the input size for ZIZO was variable between 32 and 64, it was fixed at 256 for AAC. This means longer data collection periods for AAC, however it makes it more scalable than ZIZO since IBE is constrained by the index bytes, while AAC can be scaled down to 128 or scaled up to 512 for example. The application criticality in ZIZO also is variable between Low and High, which is reflected and the adapted sensing periods, however the criticality of the application can be adjusted for by changing the ARE in AAC, from 0.01 being very strict and 0.05 being very loose. Finally, the size of the compressed data depends on the Similarity Threshold only for ZIZO, however in AAC, it depends both on the ARE and the available latent space sizes, with a possibility of having a compressed data size equal to 1% of the original data size if selected AE is the one with latent space size equal 4. Table 10 shows a comparison of the characteristics of each algorithm.

6.3.5 Integration with SRA

Since SRA operates at the CA level and uses decompressed sensor data to calculate redundancy in order to adjust the sensing frequency for the following period, and since both IBE and AAC operate on the DA level in a periodic data collection basis, AAC can be integrated with SRA easily.

Table 10: Comparison of the characteristics of ZIZO and AAC

	IBE (ZIZO)	**AAC**
Basis	Statistical Analysis	Machine Learning
Data Collection	Periodic	Periodic
Require Training	No	Yes
Adaptive	Yes	Yes
Scalable	No	Yes
Configurable	Yes	Yes

6.4 Concluding Remarks

In this chapter, we presented a smart adaptive telemetry data compression (AAC) for DT use case. The method made use of AEs trained with generic and specific sensor telemetry data, as well as a set of algorithms that use those AEs to maximize the performance of compression.

The numerical evaluation of such models and algorithms was performed using an experimental data set of a water distribution system. The main conclusions derived from such numerical analysis are: *i*) AAC produces smaller compression ratios than using a single AE, reaching remarkable relative gain above 60% for stringent reconstruction errors around 1%; *ii*) AAC achieves compression ratios one order of magnitude smaller than other benchmarking lossy compression mechanisms in literature.

Finally, the smart compression of telemetry data for DT use case is evaluated in terms of reduction of TN resources. To this aim, we considered distributed scenarios where telemetry data sources are spread among different network locations, thus needing to gather such telemetry data in a centralized location. Results showed that remarkable capacity savings measured in terms of dedicated optical connections are achieved for moderately high distributed scenarios.

Chapter 7

Autoencoder-based Telemetry Data Anomaly Detection

This chapter describes a method for two methods for AD using deep AEs that operate on both single and multiple time series, and their possible application within the architecture of a DT. Regarding the AD part, the first method, called *Single Sensor Anomaly Detection* (SS-AD), takes advantage of the specificity of the trained AEs to detect when the collected data contains an anomaly, e.g., if the sensor malfunctions or some kind of additional noise is introduced to the data from an external source. The second method, called *Multiple Sensor Anomalous Group Diagnosis* (MS-AGD), detects anomalies that can affect several sensors in a correlated way, even when they cannot be detected by SS-AD in independent time series. It does this by comparing data points with a certain degree of reconstruction error values across all the time series involved, making it able to detect subtle correlated anomalies.

7.1 Main Components

Figure 17 details the architecture previously sketched in Figure 1(b) and is an extension to Figure 12, adding to it the components involved in AD in addition to the previously described compression.

Besides the compressed telemetry data, the AAC process also computes the reconstruction error vector obtained by the selected AE (denoted as r_{st}). This error is defined as the difference between the original and reconstructed telemetry measurements. This relevant output is locally processed at the DA for AD purposes. Specifically, the *DA manager* receives a reconstruction error vector per each sensor and monitoring interval and triggers two different AD processes. On the one hand,

the SS-AD analyzes the individual reconstruction error of each sensor in order to find an anomalous error pattern such as continuous large error. On the other hand, the MS-AGD analyzes the reconstruction errors of the sensors in the device and performs a correlated analysis in order to identify subtle anomalies affecting several sensors at the same time. The diagnosis generated by each of the methods is then processed by the DA manager that generates a device diagnosis report when a remarkable event is detected by one or both methods.

Such device diagnosis report (if generated) is sent to the *CA manager* that can trigger a wider and deeper anomaly analysis. It can request to the DA of the devices under its control those reconstruction error vectors that have not been sent before. As an illustrative example, let us imagine that an anomaly in a temperature sensor has been detected in device *i*. The CA can then request the reconstruction error of the rest of temperature sensors of all the devices in the cluster, in order to perform a group analysis and detect e.g., an incipient temperature anomaly in other elements of the system. Note that, to allow this analysis, we consider that DA managers temporary store reconstruction errors even when they are not detecting any anomaly. Finally, the results of received device diagnosis reports generated by DA managers and the sensor group analysis (if proceed) generated by CA manager composes the cluster diagnosis report that is sent to the application manager.

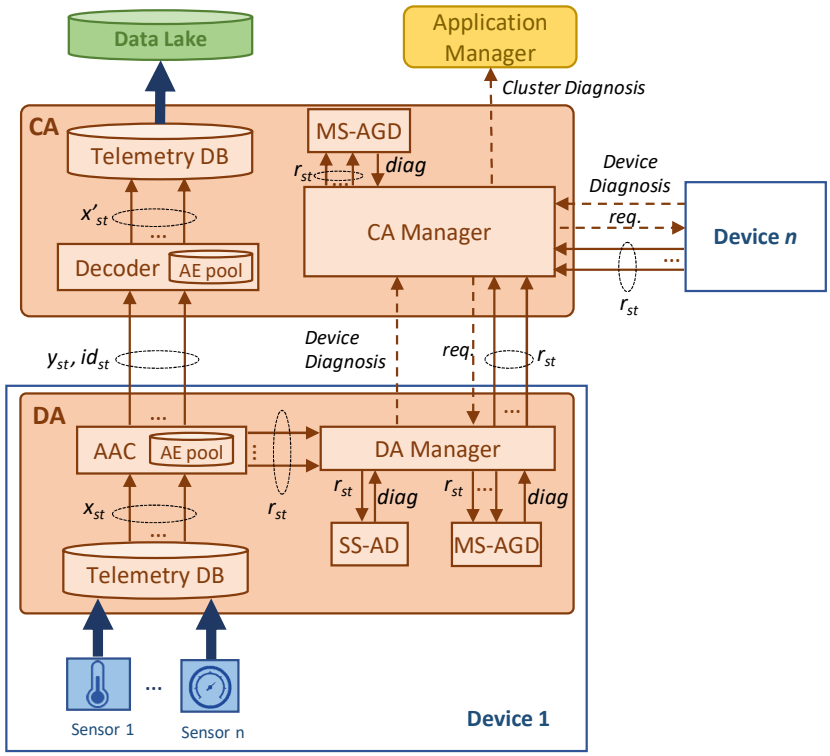


Figure 17: Detailed architecture and key components

7.2 Algorithms

7.2.1 AD Special Notation

The following table provides the main notation that is consistently used in the following algorithms, in addition to the previously introduced notation.

Table 11: Special Notation

ε_{anom}	Individual reconstruction error for AD
α	Number of consecutive error values above AD threshold
β	Number of total error values above AD threshold

7.2.2 SS-AD

Algorithm 4 details the pseudo-code of the SS-AD procedure that runs locally in the DA every time a new compressed telemetry stream is obtained and hence, a new reconstruction error vector r_{st} is available. Since the principle of AD using AEs relies in the fact that an anomalous input will be poorly reconstructed, an anomaly error detection threshold ε_{anom} is needed to perform such detection. Indeed, AD is triggered if either one of the following conditions is met: *i*) a number α of consecutive measurements produced a reconstruction error larger than threshold ε_{anom}; *ii*) a number β of total measurements (non-consecutive) produced a reconstruction error larger than threshold ε_{anom}.

Algorithm 4: SS-AD

```
INPUT: r_st, ε_anom, α, β
OUTPUT: anomaly
1.   k_cons, k_total ← 0
2.   for each r ∈ r_st do:
3.       if r > ε_anom then
4.           k_cons ← k_cons + 1
5.           k_total ← k_total + 1
6.       if k_cons == α or k_total == β then
7.           return true
8.       else
9.           k_cons ← 0
10.  return false
```

The algorithm starts by initializing the counters of consecutive and total number of measurements above the error threshold (line 1 in Algorithm 4). Then, each single error value in r_{st} vector is evaluated and compared with the threshold (lines 2-3).

When the error exceeds the thresholds, then both counters are increased in one unit (lines 4-5). At this point, it is worth checking if one of the AD conditions is met and if so, the procedure stops and returns an AD event (lines 6-7). On the contrary, it is necessary to reset the counter of consecutive values above the threshold before analyzing the next error value (lines 8-9). Finally, no anomaly event is returned in case that none of the conditions is met (line 10).

7.2.3 MS-AGD

The pseudo-code of MS-AGD procedure is detailed in Algorithm 5, which aims at computing a score that increases when several sensors within a group generate high reconstruction error at the same time. Indeed, this score has the form of a vector of w positions, indicating the score at a given time unit within the analyzed monitoring interval (which allows fine multiple AD analysis). Moreover, recall that the MS-AGD can be executed at device level e.g., analyzing all (or a subset) of the sensors of a given device, or at cluster level e.g., analyzing all (or a subset) of the sensors in each cluster. Regardless of the case, let us consider that the reconstruction errors vectors obtained at a given monitoring time interval of a given group of sensors is denoted as R. This is the main input of MS-AGD, which also requires the specific parameter γ that defines the time interval size needed to compute the score.

Algorithm 5: MS-AGD

INPUT: $R = \{r_{st}, \forall s \in S_g\}$, γ
OUTPUT: score

```
1.  score ← zeros(w)
2.  Q ← zeros (|S_g|, w)
3.  i ← 0
4.  for s ∈ S_g do
5.      i ← i + 1
6.      r_avg ← avg (R.r_st)
7.      for j == 1 … w do:
8.          if R. r_st [j] > r_avg then
9.              Q[i,j] ← 1
10.     for j == γ … w do
11.         a ← sum (Q[:,j]) / S_g
12.         b ← dotproduct (Q[:,j − γ + 1:j]) / γ
13.         score [j] ← a.b
14. return score
```

The first step is to initialize the score vector, as well as the auxiliary matrix Q that is going to facilitate score computation (lines 1-2 of Algorithm 5). In particular, Q is a sparse 0-1 matrix, where cell $\langle i|j\rangle$ is 1 if and only if the sensor i at time unit j took a measurement above the average value of that sensor within monitoring interval t.

After computing Q (lines 3-9), the score is computed for every time unit within monitoring interval (lines 10-13). The score of each time unit i is the product of components a and b. On the one hand, a is the normalized sum of 1s in Q that time i.e., which proportion of sensors generate a measurement that produced a reconstruction error above the average. On the other hand, b computes the normalized dot product of Q in the last γ time units. Note that a large value indicates that there are consecutive time units where several sensors are above the average. In particular, $b = 1$ when all sensors in S_g stay above average reconstruction error during a consecutive number γ of time units.

To better understand the rationale behind MS-AGD score, Figure 18 shows the reconstruction error r_{st} and the score in a monitoring time interval of $w = 20$ of an example with three sensors. Three different cases are depicted, assuming $\gamma = 5$: i) error stays constant and low for all the time and sensors (no anomaly, Figure 18 (a)); ii) error increases in all the sensors but not at the same time (non -correlated subtle anomaly, Figure 18 (b)); iii) error increases in all the sensors and partially coincide in time (correlated subtle anomaly, Figure 18 (c)). For the sake of simplicity, average reconstruction error is around 0.5% in all the sensors in Figure 18 (a) and around 1.5% in all the sensors in Figure 18 (b) and Figure 18 (c). Colored circles indicate when reconstruction error is above the threshold. As can be observed, score reaches significant values (above 0.5) only when several sensors exceed average reconstruction error at the same time.

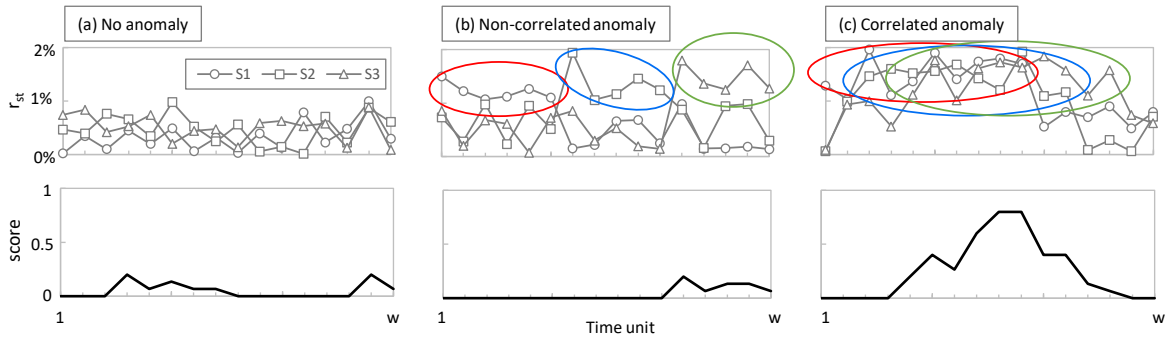


Figure 18: Example of Score

7.3 Performance Evaluation

In this section, we first introduce the simulation environment developed to evaluate the methods and algorithms presented in previous sections. Then, we analyze the performance of SS-AD, and MS-AGD using telemetry data from a real physical system.

7.3.1 Simulation Environment

The simulation environment used for testing SS-AD and MS-AGD is the same used to test AAC and described int the previous chapter with a few key changes. The AE used to calculate the reconstruction error of the series was the one with the largest LS i.e., *Z=32*. This is because the AD depends on the analysis of the reconstruction error, and thus we wanted to avoid any higher error resulting from smaller LS sizes and not due to actual data anomalies.

7.3.2 SS-AD and MS-AGD performance

In this section we focus on evaluating the performance of AD procedures assuming that specific AE are already trained and working. In particular, we configured our simulator to reproduce two different use cases: *i*) large individual anomalies for SS-AD evaluation, and *ii*) subtle time-correlated anomalies for MS-AGD evaluation.

For the first use case, we assume that SS-AD is continuously running for each sensor during 9 hours of normal operation followed by a drastic change in the pattern of the generated data (happening at time t_{anom}). In order to introduce variety of anomalies, we consider that sensor Si starts generating at t_{anom} data similar to that of sensor S_j, being $[i \neq j]$ and $[i, j] \in \{1, 2, 3\}$, thus reproducing 6 different anomalies.

Figure 19 evaluates the percentage of false positive detected by each of the sensors as a function of different values of SS-AD parameters ε_{anom} and α (β was fixed to 100). A false positive is detected if SS-AD returns True during the period of normal operation, i.e., when no anomalies are introduced. Considering the results, we can conclude that increasing ε_{anom} to 0.10 is sufficient to reduce α to short values (25 for *S1* and *S2*, and 10 for S3) that lead to zero false positive detection. Note that, the shorter α is, the faster the detection of true anomalies.

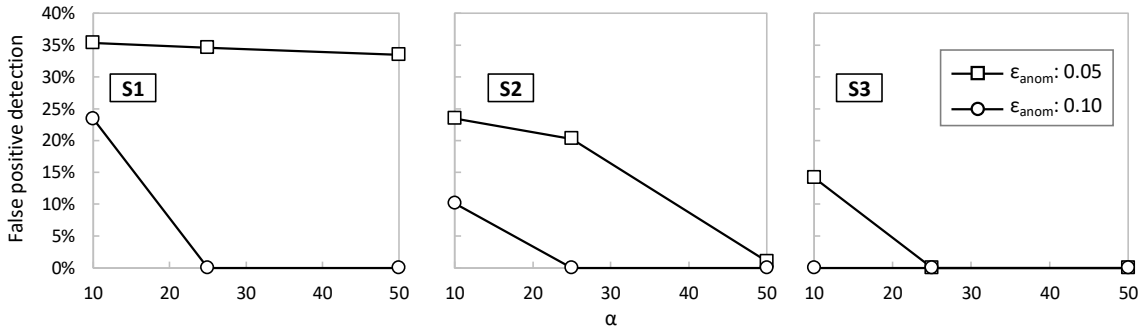


Figure 19: SS-AD: false positive detection

Then, assuming the best configuration of parameters for every sensor, Table 12 shows the detection accuracy of all the anomalies. It is worth noting that SS-AD achieves very high accuracy (>95%) for most of the considered anomalies. Indeed,

only *S1* SS-AD process is not able to detect *S3*-like data, which is reasonable due to the similarity of both *S1* and *S3* time series. Therefore, we can conclude that SS-AD performs accurate and robust detection of individual anomalies.

Table 12: SS-AD: best configuration and AD accuracy

Si	ε_{anom}	**α**	S_j		
			S1	**S2**	**S3**
S1	0.1	25	-	95.7%	0%
S2	0.1	25	95.4%	-	99.9%
S3	0.05	25	95.6%	95.5%	-

Regarding the second use case, we took advantage of WADI dataset measurements collected under perturbations that were intentionally introduced in the system. The available metadata clearly indicates the time where a perturbation starts, which we identified as t_{anom}. Figure 20 plots the three sensors' data in time period before and after t_{anom}, as well as the score computed in all such period. In view of the results, we can conclude that the proposed score clearly identifies when the correlated anomaly starts (no false positive detection is observed before t_{anom}). Note that the first-time interval where score reaches a value significantly larger than 0 is only 40 seconds later than t_{anom}, which validates MS-AGD as a prompt time-correlated AD method.

7.4 Concluding Remarks

In this chapter, we presented a two method for sensor telemetry data AD within the DT use case: single sensor anomaly detection (SS-AD) and multiple sensor anomaly diagnosis (MS-AGD). The two methods made use of AEs trained with specific sensor telemetry data, as well as a set of algorithms that use those AEs to maximize the performance of anomaly detection.

The numerical evaluation of such models and algorithms was performed using an experimental data set of a water distribution system. The main conclusions derived from such numerical analysis are: *i*) SS-AD achieves anomaly detection accuracy larger than 95% when telemetry data anomalies are injected; and *ii*) MS-AGD is able to perform prompt detection (<1 min) of subtle correlated anomalies affecting a group of sensors.

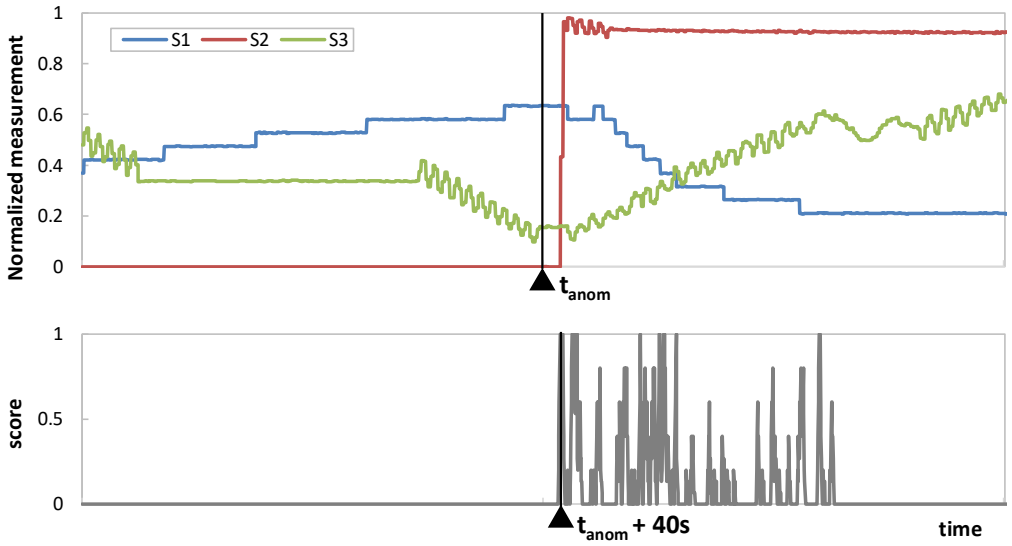


Figure 20: MS-AGD performance

Chapter 8

Knowledge Transfer for Private Data Sharing between DTs

This chapter goes beyond the domain of a single DT and describes a method to improve the operational efficiency through KT for several DTs operating under the same TN. The method, called Privacy-Ensuring Knowledge Transfer (PEKT), employs two layers of AEs to obfuscate data from one DT before sharing it with the other DT. It does this while keeping it possible to extract specific, intended insight into the state of the sharing DT, while ensuring that critical information remain hidden and unextractable from the shared data, and thus ensuring privacy.

The chapter presents a first section where the main PEKT concept is sketched. Then, an algorithm to implement PEKT for DT environments similar to the ones presented in previous chapters is detailed. Finally, some preliminary results are presented, showing the potential benefits of applying KT for secure and efficient DT coordination.

8.1 PEKT Concept

The concept of PEKT is sketched in Figure 21 with a simple example involving two physical systems (A and B). Each physical system has sensors collecting telemetry data specific to its operation and sending its private data to the respective DT through the TN to be stored in its respective data lake. However, physical system A collects some telemetry data that is of the same nature than physical system B collects and may be of interest to it.

Let us provide an example of KT between two different systems, where physical system A is a water distribution system and B is a power grid system. The data

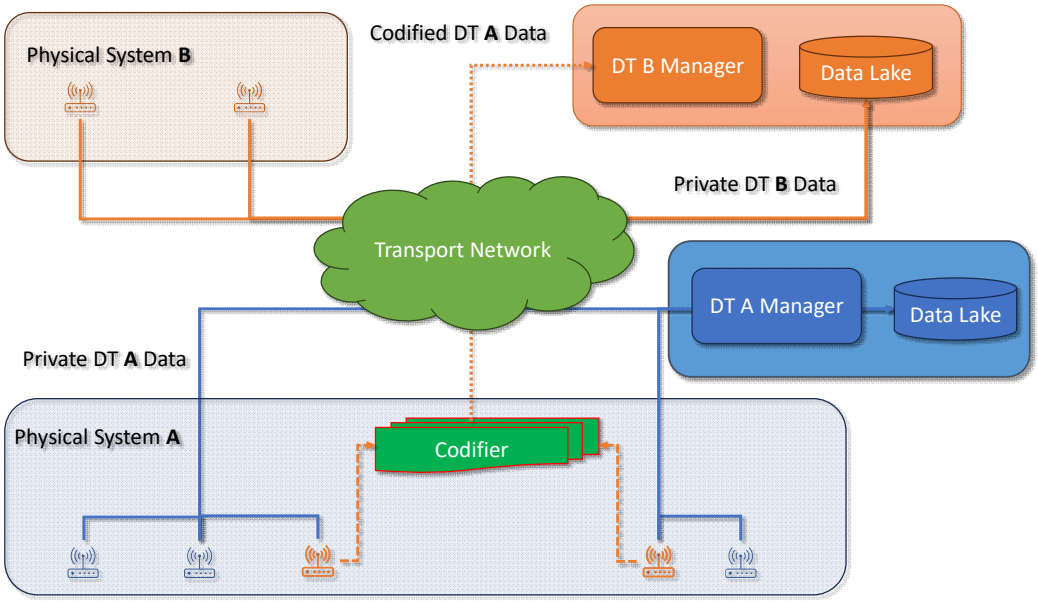


Figure 21: PEKT Concept

collected by DT of water distribution system contain measurements such as pressure, volume of water pumped, etc. as well as data from sensor measuring power consumption of devices such as pumps. The latter type of measurements are of special interest for the DT of the power grid e.g., to anticipate power consumption peaks or better adapt production to power demand profiles. However, the water distribution DT does not want to share the detailed power consumption profile of the pump.

In view of that, the data gathered by the DT of physical system A is passed to a codifier, that generates some codified data that would contain events relevant for the system B without revealing critical information. For instance, when certain changes in the power consumption profile occurs, they are sent to the power grid DT which uses it to anticipate how the power consumption demand would be affected at the network grid node where the pump is located.

More formally, the codifier generates a codified representation of the data called digest, that is shared directly with DT B manager, which uses it gain insight into the operation of physical system A. The digest is generated by a double layer of obfuscation, which ensures that the shared data is safe from data extraction attacks which can be used to reconstruct the private data generated by the sharing DT. The first layer is the layer of the AE trained to compress the data. The encoders of these compress the data into a set of latent features that have the key information, and that can be used by the decoder to reconstruct the original data with a low level of reconstruction error. However, even without the decoder, some analysis of the latent

features can be done, such as explainability studies, and insights may be gained into the data, so sharing the latent space is not safe and does not ensure hiding all the details from the receiver. The second layer of obfuscation is the layer of a ML-based regression model that is trained to predict the values of the sensor data that needs to be shared using the data from other sensors correlated to it and are collecting data in proximity to it.

8.2 PEKT Training Algorithm

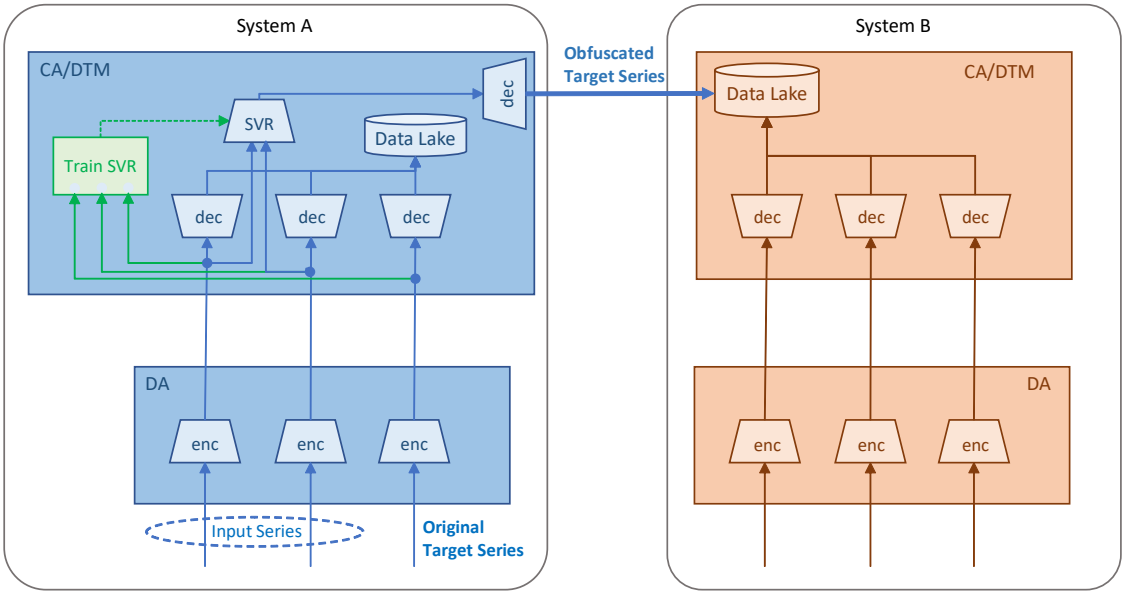


Figure 22: Double obfuscation mechanism

Figure 22 shows the codification mechanism based on double obfuscation. In physical system A there are two types of data: the data of the relevant sensor to be shared, i.e., the target time series data, and the data of the correlated sensors, i.e., the input time series data. During training phase of the regression model (green arrows in system A), the latent space of both input and target series is used to train a Support Vector Regressor (SVR) model that predicts the latent space of the target with some error. This prediction is the one that is shared with system B as obfuscated target series.

The process of training the codifier is shown in Algorithm 6, that uses as input series (denoted as x), the target series (denoted as x') and the set of trained AE for both the input series and the target series (denoted as Ψ'_s). We first normalize the target series (line 1 in Algorithm 6) and then generate its set of latent features y' (line 2), and flatten it to obtain a continuous latent features series that would serve as the

training target series (line 3). Then, we initialize the empty matrix of input series (line 4), and do the same for each of the input series, stacking the result vertically in the matrix in a way that the values in the same instance are in the same row (line 5-9). Then, we calculate the mean, generating a single time series which will act as a new single input series (line 10). An SVR model is initialized (line 11), and resultant input and target series are used to train it (line 12).

Algorithm 6: Codifier Training

```
INPUT: {x, ∀s ∈ S_g}, x', {Ψ_s, ∀s ∈ S_g}, Ψ'_s
OUTPUT: K
1.   x' ← normalize (x', minmax)
2.   y' ← Ψ'_s.encoder.propagate (x')
3    y' ← flatten(y')
4.   S ← Φ
5.   for each  Ψ  ∈ Ψ_s  do:
6.       x ← normalize (x, minmax)
7.       y ← Ψ .encoder.propagate (x)
8        y ← flatten(y)
9.       S ← vstack(S,y)
10.  S ← mean(S , axis=1)
11.  K ← SVR(kernal='rbf')
12.  K .fit(S, y')
13.  return K
```

As also shown in Figure 22, when a new window of data arrives from the input series, a similar data series would be created by applying the same transformations done to prepare that data to train the codifier, and this data series would be passed through the SVR model which would predict a set of latent features for the target series. The predicted latent features series is then passed through the decoder of the AE of the target series, generating a codified representation that only contains indications of what actually happened in the target series.

8.3 Performance Evaluation

8.3.1 Simulation Environment

In order to test the algorithm, used the same simulation environment we used for the previous 3 algorithms, the WADI dataset, and we selected a group of 4 sensors measuring values to monitor a single component, a water pump. The target series was the series of the sensor monitoring the power consumption of the pump, while the other 4 measured, pressure, voltage, and water volume. Figure 23 shows the selected series.

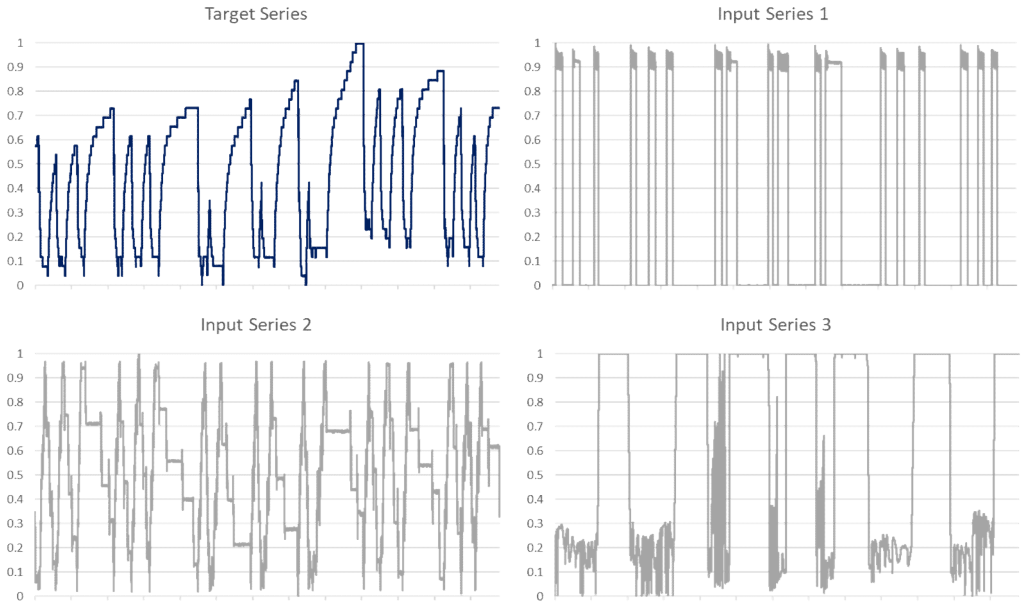


Figure 23: The target series and the 3-input series

8.3.2 Results

We first generated the set of latent spaces for the 4 series. The target and the input series, each having 256000 data points, we split into 1000 windows of size 256, to fit the input size of their respective 32LS AE. The we used their AEs to obtain the dataset of latent features, which means for each series a matrix of dimension 1000x32, and then flattened the matrix to obtain 4 new series each having 32000 data points. Then we split the newly generated 4 series, and into 3 segments: training (70%), validation (20%) and testing (10%). The training and validation segments were used to train the SVR model. The parameters of the SVR were not fine-tuned, as very accurate prediction was not intended, but the exact opposite, which is to introduce some noise to the data, so the parameters were tuned to generate a prediction error of just less than 0.1. Next, we used the classifier to generate the predicted values of the LS of the target series test segment. Figure 24 shows the normalized values of both the original LS values of the target series and the ones predicted by the classifier. As the figure shows, the values are drastically different, and thus sufficient ambiguity was introduced to the LS.

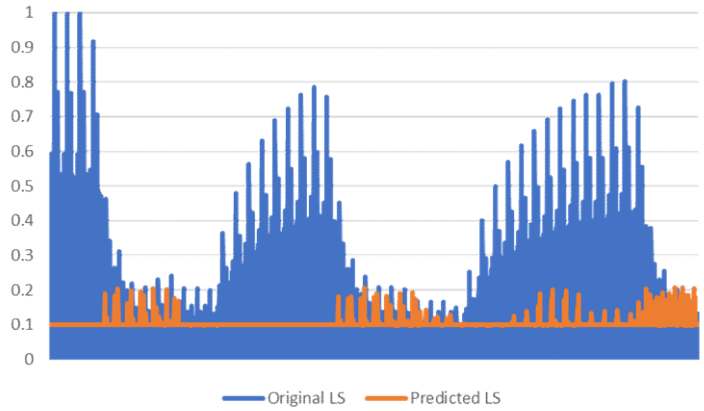


Figure 24: The normalized original and predicted LS values for the target series

Next, we split the values of the LS series into windows of 32, the input size of the decoder of the AE of LS32 of the target series and generated the decoded values which were concatenated to generate the codified segment of the data. Figure 25 shows the original test segment and the codified segment.

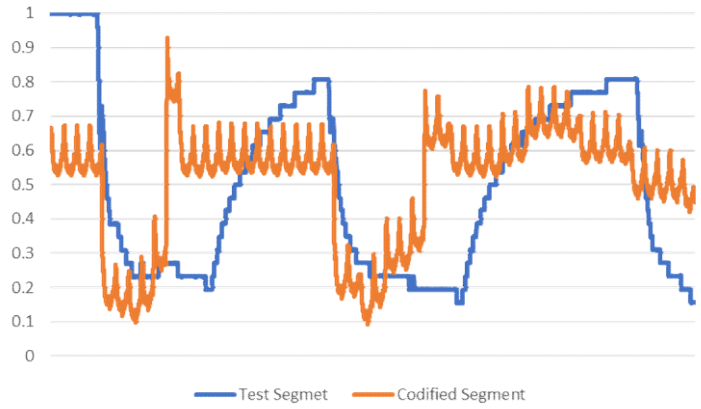


Figure 25: Original test segment and the codified segment of the target series

As the figure shows, the profile and values of the target series were completely hidden in the codified series. However, when sharp changes in the values of the original target series occur, a proportional change in the values of the codified series happens, which means that the direction and magnitude of change can be inferred by studying the codified series.

8.4 Concluding Remarks

In this chapter, we introduced a method for KT among DT operating under the same TN for the purpose of improving coordination and increasing the operational efficiency while ensuring privacy for the sharing party. The method is called PEKT, and it utilizes two levels of AEs to codify relevant information that is then passed on to the receiving party, structuring the shared data in a way that keeps the intended insight extractable.

The proof of concept of the algorithm showed that the codifier can create a digest that shows the pattern and magnitude of changes occurring in a certain time series, while almost entirely changing the profile of the series in a way that data extraction attacks are not possible.

Chapter 9

Closing Discussion

9.1 Main Contributions

This PhD thesis focused on applying several methods for smart and efficient management of SN for B5G applications. The main contributions and conclusions of this work are next detailed.

First, in Chapter 5, we proposed the ZIZO mechanism to minimize the data size. ZIZO is based on the cluster network architecture and operates on the two levels of a SN: i) Sensor Level: using a low complexity, energy efficient data compression technique called IBE. ii) CA Level: a sampling frequency adaptation technique based on statistical similarity study to minimize redundancy within collected data in a group of close proximity sensors called SRA. Through simulations, we evaluated the performance of our mechanism in terms of minimizing data transmission and energy consumption while comparing them to other techniques and showed that we can achieve a compression ratio of down to 0.05, and an energy consumption ratio percentage of down to 8%.

In Chapter 6, we presented an adaptive, AE-based telemetry data compression (AAC) method within the DT use case. The method made use of AEs trained with generic and specific sensor telemetry data, and the algorithm was optimized to maximize the compression efficiency. The numerical evaluation of such models was performed using an experimental data set of a water distribution system. As main conclusions, AAC produces smaller compression ratios than using a single AE, reaching remarkable relative gain above 60% for stringent reconstruction errors around 1%. Also, AAC achieves compression ratios one order of magnitude smaller than other benchmarking lossy compression mechanisms in literature.

In Chapter 7, we, presented AE-based single sensor anomaly detection (SS-AD) and multiple sensor anomaly (MS-AGD) for AD within the DT use case. The methods were trained with data from the same WADI dataset used to train the compression models, make use of the section of the data collected under cyber-attacks to simulate anomalies to detect. SS-AD achieves anomaly detection accuracy larger than 95% when telemetry data anomalies are injected, while MS-AGD is able to perform prompt detection (<1 min) of subtle correlated anomalies affecting a group of sensors.

Finally, in Chapter 8, we introduced the PEKT concept and detailed how it can be used to share information between different DTs in order to achieve some coordination and preserving privacy. Both AE and SVR classifier-based algorithm to codify sensor data beyond recognition were proposed. Preliminary results show that the patterns and magnitude of changes found in original data are preserved in a private way in the obfuscated data shared.

9.2 List of Publications

9.2.1 Publications in Journals

[IEEESJ21] **A. El Sayed**, H. Harb, M. Ruiz and L. Velasco, "ZIZO: A Zoom-In Zoom-Out Mechanism for Minimizing Redundancy and Saving Energy in Wireless Sensor Networks," *IEEE Sensors Journal,* vol. 21, no. 3, pp. 3452-3462, 2021.

[Sensor23] **A. El Sayed**, M. Ruiz, H. Harb and L. Velasco, "Deep Learning-Based Adaptive Compression and Anomaly Detection for Smart B5G Use Cases Operation," *IEEE Sensors Journal,* vol. 23, p. 1043, 2023.

9.2.2 Publications in Conferences

[ICTON23] **A. El Sayed**, M. Ruiz and L. Velasco, "Comparing a Statistical Similarity Based and an Autoencoder Based Approach for Time Series Data Size Reduction," in Proc. *International Conference on Transparent Optical Networks ICTON*, Bucharest, 2023.

9.3 List of Research Projects

9.3.1 EU Funded Projects

- Horizon Europe SNS JU DESIRE6G: Deep Programmability and Secure Distributed Intelligence for Real-Time End-to-End 6G Networks (G.A. 101096466).

9.3.2 National Funded Projects

- AEI/FEDER IBON: AI-Powered Intent-Based Packet and Optical Transport Networks and Edge and Cloud Computing for Beyond 5G (PID2020-114135RB-I00).
- AEI/FEDER TWINS: cogniTive 5G application-aware optical metro netWorks Integrating moNitoring, data analyticS and optimization (TEC2017-90097-R).

9.4 Collaborations

The work has been co-advised by Prof. Marc Ruiz (Universitat Politècnica de Catalunya) and Prof. Hassan Harb (American University of the Middle East).

9.5 Topics for Further Research

The work should be extended to sensors collecting media data in addition to the scalar data discussed in this thesis. Moreover, KT and PEKT concepts need to be extended to other use cases and deserves exhaustive performance analysis to reach solid conclusions.

References

[Abd20] M. Abdulkarem, K. Samsudin, F. Z. Rokhani and M. F. A Rasid, " Wireless sensor network for structural health monitoring: A contemporary review of technologies, challenges, and future direction," *Structural Health Monitoring,* vol. 19, no. 3, pp. 693-735., 2020.

[Abd21] M. A. Abdo and Y. Y. Ala Edin Awadah, "A Fuzzy Logic Adaptive Image Resize (Resolution) Level using Cross-Layering in Wireless Multimedia Sensor Networks," *International Journal of Computer Applications,* vol. 3, no. 183, pp. 33-40, 2021.

[Abu14] M. Abu Alsheikh, S. Lin, D. Niyato and H.-P. Tan, "Machine Learning in Wireless Sensor Networks: Algorithms, Strategies, and Applications," *IEEE Communications Surveys & Tutorials,* vol. 4, no. 1996-2018, p. 16, 2014.

[Adv12] advanticsys, "Sg1000 hardware," 2012. [Online]. Available: http://www.advanticsys.com/wiki/index..

[AnJ15] J. An and S. Cho, "Variational Autoencoder based Anomaly Detection," *Special Lecture on IE,* vol. 2, no. 1, pp. 1-18, 2015.

[Aza18] A. Z. Al-Marridi, A. Mohamed and A. Erbad, "Convolutional Autoencoder Approach for EEG Compression and Reconstruction in m-Health Systems," in *International Wireless Communications & Mobile Computing Conference*, 2018.

[Aza20] J. Azar, A. Makhoul, R. Couturier and J. Demerjian, "Robust IoT time series classification with data compression and deep learning," *Neurocomputing,* vol. 398, pp. 222-234, 2020.

[Bah14] J. M. Bahi, A. Makhoul and M. Medlej, "A Two Tiers Data Aggregation Scheme for Periodic Sensor Networks," *Ad-Hoc and Sensor Wireless Networks,* vol. 21, no. 1-2, pp. 77-100, 2014.

[Bar19] B. R. Barricelli, E. Casiraghi and D. Fogli, "A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications," *IEEE Access,* vol. 7, pp. 167653-167671, 2019.

[Baş19] M. Başaran, S. Schlupkothen and G. Ascheid, "Adaptive Sampling Techniques for Autonomous Agents in Wireless Sensor Networks," in *IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, 2019.

[Bis06] C. M. Bishop, Pattern Recognition and Machine Learning, New York: Springer New York, 2006.

[Bla18] D. Blalock, S. Madden and J. Guttag, "Sprintz: Time Series Compression for the Internet of Things," *Proceedings of the Association for Computing Machinery on Interactive, Mobile, Wearable and Ubiquitous Technologies,* vol. 2, no. 3, pp. 1-23, 2018.

[Bod04] P. Bodik, W. Hong, C. M. S. Guestrin, M. Paskin and R. Thibaux, "Intel Lab Data," 2004. [Online]. Available: http://db.csail.mit.edu/labdata/labdata.html.

[Cha19] S. Chandak, K. Tatwawadi, C. Wen, L. Wang, J. Aparicio and T. Weissman, "LFZip: Lossy compression of multivariate foating-point time series data via improved prediction," 2019.

[Che19] S. Chen, S. Zhang, X. Zheng and X. Ruan, "Layered adaptive compression design for efficient data collection in industrial wireless sensor networks," *Journal of Network and Computer Applications,* vol. 129, pp. 37-45, 2019.

[Coo20] A. A. Cook, G. Mısırlı and Z. Fan, "Anomaly Detection for IoT Time-Series Data: A Survey," *IEEE Internet of Things Journal,* vol. 7, no. 7, pp. 6481-6494, 2020.

[Cre18] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta and A. A. Bharath, "Generative Adversarial Networks: An Overview," *IEEE Signal Processing Magazine,* vol. 35, no. 1, pp. 53-65, 2018.

[Doi20] K. Dionisis, C. Nakas, D. Vomvas and G. Koulouras, "Applications of Wireless Sensor Networks: An Up-to-Date Survey," *Applied System Innovation,* vol. 14, no. 3, 2020.

[Eic14] F. Eichinger, P. Efros, S. Karnouskos and K. Böhm, "A Time-Series Compression Technique and its Application to the Smart Grid," *The VLDB Journal,* vol. 24, no. 24, p. 193–218, 2014.

[Ero20] T. Erol, A. F. Mendi and D. Doğan, "The Digital Twin Revolution in Healthcare," in *International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2020.

[Fer14] S. Ferdoush and X. Li, "Wireless Sensor Network System Design Using Raspberry Pi and Arduino for Environmental Monitoring Applications," *Procedia Computer Science,* vol. 34, no. 1, pp. 103-110, 2014.

[Fin11] E. Fink and H. Gandhi, "Compression of Time Series by Extracting Major Extrema," *Journal of Experimental & Theoretical Artificial Intelligence ,* vol. 23, pp. 255-270, 2011.

[Fla09] A. Flammini, P. Ferrari, D. Marioli, E. Sisinni and A. Taroni, "Wired and wireless sensor networks for industrial applications," *Microelectronics Journal,* vol. 40, no. 9, pp. 1322-1336, 2009.

[Fue20] P. C. Fuertes, F. M. Alzamora, M. H. Carot and J. A. Campos, "Building and Exploiting a Digital Twin for the Management of Drinking Water Distribution Networks," *Urban Water Journal,* vol. 17, no. 8, pp. 704-713, 2020.

[FuX19] X. Fu, G. Fortino, W. Li, P. Pace, and Y. Yang, "WSNs-assisted opportunistic network for low-latency message forwarding in sparse settings," Future Generation Computer Systems, vol. 91, pp. 223–237, 2019

[FuX20] X. Fu, G. Fortino, P. Pace, G. Aloi, and W. Li, "Environment-fusion multipath routing protocol for wireless sensor networks," Information Fusion, vol. 53, pp. 4–19, 2020

[Gao21] Y. Gao, S. Qian, Z. Li, P. Wang, F. Wang and Q. He, "Digital Twin and Its Application in Transportation Infrastructure," in *IEEE 1st International Conference on Digital Twins and Parallel Intelligence (DTPI)*, 2021.

[Ghr20] Z. Ghrib, R. Jaziri and R. Romdhane, "Hybrid approach for Anomaly Detection in Time Series Data," in *International Joint Conference on Neural Networks*, 2020.

[Goo16] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016.

[Gul22] K. Gulati, R. S. Kumar Boddu, D. Kapila, S. L. Bangare, N. Chandnani and G. Saravanan, "A review paper on wireless sensor network techniques in Internet of Things," *Materials Today,* vol. 51, no. 1, pp. 161-165, 2022.

[Guo18] Y. Guo et al., «Multidimensional Time Series Anomaly Detection: A GRU-based Gaussian Mixture Variational Autoencoder Approach,» chez *PMLR*, 2018.

[Har17] H. Harb and A. Makhoul, "Energy-efficient sensor data collection approach for industrial process monitoring," *IEEE Transactions on Industrial Informatics,* vol. 14, no. 2, p. 661–672, 2017.

[Hei00] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," *Proceedings of the*

33rd Annual Hawaii International Conference on System Sciences, Maui, HI, USA, 2000, pp. 10 pp. vol.2

[Hei02] W. B. Heinzelman, A. P. Chandrakasan and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," in *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660-670, Oct. 2002

[Hin06] G. E. Hinton and S. R. R., "Reducing the Dimensionality of Data with Neural Networks," *Science,* vol. 313, no. 5786, pp. 504-507, 2006.

[Hol17] G. Hollmig et al., «An Evaluation of Combinations of lossy Compression and Change-Detection Approaches for Time-Series Data,» *Information Systems,* vol. 65, pp. 65-77, 2017.

[Hsu17] D. Hsu, "Time Series Compression Based on Adaptive Piecewise Recurrent Autoencoder," *CoRR,* 2017.

[Int20] International Telecommunications Union, "Representative use cases and key network," 2020.

[iTr19] iTrust, Centre for Research in Cyber Security, Singapore University of Technology and Design, *Water Distribution Dataset,* Singapore, 2019.

[Iva20] S. Ivanov, K. Nikolskaya, G. Radchenko, L. Sokolinsky and M. Zymbler, "Global Smart Industry Conference (GloSIC)," in *Digital Twin of City: Concept Overview*, 2020.

[Jan19] S. Jancy and C. Jayakumar, "Sequence statistical code based data compression algorithm for wireless sensor network," *Wireless Personal Communications,* vol. 106, pp. 971-985, 2019.

[Kim18] D. Kim and al., "Squeezed Convolutional Variational AutoEncoder for Unsupervised Anomaly Detection in Edge Device Industrial Internet of Thing," in *International Conference on Information and Computer Technologies*, 2018.

[Kus17] Kusner, M. J., Paige, Brooks and J. M. Hernandez-Lobato, "Grammar Variational Autoencoder," in *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[Lia14] Y. Liang and Y. Li, "An efficient and robust data compression algorithm in wireless sensor networks," *IEEE Communications Letters,* vol. 18, no. 3, pp. 439-442, 2014.

[LiL21] L. Li, J. Yan, H. Wang and Y. Jin, "Anomaly Detection of Time Series with Smoothness-Inducing Sequential Variational Auto-Encoder," *IEEE Transactions on Neural Networks and Learning Systems,* vol. PP, pp. 1-15, 2021.

[Mad15] S. Madakam, R. Ramaswamy and S. Tripathi, "Internet of Things (IoT): A Literature Review," *Journal of Computer and Communications,* vol. 3, pp. 164-173, 2015.

[Maj22] e. a. Majid, "Applications of Wireless Sensor Networks and Internet of Things Frameworks in the Industry Revolution 4.0: A Systematic Literature Review," *Sensors,* vol. 22, no. 6, 2022.

[Ngu21] H. X. Nguyen, R. Trestian, D. To and M. Tatipamula, "Digital Twin for 5G and Beyond," *IEEE Communications Magazine,* vol. 59, no. 2, pp. 10-15, 2021.

[Ott21] D. W. Otter, J. R. Medina and J. K. Kalita, "A Survey of the Usages of Deep Learning for Natural Language Processing," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 32, no. 2, pp. 604-624, 2021.

[Pan10] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345-1359, Oct. 2010, doi: 10.1109/TKDE.2009.191.

[Pet20] R. Pethuru and P. Evangeline, "The industry use cases for the Digital Twin idea," *Advances in Computers,* vol. 117, no. 1, pp. 79-105, 2020.

[QiQ18] Q. Qi and F. Tao, "Digital Twin and Big Data Towards Smart Manufacturing and Industry 4.0: 360 Degree Comparison," *IEEE Access,* vol. 6, pp. 3585-3593, 2018.

[Rat21] M. M. Rathore, S. A. Shah, D. Shukla, E. Bentafat and S. Bakiras, "The Role of AI, Machine Learning, and Big Data in Digital Twinning: A Systematic Literature Review, Challenges, and Opportunities," *IEEE Access,* vol. 9, pp. 32030-32052, 2021.

[Red16] K. K. Reddy, S. Sarkar, V. Venugopalan and M. Giering, "Anomaly Detection and Fault Disambiguation in Large Flight Data: A Multi-modal Deep Auto-encoder Approach," in *Annual Conference of the PHM Society*, 2016.

[Rui20] M. Ruiz, , F. Tabatabaeimehr, and L. Velasco, "Knowledge management in optical networks: architecture, methods, and use cases [Invited]," J. Opt. Commun. Netw. 12, A70-A81 (2020)

[Rus20] S. Russo, A. Disch, F. Blumensaat and K. Villez, "Anomaly Detection using Deep Autoencoders for in-situ Wastewater Systems Monitoring Data," in *Watermatex2019*, Copenhagen, 2020.

[Sha14] H. Sharma and S. Sharma, "A review of sensor networks: Technologies and applications," in *Recent Advances in Engineering and Computational Sciences (RAECS)*, 2014.

[Sha18] R. Sharma, S. Biookaghazadeh, and M. Zhao. 2018. Are existing knowledge transfer techniques effective for deep learning on edge devices? In 27th

International Symposium on High-performance Parallel and Distributed Computing (HPDC'18). ACM, New York, NY, 15–16.

[Sha19] O. Sharma, "Deep Challenges Associated with Deep Learning," in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, 2019.

[Sha21] H. Sharma, A. Haque and F. Blaabjerg, "Machine Learning in Wireless Sensor Networks for Smart Cities: A Survey," *Electronics,* vol. 10, no. 9, p. 1012, 2021.

[Shr19] A. Shrestha and A. Mahmood, "Review of Deep Learning Algorithms and Architectures," *IEEE Access,* vol. 7, pp. 53040-53065, 2019.

[Sin21] M. Singh, E. Fuenmayor, E. P. Hinchy, Y. Qiao, N. Murray and D. Devine, "Digital Twin: Origin to Future," *Applied System Innovation,* vol. 4, no. 2, p. 36, 2021.

[Sta08] J. A. Stankovic, "Wireless Sensor Networks," *Computer,* vol. 41, no. 10, pp. 92-95, 2008.

[Suh16] S., Suh, D. H. Chae, H. G. Kang and S. Choi, "Echo-State Conditional Variational Autoencoder for Anomaly Detection," in *International Joint Conference on Neural Networks*, 2016.

[Sun20] K. N. Sunilkumar, S. Shivashankar and K. Keshavamurthy, "Bio-Signals Compression Using Auto-Encoder," *International Journal of Electrical and Computer Engineering,* vol. 11, no. 1, p. 424, 2020.

[Ton16] S. Tonyali, O. Cakmak, K. Akkaya, M. M. E. A. Mahmoud and I. Guvenc, "Secure Data Obfuscation Scheme to Enable Privacy-Preserving State Estimation in Smart Grid AMI Networks," in *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 709-719, Oct. 2016.

[Tra22] H. -Y. Tran, J. Hu and H. R. Pota, "Smart Meter Data Obfuscation With a Hybrid Privacy-Preserving Data Publishing Scheme Without a Trusted Third Party," in *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 16080-16095, 1 Sept.1, 2022.

[Tsc18] M. Tschannen, O. Bachem and M. Lucic, "Recent Advances in Autoencoder-Based Representation Learning," *CoRR,* vol. abs/1812.05069, 2018.

[Tub03] M. Tubaishat and S. Madria, "Sensor networks: an overview," *IEEE Potentials,* vol. 22, no. 2, pp. 20-23, 2003.

[Vel19] L. Velasco et al, "Monitoring and Data Analytics for Optical Networking: Benefits, Architectures, and Use Cases," *IEEE Network,* vol. 33, no. 6, pp. 100-108, 2019.

[Vel21] L. Velasco and al., "Autonomous and Energy Efficient Lightpath Operation Based on Digital Subcarrier Multiplexing," *IEEE Journal on Selected Areas in Communications,* vol. 39, no. 9, pp. 2864-2877, 2021.

[Vuj14] V. Vujović and M. Maksimović, "Raspberry Pi as a Wireless Sensor node: Performances and constraints," in *International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2014.

[Wan19] T. Wang, Y. Lu, Z. Cao, L. Shu, X. Zheng, A. Liu and M. Xie, "When Sensor-Cloud Meets Mobile Edge Computing," *Sensors,* vol. 23, p. 5324, 2019.

[Wei16] Weiss, K., Khoshgoftaar, T.M. & Wang, D. A survey of transfer learning. *J Big Data* **3**, 9 (2016)

[Wes18] T. D. West and M. Blackburn, "Demonstrated Benifites of a Nascent Digital Twin," *Insight,* vol. 21, no. 1, pp. 43-47, 2018.

[Won18] T. Wong and Z. Luo, "Recurrent Auto-Encoder Model for Multidimensional Time Series Representation," in *International Conference on Learning Representations*, Vancouver, 2018.

[Yan16] J. Yang, S. Tilak and T. S. Rosing, "An Interactive Context-aware Power Management Technique for Optimizing Sensor Network Lifetime," in *SENSORNETS*, 2016.

[Yic08] J. Yick, B. Mukherjee and D. Ghosal, "Wireless Sensor Network Survey," *Computer Networks,* vol. 52, no. 12, pp. 2292-2330, 2008.

[Yoa10] Y. Yao, A. Sharma, L. Golubchik and R. Govindan, "Online Anomaly Detection for Sensor Systems: A Simple and Efficient Approach," *Performance Evaluation,* vol. 67, no. 11, pp. 1059-1075, 2010.

[You23] S. H. Bang, R. Ak, A. Narayanan, Y. T. Lee, H. Cho, Privacy-Preserving Knowledge Transfer through Partial Parameter Sharing," in *Computers Proceedings of the 5th Clinical Natural Language Processing Workshop*, pp. 19-30, 2023

[Zha17] C. Zhao and R. C. Paffenroth, "Anomaly Detection with Robust Deep Autoencoders," in *KDD*, 2017.

[Zhu20] F. Zhuang *et al.*, "A Comprehensive Survey on Transfer Learning," in *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43-76, Jan. 2021