ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

INGEGNERIA E TECNOLOGIA DELL'INFORMAZIONE PER IL MONITORAGGIO STRUTTURALE E AMBIENTALE E LA GESTIONE DEI RISCHI - EIT4SEMM

Ciclo 36

Settore Concorsuale: 09/E3 - ELETTRONICA

Settore Scientifico Disciplinare: ING-INF/01 - ELETTRONICA

HARDWARE-SOFTWARE CO-DESIGN FOR LARGE-SCALE STRUCTURAL HEALTH MONITORING

Presentata da: Amirhossein Moallemi

Coordinatore Dottorato

Luca De Marchi

Supervisore

Luca Benini

Co-supervisore

Davide Brunelli

Esame finale anno 2024

ALMA MATER STUDIORUM-UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF ELECTRICAL, ELECTRONIC AND INFORMATION ENGINEERING

'GUGLIELMO MARCONI'

36th Cycle Degree in ENGINEERING AND INFORMATION TECHNOLOGY FOR STRUCTURAL AND ENVIRONMENTAL MONITORING AND RISK MANAGEMENT - EIT4SEMM

DEI

Thesis in

Hardware-Software Co-design for Large-scale Structural Health Monitoring

CANDIDATE:

Amirhossein Moallemi

Advisor:

Prof. Dr. Luca Benini

Co-Advisor:

Dr. Davide Brunelli

Final Exam year 2024

Wisdom is a constant evolution of knowledge and experience.

ABSTRACT

Modern Structural Health Monitoring (SHM) systems currently utilize a combination of low-cost, low-energy sensors and processing units to monitor the conditions of target facilities. However, utilizing a dense deployment of sensors generates a significant volume of data that must be transmitted to the cloud, requiring high bandwidth and consuming substantial power, particularly when using wireless protocols. The current cloud-based solutions cannot scale if the raw data has to be collected from thousands of buildings. To optimize the energy budget and generated data of the monitoring system, it is crucial to reduce the size of the raw data near the sensors at the edge. However, existing compression techniques at the edge suffer from a trade-off between compression and accuracy and long latency, resulting in high energy consumption. This work presents a full-stack deployment of efficient and scalable data reduction and anomaly detection pipelines for SHM systems, which does not require transmitting raw data to the cloud but relies on edge computation. First, we design and evaluate an edge SHM sensor node featuring a low-cost MEMS-based sensor network, two computational units, and a wireless communication unit. Then, we benchmark three lightweight algorithmic approaches of anomaly detection, i.e., Principal Component Analysis (PCA), Fully-Connected AutoEncoder (FC-AE), and Convolutional AutoEncoder (C-AE) implemented on the SHM node. By doing so, we decrease network traffic by $\approx 8 \cdot 10^5\times$, from 780KB/hour to less than 10 Bytes/hour for a single node installation and minimize network and cloud resource utilization, enabling the scaling of the monitoring infrastructure. Further, in another framework, we show a parallelized version of an unconventional data reduction method suited for vibration analysis based on System Identification models, which does not require retraining. Featuring the parallel nature of this algorithm, it can leverage the unique capabilities of GAP9, a multi-core RISC-V MCU based on the parallel ultra-low power (PULP) architecture, making the System Identification deployable at the node level. Compared to the sequential implementation, we achieve a maximum execution time reduction of $\approx 60\times$ and power consumption of just $48.3\,\mathrm{mW}$ while preserving the spectral accuracy of the models. Finally, we propose our last SHM application to take a step forward in Traffic Load Estimation via the SHM system. This novel signal processing and classification pipeline is able to differentiate vehicles into three categories: light, i.e., less than 10 tons; heavy, i.e., between 10 and 30 tons; and super heavy, i.e., above 30 tons, using only features extracted from vibration data with an accuracy of 96%, utilizing the mean-shift, an unsupervised clustering model. This method can poten-

tially be a more cost-effective and scalable solution for monitoring bridge loads compared to Weight-in-Motion systems, as it leverages existing SHM infrastructure and low-cost MEMS sensors to provide real-time information on vehicular loads.

Real-life case studies over bridges in Italy demonstrate that by combining near-sensor computation of lightweight algorithms, smart pre-processing, and low-power wide-area network protocols (LPWAN), we can significantly reduce data communication and cloud computing costs, while anomaly detection accuracy is not adversely affected at the edge.

ACKNOWLEDGMENTS

I first would like to thank my supervisor, Luca Benini, for all his support, input, and valuable supervision during my PhD journey.

I also would like to thank Michele Magno, Tommaso Polonelli, Andrea Acquaviva, Victor Javier Kartsch Morinigo, and Davide Brunelli for providing essential suggestions and opportunities to learn from the projects that I was involved in during my PhD.

Further, I would also like to thank all my colleagues in unibo at Micrel Lab, specially Flavio Di Nuzzo, Alessio Burrello, Marco Guermandi, Luca Zanatta, Emanuele Parisi, and ETH Zurich (PBL group) Denis Mikhaylov, Luca Pascarella, and Michael Jost for their contribution to what I was able to achieve and learning process during my Ph.D.

Next, I also want to mention the spiritual support I got from my dear friends Laura Zunarelli, Sebastian Frey, Julian Moosmann, Nazareno Bruschi, Seyed Ahmad Mirsalari, and Amirhossein Kiamarzi during these years

Finally, I would like to give special thanks to my parents, who dedicated their unconditional support during my three years of Ph.D., and my sister, who helped me in numerous ways to achieve the best.

CONTENTS

Abstract ii

Contents vi

List of Tables vii

List of Figures viii

1 Introduction 1
- 1.1 Main Contributions . . . 3
- 1.2 Manuscript organization . . . 4

2 Literature Review 6
- 2.1 Accelerometers Technology . . . 7
- 2.2 SHM frameworks . . . 8
- 2.3 Anomaly Detection at the edge . . . 9
  - 2.3.1 Statistical data modeling . . . 9
  - 2.3.2 Deep Neural Networks . . . 10
  - 2.3.3 Data Reduction . . . 11
- 2.4 System Identification at the edge . . . 12
- 2.5 SHM systems for vehicle Calssification . . . 13

3 Background 14
- 3.1 Data Compression Models . . . 15
  - 3.1.1 System Identification . . . 15
  - 3.1.2 PCA . . . 16
  - 3.1.3 Autoencoders . . . 16
- 3.2 Clustering . . . 17
  - 3.2.1 K-means . . . 17
  - 3.2.2 Mean shift . . . 17
  - 3.2.3 Gaussian Mixture Models . . . 18

3.2.4 Mathematical Comparison of Methodologies 18

4 Tiny Detector Framework **19**
4.1 Hardware Standalone Design 19
4.1.1 Power Supply unit 20
4.1.2 Sensing Unit 22
4.1.3 Metrological characterization 24
4.1.4 Computational Unit 24
4.1.5 Transmission Unit 27
4.2 Final Framework 29
4.3 Experimental Results 29
4.3.1 Noise Analysis 30
4.3.2 Time Analysis 30
4.3.3 Frequency Analysis 31

5 Applied Signal Processing **33**
5.1 Data-driven Vs Model-Based Algorithm 34
5.1.1 SHM Installation 34
5.1.2 Methods: Anomaly Detection in an SHM framework 36
5.1.3 Deployment: Sensor Vs. Cloud 39
5.2 Results: Data-Driven Vs Model-Based 42
5.2.1 Algorithm Exploration 42
5.2.2 Hyperparameters exploration 45
5.2.3 Implementation 50
5.3 System Identification 53
5.3.1 Case Study 53
5.3.2 Methods: Sequential Vs. Parallel Tall Skinny QR 54
5.4 Results: System Identification Parametric Methods 57
5.4.1 Data Compression 57
5.4.2 Implementation 58
5.5 Vehicle Classification 60
5.5.1 Case Study 60
5.5.2 Methodology: Vehicle Classification in SHM context 63
5.6 Results: Vehicle Classifications 68
5.6.1 Feature Extraction & Data Labelling 69
5.6.2 Algorithm Exploration 71
5.6.3 Hyperparameter Exploration 72

6 Conclusion **75**

Bibliography **77**

LIST OF TABLES

2.1 Structural Health Monitoring studies over the last years. Performance results refer to the distinction of damaged from non-damaged data samples. Performance is in terms of Accuracy unless it is mentioned. Abbreviations: FP: False positives, FN: False Negative. 9

4.1 Sensor characterization values provided by manufacturer 23
4.2 Estimation of Noise in the time and frequency domain 30
4.3 In-time Analysis for the Lab and Beam experiments. 30
4.4 In-frequency Analysis for the Lab and Beam experiments. 31

5.1 Performance of our pipeline changing anomaly detection algorithm with discrete wavelet transform, frequency, and time data as input space domain. . . . 43
5.2 Comparison of our proposed solution with state-of-the-art methods applied to our dataset. The $60\,\mathrm{min}$ post-processing is identically applied to all methods. 48
5.3 Deployment metrics of PCA algorithm with CF = 16, output dimension = 60 minutes, and variable input dimension. Time and Energy are only for one inference. 50
5.4 NB-IoT deployment cost for the scenarios of our pipeline $E_{sleep} = 390$ (mJ) is the energy consumption of the node in PSM. $E_{acq} = 52.596$ (mJ) is the energy consumption to acquire 1 second of data 52
5.5 Performance indicators for varying N_p and $N_{s/1p}$ when parallelizing the AR and ARMA SysId models on the GAP9 platform. 59
5.6 Time intervals of each day for the dataset. 61
5.7 Frequency analysis results of 15 minutes in 4 different time zones of the day . 64
5.8 Comparison between k-means and mean-shift for considering all the spans together for the training set while sweeping over different micro-clusters . . . 73
5.9 Comparison between the best case Classification Accuracy for different scenarios 74

LIST OF FIGURES

1.1 IoT-based SHM systems. In Panel A, the raw signal is gathered from the sensors and sent to the cloud through a gateway to analyze the structure's condition. In Panel B, the safe/damage condition is directly computed on the node. 2
1.2 Hierarchical chain of this work's contributions. 3

4.1 GAP9 structures [103] 26
4.2 The final block diagram schematic of the SHM sensor-node. 28

5.1 Overview of the installed monitoring system on the viaduct. Five sensors are linked to a gateway for streaming data to the cloud. The grey box showcases the main components of a sensor node. 35
5.2 The proposed framework to analyze the condition of a viaduct starting from raw acceleration data. In the top part of the figure, we show the hyper-parameter tuning (in red) and the initial training steps done before the monitoring system was activated for the first time. In the middle, we show the inference steps to be done continuously for safe/anomaly condition assessment. In the bottom part, we show the possibility of updating the signal reconstruction algorithms after the pipeline detects an abnormal event to avoid the increase of false-positive alarms due to the bridge's static deformation caused by wind or aging. 36
5.3 Top panel: Twelve minutes of mean-centered raw acceleration data of the $z-$ axis of the middle sensor installed on a bridge span. Peaks are associated with vehicle passages. Left panel: Zoom of a 5-second window containing the oscillation associated with the passage of a vehicle. Right panel: The frequency response of the window of the signal is highlighted with the dashed rectangle. 37
5.4 PCA output mean square error (MSE) on the test dataset. The input window dimension is set to 5 seconds. The solid line is obtained by applying the post-processing with window dimension = $1\,\mathrm{h}$. 38
5.5 Three deployment scenarios of our anomaly detection pipeline. Green arrows highlight the inference steps, while red ones highlight the re-training and updating of the model over time. 40

5.6 ROC curve for different input signal domains, i.e., time, frequency, and time-frequency. 44
5.7 The energy filtering step impacts the PCA output MSE. In the top panel, we show the MSE when the energy filtering is not applied. In the bottom panel, we show the improved result with its application. 45
5.8 Effect of input-output dimensions sweep on the performance of the best detector (PCA). 46
5.9 MSE distribution while changing CF parameter 47
5.10 Performance of the PCA classifier while sweeping over several severities of anomalies w.r.t real case scenario of the bridge. 47
5.11 CF tuning versus accuracy, memory, and energy. Horizontal red line points to the limit of MCU memory. Dotted lines represent not deployable solutions. . 51
5.12 Two different implementations of the QR decomposition deployed to solve the System Identification model parameters. Panel A: The sequential implementation of QR decomposition at the edge processing each chunk of data sequentially. Panel B: The parallel implementation of QR decomposition at the edge processing multiple chunks of data simultaneously. 54
5.13 Workflow of the proposed P-TSQR-based SysId approach proposed in this application in which merely modal parameters are transmitted to the cloud replacing the raw data. 55
5.14 PSDs of the AR/ARMA models in a particular configuration (Np = 25 Ns1p = 35), comparison between GAP9 and Matlab 58
5.15 Time gain achieved by moving from sequential to parallel implementation. . . 60
5.16 The real-life case study used for vehicle classification application. Panel A: The weight-in-motion (WiM) system stores several metrics, such as weight and velocity. Panel B: Block diagram of the 5 spans of the viaduct under study. Panel C: SHM Framework for data acquisition installed under the viaduct. . . 61
5.17 The proposed framework of this work: (A) Two data acquisition systems of our system, namely, Accelerometer and Weight-In-Motion device. (B) Data pre-processing chain applied to 2D raw vibration to extract a 1D trace showcases the raw data. P1 to P4 are the L2-norm of the x-z plane, band pass Butterworth filter, overlap windowing, and energy smooth trace, respectively. Finally, the smooth traces are applied to PCA to extract two thresholds for vehicle identification. (C) Vehicle identification and feature extraction. (D) Labeling the extracted features by coinciding with WIM data. (E) Training and (F) Different validation studies. 63
5.18 The preprocessing chain applied to the raw data: A) Acceleration Raw data of z axis on top and x axis at the bottom, B) Result of the L_2 normalization of $x - z$ axes, C) Result of the 4^{th} order filter applied to the normalized values at the former step. D) Extracted energy values as the signature of each vehicle passage. 64
5.19 From raw data to the boxed version of each vehicle passage. Panel A) 15 minutes of raw data B) Boxed vehicles. 66
5.20 Distribution of clusters for a portion of SHM dataset showing 4 micro classes in light blue, dark blue, green, and red and thresholds to classify vehicles into three macro classes, i.e., Light, heavy, and super heavy. 68

5.21 Four different vehicle identicator metrics, namely Grossweight (panel A), Velocity (panel B), Momentum (panel C), and Kinetic energy(panel D) vs. the maximum amplitude of each vehicle event. In green the light class, in orange the heavy class, and in red super heavy class vehicles. 69
5.22 Gross weight distribution of EU - laws 70
5.23 Distribution of the extracted features over the gross weight. In green, the light class; in orange, the heavy class; and in red, super heavy class vehicles. . . . 70
5.24 Classification Accuracy for the two scenarios of the unsupervised classification of vehicles 71
5.25 Result of varying clusters of clustering methods. 72
5.26 Span Exploration Results in the two scenarios described in Sec. 5.5.1 73

CHAPTER

1

INTRODUCTION

Large-scale civil infrastructures have shown a worldwide expansion in recent years due to increasing investments of the top economies in the world [31, 42]. Other than civil structures, the safety of aerospace vehicles, industrial machines, and humongous electrical generators is another critical issue that took the government's attention in the last decades to tackle the environmental impacts affecting such complex systems [92, 93]. Remarkably, these infrastructures are evolving into increasingly sophisticated and complex systems. Indeed, advancements in technology and engineering have led to the appearance of long-span viaducts, extensive underground tunnels, and sprawling skyscraper districts in modern cities. Although these innovative complex structures enrich cities' landscapes, they present new challenges in design, construction, and, most importantly, ongoing maintenance [71]. Despite recent achievements in structural design ensuring the robustness of the structures, continuous and hazardous vulnerabilities persist. Threats such as extreme weather conditions (e.g., high winds, heavy rainfall), the impact of massive vehicular loads, and the ever-present risk of earthquakes remain significant concerns [54, 112]

For instance, evaluating the performance of bridges is essential in managing transportation infrastructure systems in EU countries, given the continuous increase in traffic loads and structural aging. Hence, optimizing one of the bottlenecks of transportation systems, i.e., viaducts, is a relevant study case for governments, where countries like Italy have more than $30\,\mathrm{K}$ long-span bridges and viaducts at the country level [28]. In particular, increasing traffic loads represent a key factor to consider when assessing existing structures, often standing out as the most substantial variable action impacting bridge performance. The recent tragic incident involving the *Polcevera viaduct's collapse* in Genoa, Italy, resulting in the loss of over 40 lives, serves as a heartbreaking reminder of the critical need for proactive structural monitoring and maintenance. This incident emphasizes the limitations of periodic or sporadic human-assisted assessments of structures. It has become increasingly evident that traditional approaches to structural health monitoring are insufficient to guarantee safety and infrastructure integrity. Consequently, a paradigm change towards continuously observing structural integrity and automatic anomaly detection is becoming a key requirement for civil infrastruc-

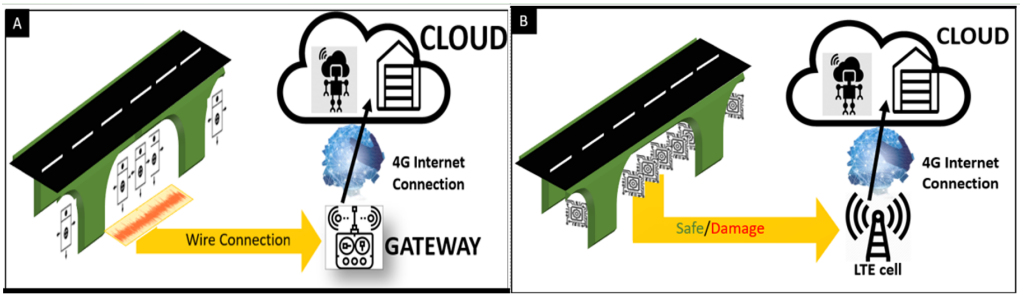


Figure 1.1: IoT-based SHM systems. In Panel A, the raw signal is gathered from the sensors and sent to the cloud through a gateway to analyze the structure's condition. In Panel B, the safe/damage condition is directly computed on the node.

ture maintenance [43, 73].

In the last two decades, the new field of Structural Health Monitoring (SHM) has emerged, exploring a wide range of techniques to continuously assess the conditions of structures, which are susceptible to various damaging phenomena [44]. To effectively assess the integrity of a structure, an SHM system includes data acquisition, data transmission, data processing, and data interpretation steps [84, 87]. Moreover, the new field of automated SHM, which tracks the real-time online state of structures using dense sensor networks, is gaining prominence to replace traditional SHM approaches [24, 95]. This idea, coupled with the advancements in the Internet of Things (IoT) [9], presents a transformative shift in monitoring large structures, offering cost-effective alternatives to traditional human crew deployments [50]. Each distributed sensor network over the target structure in the modern SHM systems embeds several low-cost sensors and a computational unit that can sense and transmit data to the centralized resources, i.e., the cloud, where information is stored and further processed to assess the level of structural integrity [32, 96]. A simple block diagram of such approaches is shown in Fig. 1.1-A. This multi-level architecture allows for efficient and comprehensive structural health monitoring in real-time, enabling timely interventions and maintenance strategies. Among the various monitoring techniques, vibration-based monitoring, in particular, is one of the most effective techniques for inspecting structures in the dynamic regime, i.e., structures that are completely characterized by frequency-related quantities [57, 93].

The amount of information and data gathered by new-generation SHM systems is exponentially growing, moving from a few measurements every hour from a few sensors to continuous high-frequency data streams from dense sensor networks [6, 40]. Indeed, this sensor-to-cloud continuous streaming generates a big volume of data, in the range of hundreds of MB to even GB per day for each individual structure with a dense SHM system monitoring the structure [72], leading to expensive and non-scalable solutions that barely adapt to large-scale scenarios [77]. Moreover, the continuous flow of long-time acquisition sessions requires large bandwidths and grid-based powering since the transmission phase is costly in energy consumption and can rapidly exhaust the capacity provided by battery-operated supplies [109]. Hence, network throughput and storage capabilities in the cloud have become major concerns in modern SHM systems, indicating the need to offload a portion of the computing workload to the processors near the sensors, i.e., the edge. Indeed, to overcome the challenges of streaming MB throughput and large volumes of data, embedded signal processing algorithms directly near the sensors could minimize the volume of data transmitted and enhance the energy efficiency of the system [68]. This introduces a new trend of intelligent SHM systems based on edge processing to avoid communicating raw data to the cloud. In this paradigm,

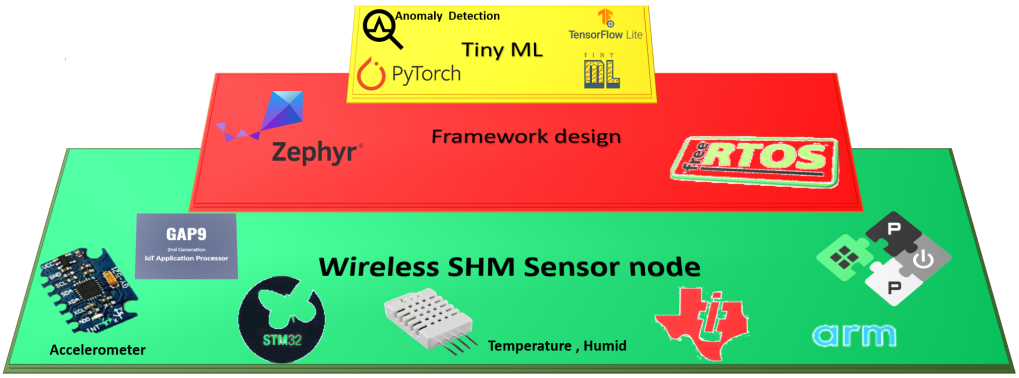


Figure 1.2: Hierarchical chain of this work's contributions.

raw data is processed near the sensors, reducing network traffic drastically from hundreds of MB to even less than MB per day between the edge sensor and the cloud [123]. Furthermore, reducing the network's throughput not only solves issues such as data security and storing an enlarged amount of data but also decreases energy consumption and cloud maintenance budgets [86]. However, implementing advanced signal processing, machine learning, and deep neural networks on resource-constrained computational units at the edge is not intuitive due to limitations such as limited memory sources and long inference time [22].

1.1 Main Contributions

This study addresses the challenges in deploying complex signal processing algorithms on resource-constrained devices at the edge to explore the potential transition from cloud to edge computing for Structural Health Monitoring applications. By doing so, these modern SHM systems provide promising approaches with more than ten years of battery lifetime and country-level scalability by avoiding streaming, processing, and storing raw sensor data in the cloud and instead transmitting reduced extracted features to the cloud.

An essential step in moving from cloud to edge is to use a sensor node equipped with wireless communication capable of performing real-time structural health monitoring. So far, the proposed SHM sensor nodes feature low-cost MEMS sensors and communicate data through the wire and wireless protocols. However, the computational unit of such nodes demands long latency to infer advanced signal processing. Hence, an SHM sensor node has been redesigned, utilizing two computational units, including a multi-core unit designed explicitly for inferring machine learning and advanced signal processing. Furthermore, different techniques have been proposed for anomaly detection over structures in literature, ranging from simple regressive models [39] to deep neural networks [115]. However, they are usually tested on simulated data, not taking into account real-condition perturbations such as wind or climate fluctuations [52, 104]. Besides that, these techniques are typically deployed on cloud servers. Hence, they imply comprehensive data collection from the sensor network. This work presents two novel frameworks for data compressing and anomaly detection from extreme edge sensors to overcome challenges such as long latency and energy consumption. Fig. 1.2 shows the hierarchical presentation of the main contributions of this work.

In particular, this work presents the following contributions to show the feasibility of a paradigm shift from cloud to edge by studying big-volume data sets collected over real-life study cases in Italy:

- Chap. 4 introduces an SHM sensor based on low-cost MEMS accelerometers featuring a single-core computational unit running on RTOS for data acquisition and communication combined with a multi-core processing unit merely for inferring advanced machine learning and deep neural network.
- Sec. 4.1.2 provides an investigation of the measurement accuracy of analog and digital MEMS configured in High-Performance and Low-Power mode compared with a seismic piezoelectrical accelerometer using both in-lab experiments and measurements taken on a real-world structure.
- Sec. 5.1, compare data-driven and model-driven unsupervised anomaly detection approaches to monitor the behavior of the viaduct, namely a Principal Component Analysis (PCA) model and two autoencoders. By deploying a real-life dataset, the PCA shows the best performance with 98.8% accuracy in detecting the structural changes after the interventions. Further, an assessment of the anomaly detection approach's robustness in depth is made by synthetically generating new anomalies from the original real-life one.
- Sec. 5.1 also provides the implementation of the anomaly detection pipeline on a low-power microcontroller for online inference with ≈ 74 uJ energy consumption for each inference. This study shows the trade-off between accuracy and power consumption by tuning the hyperparameters of our best-performing anomaly detector, the PCA. This section shows that by increasing the Compression Factor (CF) of the PCA (i.e., the ratio of the original space and the latent space) from 16 to 24, the framework still achieves 92.97% accuracy in detecting structural changes (i.e., distinguishing anomalies from normal samples) while consuming only 39 uJ per inference.
- Sec. 5.3 presents a parallel implementation of output-only system identification algorithm for data compression on a multi-core RISC-V MCU, GAP9, achieving a maximum worst-case execution time of $1.65\,\mathrm{s}$ @$110\,\mathrm{MHz}$ clock system, with an energy consumption of $80.1\,\mathrm{mJ}$.
- Sec. 5.5 presents a framework to classify vehicles into three classes with a classification accuracy of 96.87% in the best-case scenario using a real-case viaduct scenario with raw data acquired from the accelerometers and labeled data captured by Weight-In-Motion (WiM) from a viaduct in regular operation on a highway in northern Italy.
- Sec. 5.5 presents a comparison between the unsupervised Machine Learning models, namely K-means, mean shift, and Gaussian Mixture Model (GMM), showing that mean-shift outperforms k-means by an average of 3.91%, while it is more robust than GMM since the mean-shift has a standard deviation of 3.60% in classification accuracy for different sections of the bridge whereas it is 5.91% for GMM.

1.2 Manuscript organization

This work provides different sections of a paradigm shift from the cloud to the edge near the sensors by tackling different challenges of big data datasets on real-world case studies. To delve into it, in Chap. 2, we review the related works in the literature, providing the gap between the current work and previous works. Further, Chap. 3 provides a background review of the methods utilized for anomaly detection, data compression, and vehicle classification. Then, the Chap. 4 chapter describes the design flow of the SHM sensor node used in this

work. Chap. 5 provides the main algorithmic contribution of this work, where three different frameworks are discussed for data reduction and anomaly detection at the edge, followed by unsupervised classification of the vehicles deploying SHM sensor node vibration data. Finally, Chap. 6 concludes this work.

CHAPTER

2

LITERATURE REVIEW

This chapter provides a literature review of this manuscript's applications and research domain. The first two sections are hardware-oriented, which guided the design choices we made in our SHM ad hoc sensor node, while the last sections are more software-oriented, justifying the gap between the current and SToA work in the SHM domain.

In the first part of this chapter, we review the latest works focusing on characterizing low-cost MEMS-based sensors and the feasibility of replacing costly piezoelectric sensors with such sensors. Further, we provide the most recent commercial and custom SHM sensor node designs.

In the second part of this chapter, we focus more on the recent lightweight, embeddable solutions deployable at the edge, providing automated low-latency real-time responses for structural maintenance. At first, we study the key anomaly indicators suitable for resource-constrained devices at the edge to detect anomalous behavior of the structure. Further, we present a literature review of an unconventional data reduction technique at the edge, namely, system identification and its different implementations on computational-limited tiny devices. Finally, we tackle one of the most recent applications of the SHM system, i.e., Traffic Load Estimation (TLE), by reporting the challenges and solutions in recent works.

2.1 Accelerometers Technology

In recent years, MEMS technology has become important for several applications, such as bio-engineering [83], automation [67], and structural health monitoring [32]. Since their introduction, the reliability and performance of such sensors compared to the earlier, more expensive ones have challenged the community. Some works in the literature provided generic reviews comparing wireless MEMS-based accelerometer sensor boards for SHM [92] and Seismology [38]. Several evaluations characterized early analog MEMS performances with lab-based frames. For example, the work in [7] compared one analog MEMS-based with PCB accelerometers for modal analysis with three different excitations. Further, [3] delved more into analog MEMS by characterizing four different sensors in noise level, frequency, and sensitivity metrics. To avoid the relatively high noise level of the early MEMS accelerometer, the work in [59] designed two custom sensors for SHM applications with low bandwidth, thus resulting in a diminished noise level of MEMS.

More advanced analog and digital MEMS with deployability in embedded systems have been introduced to the community during the last decade, opening an ocean of options. To characterize these new MEMS, [90] evaluated two analog and four digital commercial MEMS sensors targeting frequency and damping identification for civil structures. By experimenting on a small concrete slab structure, they conclude that low-cost MEMS are feasible options to replace expensive piezoelectric ones. However, to conduct this conclusion, testing conditions differ for each sensor type, thus introducing heterogeneity in the dataset. Similarly, [97] computed displacement over a small-scale reinforced concrete (RC) beam to detect cracks exploiting four different accelerometer sensors. Although MEMS performed better to detect early cracks in the beam, PZT detected the final failure of the structure. Small RC structures characterized sensors better than steel-frame; nevertheless, a real-life scenario with a long, aged concrete highway where ambient noise plays a critical role is missing in the above-mentioned characterizations.

The work [16] provides the most reliable digital and analog sensors characterization by prototyping a self-made tri-axial accelerometer, i.e., Kionix KXR94-2050, and a referenced accelerometer PCB 356A16, to validate the applicability of MEMS practically over the cable-stayed bridge in Italy. Experimental Model Analysis (EMA) and Finite Element Analytical estimations (FEA) demonstrate that MEMS accelerometers can be a reliable substitute for expensive piezoelectric sensors. In a similar vein, we further investigate two scenarios (one real case and one laboratory) targeting a variety of low-cost commercial MEMS accelerometers.

Compared to other works, we initially characterize analog and digital MEMS vs piezoelectric sensors in real-life case experiments. Furthermore, benefiting from recent digital MEMS's high-performance and low-power features, we characterize these modes in both the time and frequency domains.

2.2 SHM frameworks

Developing embedded monitoring devices featuring IoT systems for civil structures has been actively investigated in recent years [2, 36, 88]. Two important features to achieve in modern SHM systems are the energy consumption and computational power of such systems, which guarantee long-lasting devices. To merge to an optimum solution for a real-time, zero-latency, (ultra) low-power SHM system, we can summarize two categories: i) Customized SHM nodes and ii) Commercial nodes.

The earliest wireless monitoring system was developed by [69], introducing computational power near the sensor. Furthermore, recent studies mainly focus on low-energy radio transmission and high computational power near the node to automate the structural monitoring at the edge. For instance, Polonelli et.al [85] designed an SHM monitoring node for tracking cracks in concrete and other construction material structures utilizing a LoRAWAN communication unit, making it suitable for long-lasting deployment as it guarantees more than $10\,\mathrm{years}$ of battery lifetime. However, the bandwidth and sample rate of crack metering are orders of magnitude lower than what is needed for vibration-based SHM. Focus on much higher bandwidth vibration-based analysis, which is as essential for SHM as static analysis since it provides complementary information [45, 125]. For instance, Muttillo et.al [75] designed an IoT sensor system for structural damage indicator evaluation following the criteria of low-power and computationally powerful systems. They propose a grid-powered IoT system that acquires synchronized MEMS-based accelerations that communicate to the gateway via the RS485 transceiver module. They show that the acquired data from two nodes can detect damages over an in-lab experimental setup.

Moreover, in [45], is another low-cost distributed system that is designed to execute a digital filtering step on a low-cost microcontroller STM32 to reduce the signal-to-noise ratio of MEMS devices, which was a new step in executing signal processing on resource-constrained devices. However, transmitting data via WiFi protocol. The two former cases are unsustainable grid-based systems consuming high energy.

The most recent battery-based SHM systems are proposed in [33], where a combination of cost-effective MEMS accelerometer and Narrowband IoT protocol (NB-IoT) to establish a long-distance connection with 4G infrastructure networks are the two key features makes this node suitable for long-term monitoring. Indeed, they show that a more than ten-year lifetime is achievable with a 17000 mAh battery or completely energy-neutral operation with a small solar panel. Similarly, STM32 has introduced a MEMS-based, multi-sensor, battery-based commercial node (STWIN) [98], specifically for condition monitoring and predictive maintenance applications. The STWIN core system board features several wired and wireless connectivity options with an ultra-low-power microcontroller based on the high-performance RISC core, operating at up to 120 MHz and equipped with 640 Kb SRAM and 2 MB Flash memory.

In our SHM customized sensor node, we follow the direction of the two aforementioned works in [33, 98], optimizing the computational sources and power management systems. In particular, we deploy a wide range of sensors proposed in [98] by means of embedded analog and digital accelerometer on-board while adding connectors to add other sensors to the board which is not present in [33]. Further, we follow the work in [33] to use NB-IoT as a transmission unit as they showed a ten-year battery lifetime. The main contribution of our node compared to the previous works in literature is adding a multi-core computational source on board, which is solely placed for performing ML/DNN algorithms at the edge.

	Structure	Sensors	Data Type	Detection Model	Train Device	Test Device	Performance
Statistical Data Modelling							
Ling et al. [62]	Simulated Steel Frame Structure	120	Acceleration	Autoregressive model	Remote Server	Remote Server [23]	1 FP, 1 FN
Santos et al. [35]	Simulated Five Bay Structure	4×29	Acceleration	FFT + Peak Detection	N.A.	MICAz [29]	0 FN, 2 FP
Verma et al. [105]	Simulated Steel Beam Bridge	2 14	Acceleration	Features + Gaussian Model	N.A.	N.A.	85-96% 96.3-100%
Deep Neural Networks							
Acvi et al. [11]	BM benchmark [37]	12	Acceleration	1D-CNN	Intel core-i7 [55]	Intel core-i7 [55]	N.A.
Data reduction							
Liu et al. [64]	Simulated Lab-Scale Bridge	5	Acceleration	Autoencoder	N.A.	N.A.	N.A.
Nie et al. [78]	Simulated Lab-Scale Bridge	9 24	Acceleration	FMPCA	Laptop	Remote Server	100% 100%
Our Work	Real Viaduct	1	Acceleration	AE / PCA	STM32L476	STM32L476	98.8%

Table 2.1: Structural Health Monitoring studies over the last years. Performance results refer to the distinction of damaged from non-damaged data samples. Performance is in terms of Accuracy unless it is mentioned. Abbreviations: FP: False positives, FN: False Negative.

2.3 Anomaly Detection at the edge

Structural Health Monitoring systems have become widespread in the last decade. They are usually based on sensor networks to monitor the structure's vibration under test [65]. An essential expected function of a modern SHM system is the automated detection of structural anomalies. To solve this problem, we can distinguish between three main classes of approaches: i) statistical data modeling, ii) machine learning, and iii) data reduction approaches. Table 2.1 summarizes the embedded SHM solutions deployable at the edge.

2.3.1 Statistical data modeling

The first approaches in continuous SHM systems were based on modeling data distribution and extracting abnormal patterns. Ling et al. [62] exploit auto-regressive (AR) and auto-regressive with extra input (ARX) models to detect anomalies on a simulated steel frame structure to localized damage pattern recognition problems in SHM. The authors compute a set of statistical features on a cluster of nodes where sensors communicate via Random Gossip protocol to detect and localize the damage, implying that an individual node cannot detect damages to the structure. Although they report at most 1 False Negative (FN) and 1 False Positive (FP) detection, they performed experiments with four laboratory computed datasets. Similarly, auto-regressive models are employed in, e.g., [47, 48, 117] to extract features from raw vibration data. One of the most recent works is Entezami et al. [39], which proposes an anomaly detection framework exploiting the recorded raw vibrations dataset of the Tianjin Yonghe cable-stayed bridge in China. First, an auto-regressive moving average (ARMA) extracts features to reduce data occupation. Then, a k-Nearest Neighbours algorithm classifies the samples, achieving as low as $1.56\,\%$ of misclassification. Despite the optimal results achieved, this work relies on a set of hand-tuned parameters, which impair the model's generality over time. To retrain these parameters, these models need the entire history of the data, which (i) is not always available and (ii) causes the system to necessitate a single cloud orchestrating unit.

The most recent study based on data modeling is [105], which takes advantage of real-case vibration data of a bridge in China and datasets from laboratory structures. They propose an

approach called "in-network damage detection on edge" to detect bridge structure damage. They collect statistical features of the input data into an *m*-dimensional feature vector. Then, they fit a Gaussian distribution model on the training set and consider anomalies as the tail of this distribution. Although the trained model's accuracy on the recorded Yonghe Bridge in China reaches $100\,\%$, it decreases to $96\,\%$ for the secondary simulated structure case. Furthermore, for all the experiments in this work, accuracy varies between $85\,\%$-$100\,\%$. This high fluctuation in performance is due to the reduced number of extracted features, which impairs the capability of this approach to model the structure's behavior in different positions with several sensors. While the lack of adjustability to the structure's behavior over time is a limit of their work, we propose a solution to update the model adaptively after a behavior change has been observed.

Santos et al. [35] is the only approach fully deployed on the edge. It computes the Fast Fourier Transformation (FFT) of input vibration data and the difference in peak frequency of each consecutive $0.5\,\mathrm{s}$ time window at the node. Then, computed natural frequencies are sent to sensor heads (i.e., Gateway) to estimate the structure's status using a threshold-based algorithm, which results in a perfect damage detection with only $2\,\mathrm{False}$ Positives (FP). Transmitting only natural frequencies causes network traffic of $6.72\,\mathrm{kB/h}$. In a similar vein, we further decrease network traffic to only $10\,\mathrm{B/h}$ by applying the Principal Component Analysis for data compression and classification directly on the node. Noteworthy, as demonstrated in Sec. 5.2.1, employing frequency features strongly impairs anomaly detection performance on our structure, making this approach unsuitable for our problem. Similar to Santos et al. [35], other works, e.g., [119, 122, 126], study the pros and cons of cloud computing and edge computing in the context of SHM systems.

To the best of our knowledge, all statistical data-modeling approaches exploited expensive piezoelectric accelerometers to collect data. In contrast, in our frameworks, we replace such sensors with low-cost, low-power (but higher noise) MEMS accelerometers.

2.3.2 Deep Neural Networks

In [1, 11], the authors present two DNN-based approaches. A 1D-CNN is used in [1] to estimate the Probability of Damage (PoD) on the BM benchmark [37]. A PoD close to $0\,\mathrm{points}$ to the normal case, whereas a PoD of $1\,\mathrm{corresponds}$ to the damaged condition. Evaluating nine scenarios of increasing damage severity shows that their 1D-CNN correctly ranks the scenarios from one to nine by correctly predicting an increasing damage condition. Compared to conventional 2D CNNs, 1D CNNs require less computational complexity and thus take less time to train the model. However, this model still requires data generated by a cluster of nodes, not a single node, to achieve high accuracy, which is unsuitable for online training on-edge nodes.

On a totally different input data, images, Wu et al. [111] present an approach for online inference. The authors exploit two deep convolutional neural networks, namely VGG16 and ResNet18, for crack and corrosion detection of structures from image data. They apply aggressive pruning to reduce the complexity while maintaining a high detection accuracy (they reduce VGG16 memory footprint to $44\,\mathrm{MB}$ and ResNet18 to $2\,\mathrm{MB}$). Running the algorithms on a Jetson TX2 platform, the authors achieve $94.6\,\%$-$98.5\,\%$ detection accuracy for crack detection on different nuclear power plant structures and $82.8\,\%$ detection accuracy in corrosion images of different metallic surfaces. Despite the performance, we do not employ deep supervised neural networks since they require a large training labeled dataset that is unavailable in our case of application provided in Sec. 5.1, and, more in general, labeled anomaly data is not available in typical structural health monitoring installations.

2.3.3 Data Reduction

The last category of works exploits compression algorithms for damage detection. These algorithms first compress and reconstruct the input data and then compute the difference between original and reconstructed signals. The higher the difference between the original and the reconstructed signals, the higher the probability of damage. In this context, autoencoder (AE) neural networks are among the most popular approaches. For example, [64] uses an AE for damage diagnosis on a laboratory's synthetic bridge for indirect bridge monitoring scenarios, outperforming all other anomaly detection algorithms with MSE ≈ 5 in computing 30 levels of damage severity. Given the promising performance of the method, we also test autoencoder-based anomaly detection in our work. Furthermore, linear processing-based compression methods such as principal component analysis (PCA) also achieve good performance in SHM for damage detection (e.g., [4, 12, 74, 78]). For example, [78] describes moving PCA on vibration data. They show the effectiveness of compression by evaluating the model over a laboratory beam bridge and recorded data of a bridge in Guangdong, China, with $100\,\%$ damage identification. Even though the works mentioned above can reach perfect accuracy, training, and inferring are performances on unconstrained remote devices (e.g., $i7intel$ processor) after data transmission and collection.

In our application, we aim to tackle these models' generalizability and deployability by introducing a new lightweight pipeline. Compared to other SHM works, we propose a method to constantly update the anomaly detector, tackling the time variability of the structure dynamic over time; also, our approach entirely relies on unsupervised data, not necessitating labels as other DNN-based approaches. Finally, to the best of our knowledge, we are the first to deploy and analyze the performance of a complete anomaly detection pipeline on an in-situ sensor network utilizing real-life SHM system installation on a viaduct.

2.4 System Identification at the edge

Implementing data reduction techniques at the sensor level is crucial to minimize the amount of data transmitted, resulting in shorter transmission times and negligible impact on sensor energy budgets [19]. Different reduction techniques for vibration data were successfully implemented onboard, spanning from Principal Component Analysis (PCA) to Compressed Sensing (CS). For example, in [19, 26], authors' embedded lightweight versions of PCA in microprocessors, reaching a compression level of $1.4\times$ and $15\times$, respectively. In [127], an adapted version of CS has been presented to handle vibration data showing good performances for reduction levels up to $6\times$. However, all these strategies present some drawbacks, such as the fact that they need training data for the optimal definition of compression parameters, along with the limited compression level they can allow for to ensure a sufficient quality in the retrieved structural properties [127].

Alternatively, System Identification (SysId) has been proposed in [123] as an unconventional but promising data compression method for vibration data processing, allowing up to $50\times$ dimension reduction, which is at least an order of magnitude higher than the ones mentioned above. SysId is a signal-processing technique that builds a mathematical model of a dynamic system based on a linear time-invariant filter, whose taps, also called model parameters (Θ), can reproduce the observed system dynamics. By knowing model parameters, the power spectrum can be computed, from which all the frequency-related properties of the structure (e.g., natural frequencies) can be extracted. The advantage of SysId is that just a dozen model parameters are sufficient to accurately replicate the observed system response, even for the most complicated geometries [25]. Thus, by transmitting Θ instead of raw data, which usually amounts to thousands of samples, one can reach very large compressing factors.

However, implementing SysId algorithms involves computationally and memory-intensive operations. Few attempts have been made in the literature to implement SysId in near-sensor scenarios. One of the very first implementations can be found in [60], in which authors succeeded in porting SysId filters on the Imote sensor platform; nonetheless, the limitations on storage constraints of this board forced the adoption of input-output correlation-based schemes, which are not compatible with the execution of output-only solutions as the ones of real interest for on-condition maintenance where the exciting stimulus is always non-measurable. These issues were overcome in a more recent work [123] by resorting to advanced algebraic procedures taken from the big data processing framework, which allowed the fitting output-only models on a prototype sensor equipped with an STM32L5 based on an ARM Cortex-M33 microcontroller unit (MCU). Nevertheless, the implementation offered in [123] suffers from one crucial limitation, which is the long execution time (greater than $120\,\mathrm{s}$ in the most cumbersome scenario) due to the sequential nature of the computational workflow forced by the single-core architecture of the computing processor. Eventually, this evidences the lack of on-sensor SysId implementations compatible with continuous and real-time diagnostic functionalities.

In this application, we offer a significant step forward in deploying SysId as an effective data compression technique for extreme edge sensors by tackling the challenges of long latency and high power consumption.

2.5 SHM systems for vehicle Calssification

Although the recent advancements in traffic estimation show promising practical models, there is still a need for accurate and real-time classification of vehicles in terms of dynamic weight estimation over bridges [49]. While using WiM measurements provides a precise assessment of the weight of vehicles passing over a bridge, WiM systems are expensive to install and maintain. Such systems are constrained to limitations concerning scalability and maintenance costs (e.g., [51]); hence, alternatives based on vibrations could provide more cost-effectiveness with the same accuracy as the WIM system. Various studies have shown excellent results using different sensors such as magnetic sensors [34], smart cameras [58], accelerometers, infrared, ultrasonic, and fiber optic acoustic sensors [79], and each of them requires specific algorithms correlated to the type of data. Moreover, the location of the deployed sensors and the traffic flow variability significantly impact vehicle identification performance. For example, using smart cameras, in [58], permits traditional object detectors and random forests algorithms to detect, classify, and count vehicles on public bridges (up to $92.1\,\%$ accuracy) in heavy traffic situations. Nevertheless, the light level of the bridge can often reduce the performance to $74.8\,\%$ remarkably. Magnetic sensors with adaptive thresholds and classification trees are used by [34, 108] and achieve $84.7\,\%$ to $99.9\,\%$ performance depending on the type of traffic flow. It is worth noticing that [34] shows the highest accuracy because of a specific, unrealistic deployment and test, i.e., slow speed by one vehicle all the time. Other works [63, 79, 114] use other types of sensors, such as fiber optic sensors and infrared, to achieve high performance but share similar issues or are too expensive for the deployment, e.g., single roadside-installed sensor fiber strips become unfeasible outside urban environments.

MEMS accelerometers can be used for Vehicle classification and are becoming an interesting solution because of the cheap infrastructure setup. These devices already used for SHM [118, 124] demonstrated to be very accurate and comparable to the other technologies [33, 46]. For example, [8] compared analog MEMS-based accelerometers with traditional SHM instrumentation for modal analysis with three different excitations. The quality of the measures in terms of noise level, frequency, and sensitivity metrics was comparable. [80] provides the most reliable analysis of digital and analog MEMS sensors, where several commercial MEMS devices were used as a reliable replacement for expensive piezoelectric sensors. Moreover, it examines a real case and a laboratory scenario to demonstrate the MEMS applicability for SHM on real bridges.

Pioneering work [18] shows how a four-step algorithm can turn a MEMS-based SHM installation into a Vehicles and Traffic Estimation system. However, [18] has not provided any evaluation of the vehicle classification, as they merely provide a student test for the two clusters. The most recent work for traffic load estimation [20] deploys a supervised ML algorithm (Support Vector Machine) reaching only an absolute error between the estimated label and the labeled data of $0.47\,\mathrm{for}$ light vehicles and $0.21\,\mathrm{for}$ heavy vehicles on a $60\,\mathrm{s}$ window. However, collecting labeled datasets at the country level is not feasible. Further, grouping vehicles into two groups is not sufficient enough to adequately estimate the dynamic load over the viaduct.

In this application, we offer a step forward in deploying vibration-based SHM systems for traffic load estimating by proposing a framework to classify vehicles into three classes: light, heavy, and super-heavy, utilizing an unsupervised ML clustering approach, i.e., mean shift. Further, we evaluate our framework by labeled data captured by Weight-In-Motion (WiM) from a viaduct in regular operation on a highway in northern Italy.

CHAPTER

3

BACKGROUND

This chapter briefly discusses the background knowledge of signal processing and machine learning algorithms used to develop pipelines and different applications.

It describes different data compression algorithms that can be used to reduce the volume of data sent to the cloud, leading to decreasing maintenance costs in structural systems. At last, three unsupervised clustering algorithms are described to develop vehicle classification pipelines in Sec. 5.5.

3.1 Data Compression Models

3.1.1 System Identification

System Identification (SysId) models are suite algorithms building a mathematical model for the observed signal in the form of a causal linear time-invariant filter whose frequency response function can be used for spectral analysis. They are built on top of AutoRegresive (AR) models, deploying regression techniques to solve the transfer function of the mentioned filter.

Let $x[k]$ and $y[k]$ be the input and output of the system at timestamp kT_s, where T_s is the sampling frequency of the system. Then, the most generic autoregressive equation at sample $k \in \{0, 1, 2, ..., N-1\}$ is demonstrated at Eq.3.1 in which q and p indicate the number of parameters retains memory of previous p input and q output instances. Thus, $N_p = p + q + 1$ is the total number of model coefficients to be resolved. Further, θ and γ are the feedback and feed-forward taps of the corresponding filter.

$$y[k] + \sum_{i=0}^{q} \theta y[k - iT_s] = \sum_{s=0}^{p} \gamma x[k - sT_s] \tag{3.1}$$

Notice that the mentioned taps (θ and γ) of the filter are called model parameters whose knowledge is sufficient to estimate the power spectral density of the input signal. As mentioned, they can be obtained by solving the autoregressive equations. In the field of structural analysis, only the output response of a structure is captured, without any information obtained from the input that caused such an excitation. This leads to the description of two output-only AR models for extracting filter taps, namely AutoRegressive (AR) and AutoRegressive Moving Average (ARMA), while other methods are not discussed in this manuscript. Since the input is unknown to solve Eq. 3.1, the input signal is estimated with a Gaussian term with zero mean and determined variance ($e[k]$ in Eq. 3.2, 3.3). Thus, the Eq. 3.1 can be derived to be written as Eq. 3.2 and Eq. 3.3 for AR and ARMA models, respectively.

$$y[k] + \sum_{i=0}^{q} \theta y[k - iT_s] = e[k] \tag{3.2}$$

$$y[k] + \sum_{i=0}^{q} \theta y[k - iT_s] = e[k] + \sum_{s=0}^{p} \gamma x[k - sT_s] \tag{3.3}$$

By means of algebraic manipulation of the former two equations, i.e., Eq. 3.2, 3.3, all the structural features of interests can be obtained either in time (filter impulse response function - IRF) or frequency (frequency response function - FRF) in the form of 3.4.

$$H_y(f) = \frac{\sum_{s=0}^{p} \gamma_s e^{-j2\pi f_s T_s}}{1 + \sum_{i=0}^{q} \theta_i e^{-j2\pi f_i T_s}} \tag{3.4}$$

Finally, the power spectral density of the system can be extracted as the square magnitude of the FRF response, which can be derived from Eq.3.4.

The length N is conveniently selected proportionally to N_p according to $N = N_p N_{s/1p}$, $N_{s/1p}$ being the number of time samples necessary to identify one single model parameter

accurately. Note that $N_{s/1p}$ shall be determined by the complexity of the dataset and the use case of the structure under study. The lower $N_{s/1p}$ is, the less accurate the estimated Power Spectral Density of the observed signal. Therefore, SysId aims at computing the N_p model parameters, a task that can be fulfilled by means of ordinary least-squares (OLS) applied to the linear regression form of Eq. (3.2) and (3.3), which generically reads as

$$Y = \Psi\Theta \tag{3.5}$$

with $Y \in \mathbb{R}^{N\times 1}$ and $\Psi \in \mathbb{R}^{N\times N_p}$ being the measured vibration response and the regression matrix, respectively[1].

3.1.2 PCA

Principal Component Analysis (PCA) is a method to deal with high dimensional correlated data by transforming them into minimally correlated data [30]. Exploiting the covariance matrix of high dimensional data, the PCA projects it into a new space where the axes correspond to the eigenvectors of the covariance matrix, ordered by the value of their eigenvalues. PCA reduces data size by preserving only directions that retain most of the information [120] (the ones with the associated higher eigenvalues). Considering a $M \times N$ dimensional data matrix $x = \begin{bmatrix} x_1, & x_2, & x_3, & \dots & , x_N \end{bmatrix}$ where x_k is a column vector of M features representing a sample, its normalized covariance matrix is

$$\Sigma = \frac{1}{N-1}\sum_{k=0}^{N}(x_k - \tilde{x})(x_k - \tilde{x})^T \tag{3.6}$$

where Σ is a square $M \times M$ matrix. Its diagonal holds the variance of each individual sample, and off-diagonal values are covariances of sample combinations. Using eigenvalue decomposition, we can write

$$\Sigma = V\Lambda V^{-1}. \tag{3.7}$$

where V columns represent the eigenvectors, and the principal diagonal of Λ contains corresponding eigenvalues. It can be proven that $\mathbf{V}^k \in \mathbb{R}^k$ is a basis of the sub-space of dimensions $\mathbb{R}^k$ which retains the highest similarity with the original one. Thus, the sub-space representation of the X can be extracted by Eq. 3.8 deploying the set of eigenvectors chosen for compressing the data.

$$Z(x) = V^\top X \tag{3.8}$$

Further, to reconstruct the original signal from the sub-space (Z(x)), one can use Eq. 3.9 to obtain the high dimensional signal [106].

$$X'(Z(x)) = Z(x)V^\top \tag{3.9}$$

3.1.3 Autoencoders

Autoencoders are neural networks composed of two or more layers used to compress data and detect anomalies [82]. Autoencoders can be segmented into two parts: Encoder and Decoder. The encoder, $f_E(x)$, projects the input data $x \in \mathbb{R}^M$ into a lower-dimensions hidden space

[1] For the definition of Ψ, see [123]

$h \in \mathbb{R}^k$, exploiting one or multiple layers, either fully connected, convolutional or recurrent [81]. An example of a single-layer encoder is

$$h = f_E(x) = \Phi(W_E x + b_E) \tag{3.10}$$

where W is the weight matrix, Φ is the activation function of a single layer, and x is the input vector to the network. The decoder $f_D(h)$ projects back the compressed signal h to its original space, creating a new signal $\bar{x} \in \mathbb{R}^M$ as

$$\bar{x} = f_D(h) = \Phi(W_D h + b_D). \tag{3.11}$$

The model's training favors the similarity of x and $\bar{x}$ without employing data labels, teaching the encoder to find the best-hidden space that mainly preserves the features of the original one. During training, the loss function is represented by a similarity metric between the original and the reconstructed signal.

The same metrics are also exploited to employ the autoencoder as an anomaly detector. Reconstructed signals, similar to those encountered during training, result in a low reconstruction error. On the other hand, reconstructing signals with different characteristics than those used for training are badly reconstructed. To detect anomalies, only normal signals are fed to the autoencoder for training. Therefore, new anomalous signals encountered during the test phase are poorly reconstructed, with a higher mean square error (MSE), and thus identified as anomalies.

3.2 Clustering

This section briefly introduces the three unsupervised algorithms deployed in this work: K-means, mean shift, and Gaussian Mixture Models (GMM). Consider the $M \times N$ data-matrix, as a dataset to be clustered into C classes, where M represents the number of samples and N represents the number of features (x_i) in each sample (X), hence $X = \begin{bmatrix} x_1, & x_2, & x_3, & \dots & , x_N \end{bmatrix}$.

3.2.1 K-means

K-means is one the most popular unsupervised "machine learning" (ML) models to classify non-labeled data [10, 66]. It aims to separate data into C clusters to optimize a clustering criterion: inertia or within-cluster sum-of-squares. The feature space is initially populated with C clusters, to which samples are then allocated. To determine the distance between each node and a cluster centroid (D), Eq. 3.12 is deployed. Each sample is then assigned to the closest centroid, where this centroid has the smallest Euclidean distance from it.

$$D = ||x_i - c_i||^2 \quad \forall c_i \in C \quad \text{and} \quad \forall x_i \in X \tag{3.12}$$

The process continues with the computation of a new cluster centroid. By calculating the arithmetic mean of all samples in the cluster, the new centroid is found. Finally, the K-means algorithm converges when the centroid movements become insignificant, or the maximum number of iterations is reached.

3.2.2 Mean shift

Mean shift is another unsupervised ML method that targets finding the dense area, i.e., modes of the data, in a discrete dataset [27]. Similar to k-means, mean-shift is an iterative non-parametric algorithm that converges when the cluster's center stops fluctuating. Let initial

point C with a radius r be the starting point of the algorithm. Further, at each iteration of the algorithm kernel function, $K(x_i - x)$ is applied to the point in the neighborhood of the given point x_i to weigh them for re-computing the mean. Finally, the weighted mean of the density in the window r can be computed with Eq .3.13.

$$\mathbf{m(x)} = \frac{\sum_{x_i \in N(x)} K(x_i - x) x_i}{\sum_{x_i \in N(x)} K(x_i - x)} \tag{3.13}$$

Where $N(x)$ is the neighborhood of the x. The difference $m(x) - x$ is called the mean-shift, and $m(x)$ is the new x in Eq .3.13.

3.2.3 Gaussian Mixture Models

According to Gaussian models, data can be represented as a combination of a finite number of Gaussian Distributions (GD), each with unknown covariance and mean parameters. The central objective is determining the optimal GD parameters for the given dataset [17]. Initially, the number of the GDs and initial parameters, namely mean and covariance, are randomly set. With the randomly chosen parameters, the algorithm extracts a probability that a sample belongs to a particular cluster. Intuitively, a point closer to the Gaussian centroid is likelier to belong to that cluster. Further, with the computed probabilities, the Gaussian Mixture Model defines a new set of parameters for each GD in the mixture to maximize the probabilities of the samples within the cluster. Note that the update of new parameters is performed via an expectation-maximization algorithm for the mixture of GD. In conclusion, the convergence of the GMM occurs in the absence of any modifications to its parameters [17].

3.2.4 Mathematical Comparison of Methodologies

The similarity in the preceding model, as discussed earlier, prompts a review of the distinctions among these models. The aforementioned mathematical models, i.e., K-means, mean shift, and GMMs, could be distinguished via centroids' updating approach at each algorithm iteration. To be exact, K-means and mean shift algorithms diverge in their approach to updating the central point of the centroid. Specifically, while K-means employs an arithmetic mean calculation using all data points within a cluster, mean shift computes a weighted average of these samples instead. Further, the mean shift differs from the GMMs since it is a non-parametric method. The dataset determines the more intricate specifications of mean shift, thereby resulting in a gradient that deviates from the Gaussian model's bell shape. Finally, one may argue that K-means and GMMs are similar; however, GMMs are more flexible due to the standard deviation. This allows the model's shape not to be constrained to the circle and take any other possible eclipse.

CHAPTER

4

TINY DETECTOR FRAMEWORK

4.1 Hardware Standalone Design

This section provides an in-depth analysis and design choices for the SHM node that we developed. Initially, this section provides an introduction to the power management unit, translating the input voltage to 6 different domains. Further, it provides an evaluation methodology to characterize MEMS-based sensors vs traditional accurate, costly piezoelectric accelerometers in the time and frequency domain. Next, the two microprocessors used in the board are described, in which STM32L4 single-core ARM-based one is used for acquisition and communication to the cloud while the multi-core RISCV-based one is merely used for the ML or neural network inference. Lastly, we describe wire and wireless communication units used to stream raw data or extracted features to the cloud.

4.1.1 Power Supply unit

SHM systems featuring accurate piezoelectric sensors are powered on a grid-based system, consuming a large amount of energy [127]. Lately, new MEMS-based SHM systems have merged, proposing battery-based systems that guarantee ten years of lifetime [33]. Further, sustainability is another major concern in the emerging SHM sensor nodes, which require an energy-harvesting system to recharge batteries for the in-field installation. This section presents different sections of our power supply unit, enabling both battery-based and power-grid-based solutions for the sensor node.

Voltage Regulators

This section describes the three different DC-DC voltage regulators used to convert the input voltage to $3.3\,\mathrm{V}$, by *RT8097AHGE*, a buck converter with only $22\,\mu\mathrm{A}$ quiescent current, $2.7\,\mathrm{V}$ with *LDK130M-R* which used in the [98] to powerup the analog parts of the MCU and ease audio signal read-out for the MCU, and finally a $1.8\,\mathrm{V}$, *ST1PS01EJR* another buck convertor with $500\,\mathrm{nA}$ to distinguish the GAP9 from the ST MCU power domains. Combining the former three sub-domains would provide analog and digital domains for the proposed SHM node. We selected the former devices to deploy based on two criteria: (i) low quiescent current to minimize the overall power consumption of the node and (ii) programmability of the mentioned converters to provide a flexible power management system in idle cases.

RT8097AHGE

The RT8097A is a $2\,\mathrm{A}$ synchronous step-down DC-DC converter designed for simplicity and efficiency. An input supply voltage within the range of $2.7\,\mathrm{V}$ to $6\,\mathrm{V}$ is provided, and it is equipped with an internal $0.6\,\mathrm{V}$ reference voltage to optimize performance. It operates at a nearly constant switching frequency across line, load, and output voltage conditions. Operating in automatic power saving mode, the converter maintains high efficiency during light load operation, an essential feature for systems with different subsystems, such as the load for this converter. Safety features include input under-voltage lockout, output under-voltage protection, and over-temperature protection (thermal shutdown), ensuring secure and smooth operation across diverse operating conditions.

LDK130M-R

The LDK130 voltage regulator is a low-dropout design offering a maximum current output of $300\,\mathrm{mA}$, operating within an input supply voltage range of $1.9\,\mathrm{V}$ to $5.5\,\mathrm{V}$—ceramic capacitors on the output guarantee the stabilization of this converter. It includes a very low drop voltage ($<100\,\mathrm{mV}$), minimal quiescent current ($30\,\mu\mathrm{A}$), making it particularly well-suited for applications requiring low power in battery-operated devices. An enable logic control function allows the LDK130 to enter shutdown mode, effectively reducing total current consumption to less than $1\,\mu\mathrm{A}$. Moreover, the device is equipped with short-circuit constant current limiting and thermal protection features, enhancing its overall reliability and safety.

ST1PS01EJR

The ST1PS01 is a compact synchronous step-down converter with nano-quiescent properties, capable of delivering an output current of up to $400\,\mathrm{mA}$ within an input voltage range of $1.8\,\mathrm{V}$ to $5.5\,\mathrm{V}$. Its main application is tailored for high-efficiency cases. Moreover, this converter

allows dynamic output voltage selection through two digital control inputs. The output voltage can be set at $0.62\,\mathrm{V}$ to $3.3\,\mathrm{V}$. Thanks to its enhanced peak current control, the ST1PS01 achieves remarkably high-efficiency conversion using only a $2.2\,\mu\mathrm{H}$ inductor and two small capacitors. The incorporation of advanced design circuitry minimizes quiescent current to nano amper. This converter also features an automatic power save mode that will be deployed with light loads. During the power saving mode, most of the internal blocks are turned off, reaching an ultra-low power consumption.

Charger Controller

The STNS01 is a linear charger designed for single-cell Li-Ion batteries and incorporates an LDO regulator and various battery protection functions. This device allows the programming of the fast-charge current through an external resistor. Precharge and termination currents are scaled accordingly, with a floating voltage value set at $4.2\,\mathrm{V}$. The input supply voltage acts as a dual purpose, charging the battery and providing power to the voltage regulator. Without a valid input voltage and a non-empty battery, the STNS01 automatically switches to battery power. Device protection circuitry is provided against potential damage during fault conditions. The STNS01 also integrates over-discharge and overcurrent protection circuitry. Additionally, it includes a charger enabling input, enabling the cessation of the charging process upon detection of battery over temperature by external circuitry. It enables shutdown mode, significantly reducing battery power consumption to less than $500\,\mathrm{nA}$, maximizing battery life.

System Design

The system can be powered by two approaches: via battery or a mini-USB of type B; thus, one can use the system in the battery-based or grid-based supply via USB. Further, the compatible battery charger with an integrated power switch can also be used for Li-Ion/Li-Polymer to recharge the battery while deploying an energy harvester.

To optimize the system's power consumption, the power domain is split into three main domains, including an analog part and two different digital domains. This branching enables each part of the system to act individually and switch to idle mode without interfering with other parts of the system. The LDK130M-R converter is deployed to translate the input voltage into a stable output $2.7\,\mathrm{V}$ for analog domains of the system. Notably, the always-on computational source of our design features from analog domains requires its specific power reference.

In the digital domain phase, ST1PS01EJR converts the input voltage to a stable $1.8\,\mathrm{V}$ with $400\,\mathrm{mA}$ output current for the multi-core processor power domain. Since the GAP9 platform is an ultra-low power design, it can be switched on with a maximum $1.8\,\mathrm{V}$ and consumes a maximum of $150\,\mathrm{mA}$ quiescent current in fully operation mode. Further, since the GAP9 is used only for advanced signal processing applications or ML and DNN inference models, this section of the node can be turned off without touching other parts of the system. The RT8097AHGE provides power to the digital domain, which provides $3.3\,\mathrm{V}$ in its output with $2\,\mathrm{A}$ output current. To further optimize the power consumption and also detach each part of the digital domain from one another, this portion is divided into four parts, namely MCU, NB-IoT, sensors, and Sd-Card domains, as shown in Fig. 4.2. The MCU is only connected to the STM32L496ZGT6P. Further, the NB-IOT is connected to the NB-IoT module and the CAN, which acts as the wired connection port. Moreover, sensors are powered on by the sensor section. Finally, the SD-card domain is used to power the SD-card logger part. In this way, one can also measure each part of the system's current consumption individually.

4.1.2 Sensing Unit

Resolution is one of the most essential performance requirements for SHM applications. So far, high-accurate, low-noise density piezoelectric-based sensors are widely recognized as the most accurate transducers for such operations [90]. In the last decades, MEMS (Micro-Electro-Mechanical Systems) capacitive accelerometers have also been introduced, and experiments have been conducted with such scenarios. Their meager cost and power permit the design and deployment of a steady measurement infrastructure for continuous monitoring to scale up to hundreds of measurement points for a single building, which is unfeasible using piezo accelerometers, two orders of magnitude more expensive. Nevertheless, MEMS accelerometers do not outperform piezo accelerometers because commercial devices are designed to measure larger bandwidth signals and are characterized by lower sensitivity, thus requiring complex signal conditioning electronics to achieve acceptable noise floor [16]. For this reason, to unveil the potential of these devices, a comparison with piezo accelerometers and a characterization of representative MEMS devices, currently missing in the literature, is needed. This section presents the characterization in both time and frequency domains of representative MEMS devices, focusing on SHM application-specific metrics and using either a laboratory set-up or a real infrastructure.

Sensors

This section describes the three different sensor technologies chosen for the comparisons to choose the best ones for our sensor node, including a highly accurate expensive piezoelectric as the ground-truth sensor, PCB393B12, an ultra-compact linear low-cost analog MEMS, namely LIS344ALH, and a dual-mode always-on 3D digital MEMS, namely ISM330DHCX. Other than SHM, innovative infrastructure and inertial navigation are fields of interest for such sensors. We selected the devices to test based on two criteria: (i) comparing elder analog MEMS devices with more recent digital ones and (ii) selecting a digital device that works in both Low-Power and High-Performance modes. Table 4.1 provides each sensor's nominal and mechanical characterization.

PCB393B12

The PCB393B12 is a uni-axial Integrated Circuit Piezoelectric (ICP) accelerometer sensor benefiting from a low-cost coaxial cable connector to interface with the data logger. This sensor operates by applying a constant current signal. The ICP technology converts the high-impedance acquired data to a low-impedance output signal capable of unconditionally transmitting lines with long cables. Furthermore, the low-noise output voltage is compatible with data analysis methodologies. Fields like smart infrastructure, earthquake detection, and structural monitoring are the main applications of such sensors.

LIS344ALH

The LIS344ALH is an ultra-compact three-axis linear accelerometer, including a sensing element and an IC interface system. The sensing element is fabricated by the STMicroelectronics (ST) production line for sensors and actuators in silicon, and it is adept at detecting accelerations. Similarly, the IC interface is manufactured, deploying the CMOS process with a high level of integration developed by ST. The primary task of the IC interface is to convert the information acquired by the sensing element into an analog signal for the external world. Other

Sensor	Range $[\pm g]$	Noise $[\mu g/\sqrt{Hz}]$	Voltage [V]	Signal Type	Signal Interface	Temp. $[^{\circ}C]$	ODR [Hz]
PCB	0.5	0.3 – 1.3	18.0 – 30.0	Analog	ADC	-50 – 180	0.15 – 1000
LIS	2,6	50	2.4 – 3.6	Analog	ADC	-40 – 85	1.00 – 500
ISM	2,4,8,16	60.0 – 100.0	1.7 – 3.6	Digital	I2C,SPI	-40 – 105	1.60 – 6667

Table 4.1: Sensor characterization values provided by manufacturer

than condition monitoring, its other applications are gaming, robotics, and inertial navigation.

ISM330DHCX

The ISM330DHCX is a system-in-package including a high-performance 3D digital accelerometer and 3D digital gyroscope tailored for Industry 4.0 applications. The manufacturing process for various sensing elements and IC interfaces is similar to the one described in Sec. 4.1.2. Since it is a digital system, it introduces adjustability to the system. For instance, a set of programable computational features such as a Machine Learning (ML) core, an accessible and programmable Finite State Machine (FSM), and $9\,\mathrm{kB}$ FIFO to store data temporarily and perform real-time analysis provide the user with an intelligent sensor at low power. Furthermore, this accelerometer benefits from two modes, namely, high-performance and low-power, where these modes can be used to reduce the system's total energy consumption.

Experiment Description

In-Lab

This experiment utilized a Material Test System (MTS) shaker to excite vertically the sensors described in Sec. 4.1.2 with sinusoidal stimuli. It was carried out to investigate the performance of measurement systems benefiting from commercial MEMS and piezoelectric accelerometers. Since structures operate under low frequency, we fixed the excitation frequency of the shaker at $10\,\mathrm{Hz}$ while sweeping the amplitude range of the excitation from $30\,\mathrm{to}\,250\,\mu\mathrm{m}$. The intuition behind this experiment was to simulate various ranges of input excitation to estimate real-life random value input excitations in long-span bridges or structures. A plate is attached to the shaker, holding the mounted sensors. Screws fix the MEMS sensors, and the piezo one is attached to the bottom of the plate by a steel magnet connector.

Concrete Beam

To assess the performance of devices under test in a real-world scenario, we carried out a set of measurements on a concrete beam. The beam comprised a concrete slab supported by two steel towers at each end. The total length of the beam is $25.9\,\mathrm{m}$, while the width is $1.6\,\mathrm{m}$. The experiment started by charging the beam with an even number of plates ($1 \times 1 \times 0.25\,\mathrm{m}$), each weighing $1800\,\mathrm{kg}$. After charging four plates over the beam, additional wedges were added to the towers holding the beam to avoid rigid torsion rotation. The experiment aimed to charge and discharge the beam until cracks appeared and ended after charging the beam with twelve plates on top of it.

4.1.3 Metrological characterization

The primary task of structural health monitoring is to collect building health information, unveiling potentially harmful issues such as damage and aging. In dynamic monitoring applications, a set of sensors collect accelerometric data to feed modal identification algorithms and perform early damage detection and structural health assessment analysis. We compare accelerometer performance in the frequency and time domain by analyzing the measurements taken during the two experiments described in Sec. 4.1.2.

We first set the relevant metrics for damage detection to evaluate measurement quality in the time domain. Structural vibrations can be modeled as dampened oscillations depending on three parameters, namely frequency, amplitude, and damping factor. As such, by fitting the sensor measurements with this model using the Ordinary Least Squares method, we evaluate the accuracy of the selected accelerometers in inferring these structural parameters. The model for the In-Lab experiments, described in Eq. 4.1, is parametric with respect to signal amplitude (c_0), frequency (c_1) and phase (c_2).

$$f_{lab}(t) = c_0 sin(c_1 t + c_2) \tag{4.1}$$

Eq. 4.2 approximates the dynamic behavior of the beam as a single degree of freedom spring-mass-damper system. For this purpose, a further coefficient (c_3) is introduced to model the decay factor of structural oscillations in time.

$$f_{beam}(t) = c_0 e^{-c_3 t} sin(c_1 t + c_2) \tag{4.2}$$

The frequency-domain analysis focuses on assessing the quality of the power spectral density (PSD) that can be obtained when analyzing the measurements of the different devices. Describing the accuracy of the PSD of a signal is of paramount importance since most frequency-domain modal identification and damage detection algorithms are built on top of PSD computation. For each experiment, we estimate the PSD of the measured signal using the Welch method, choose the most prominent peak and compute three metrics: (i) natural frequency, (ii) amplitude, and (iii) width. Peak frequency is an important parameter to assess structural health, and several state-of-the-art damage detection pipelines observe shifts in the natural frequencies of a structure to detect anomalies [15]. Peak amplitude and width are also relevant since they play a role in estimating modal shape and structural damping, two modal parameters that can be observed to detect changes in structural dynamic behavior. According to the Half-Power method, peak width is estimated as the distance between the peak intercepts at amplitude $p/\sqrt{2}$, where p is the peak height. For the in-lab experiment, the most prominent peak corresponds to the tone in the spectrum corresponding to the frequency of the sinusoidal input stimulus applied by the MTS machine. Instead, for beam measurements, the most prominent peak represents the first modal frequency of the structure. To complete frequency domain analysis, we estimate device noise from a "silent" portion of the real-life experiments where no excitation was applied to the sensors. We estimate noise in the frequency domain, computing the square root of the average of the noise PSD, and in the time domain, computing the root mean square (RMS) of the measured noise signal.

4.1.4 Computational Unit

Resource constraint devices near the sensors are not merely devoted to reading and transmitting data to the storage facilities. Technological advancements have led to advanced systems capable of performing signal processing, as well as lightweight ML and DNN networks. The current SHM sensor nodes deploy single-core devices as the main processing unit. These

devices at the edge are in charge of processing lightweight and low-latency algorithms. Moreover, such devices provide Real-Time Operating Systems (RTOS) capabilities to parallelize several tasks in the system. However, memory limitations and energy consumption at full operation mode are two key challenges for such devices. More powerful devices in terms of memory and execution time have been promoted in the last few years, supporting not only hierarchical memory up to the range of MB but also featuring a cluster of cores supporting parallel computing to replace sequential implementations deployed on single-core devices.

In this vein, we used a single-core device at the main core of the SHM node, which acts as a three-gate port, one for sensor readout, data handling, and data transmission. Further, the multi-core MCU is deployed to translate the raw data into knowledge via time and frequency domain analysis. Furthermore, this multi-core device is designed and optimized for inferring lightweight ML and DNN networks, making it a perfect match for data-driven applications on the edge.

STM32L

The STM32 MCU processor family covers a wide range of single-core microcontrollers targeting three different criteria, i.e., high-performance with real-time constraints providing digital signal processing capabilities (*F & H Series*), ultra-low power consumption (*L Series*), and wireless connectivity for IoT use-cases (*W Series*). STM32 MCUs are Arm®Cortex®-M-based devices with a 32-bit instruction set architecture (ISA) [116]. The nominal operation frequency of these devices varies from a few MHz to a maximum of $400\,\mathrm{MHz}$ depending on the series. They are mostly single-core devices featuring Real-Time-Operating free Systems (RTOS) to handle several tasks in near real-time. Next, they include a wide range of peripherals, making them suitable for embedded systems, enabling acquisition and processing at the edge. The STM32 memory is limited to $1.4\,\mathrm{MB}$ of Static SRAM (SRAM) as the primary working memory to store variables and stack data. This technology does not require to be periodically refreshed to maintain its contents, unlike dynamic RAM (DRAM), and is faster than Flash memory. In particular, *H series* devices' SRAM is in the range of $564\,\mathrm{kB}$ to $1.4\,\mathrm{MB}$, while *F series* covers less RAM with $32\,\mathrm{kB}$ to maximum $256\,\mathrm{kB}$. Further, the Flash memory provided in each series varies between 1 to $2\,\mathrm{MB}$, used to store code and make data persistent when the device is turned off. For the context of this work, we have used *L & F Series* to develop the monitoring firmware of the monitoring devices.

Gap9

The recent resource-constrained processors deployed at the extreme edge are single- or multi-core devices that GAP9 MCU [103] is an ultra-low-power multi-core microprocessor targeted for IoT applications at the extreme edge. Most of these devices are based on either 32-bit or 64-bit instruction sets for arithmetic operations, in which GAP9 fits in the 32-bit category for floating point and up to 64-bit for fixed point operation. Compared to the other processors, the Floating Point Unit (FPU) of GAP9 also supports 16-bit and 16-bit alt operations along with 32-bits. GAP9 features a single-core MCU-class core (fabric controller) orchestrating system-level operations (e.g., system boot and I/O connectivity) and a 9-core compute cluster to support parallel execution. All the cores adhere to the RISC-V standard and support the PULP ISA extensions [91]. The nominal frequency of the cluster can be increased up to $370\,\mathrm{MHz}$ while keeping the power consumption in the nominal operating mode below $50\,\mathrm{mW}$. Conversely, other multi-core RISCV-based MCUs or dual-core ARM Cortex devices provide fewer cores with the same or smaller clock frequency. For instance, GAP8 is a multi-core

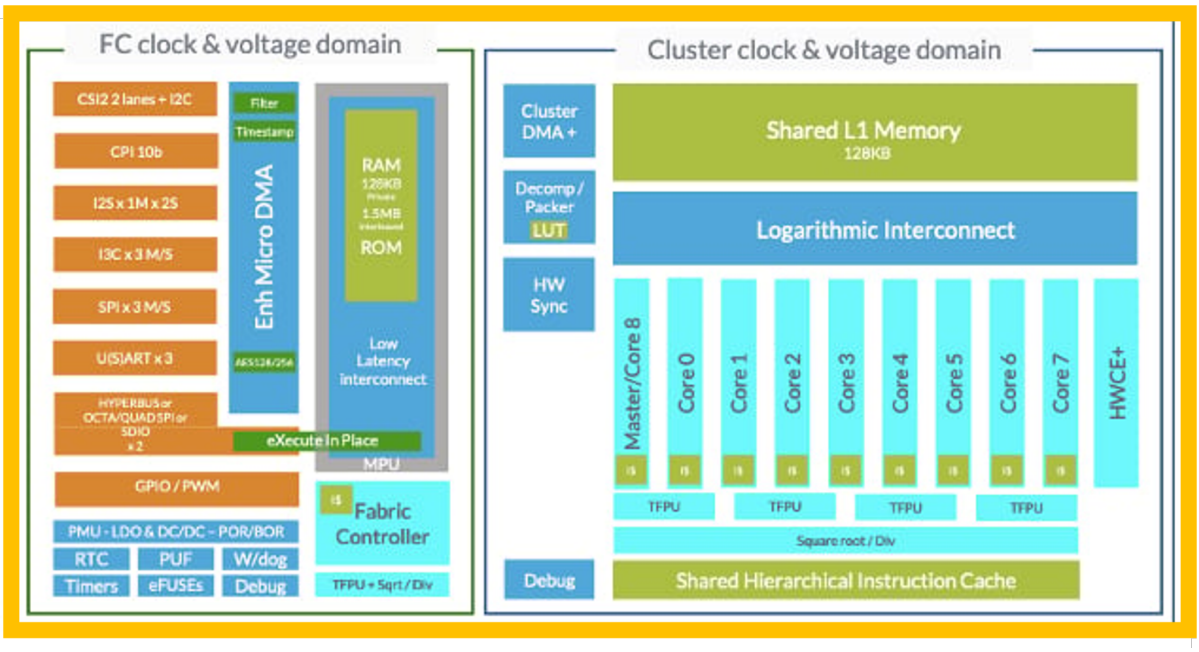


Figure 4.1: GAP9 structures [103]

parallel platform, an older version of the save RISC-V ISA family, with 9 cores, one fabric controller, and 8 cluster cores. The lack of FPU in GAP8 and the maximum frequency of $250\,\mathrm{MHz}$ of its fabric controller and $175\,\mathrm{MHz}$ for cluster cores are the main differences from GAP9. Another example is the ESP32 MCU series, which are dual-core processors with CPU clock frequency in the range of 80 to $240\,\mathrm{MHz}$ with 32-bit RISC architecture. Other multi-core devices based on Cortex Arm architecture are dual-core systems with symmetric (same) or asymmetric (different) processing cores. For instance, $STM32H745ZIT6$ is a dual-core microcontroller combining an Arm Cortex-M7 core running at $480\,\mathrm{MHz}$ and a second Arm Cortex-M4 processor running at $240\,\mathrm{MHz}$. However, less current consumption and a number of cores make GAP9 a suitable MCU for parallelizable applications. GAP9 speeds up the execution of Digital Signal Processing (DSP) algorithms thanks to dedicated ISA extensions such as post-incremented Load (LD) and Store (ST), hardware loops, and packed Single Instruction Multiple Data (SIMD) instructions [103] and 4 dedicated floating-point units (FPUs) shared among the cores supporting 16-bit and 32-bit precision operations.

Fig. 4.1 depicts a simplified block diagram of the GAP9 architecture. It has a hierarchical memory architecture with $128\,\mathrm{kB}$ of single clock latency tightly coupled data memory, namely, L_1 , as well as L_2 SRAM with $1.6\,\mathrm{MB}$ and an L_2 non-volatile memory of $2\,\mathrm{MB}$. Moreover, L_1 interconnect minimizes access contentions to the SRAM banks via a word-level interleaving scheme to evenly distribute the requests. Similar to memory, GAP9 also has a hierarchical program cache featuring 8 512-B private per core. Next, a joint combination of a parallel code and the $4\,\mathrm{kB}$ shared cache with two-cycle latency maximizes efficiency. Finally, the *event unit* embedded in GAP9 is a hardware unit accelerating the fine-grained parallel thread dispatching and synchronization, critical tasks for many applications exploiting single program multiple data parallelization schemes. Further, this unit is in charge of clock gating of the idle cores waiting for synchronization and enables resuming execution in two cycles [91].

4.1.5 Transmission Unit

Transmitting data is another major challenge in SHM sensor nodes. Most nodes that deploy piezoelectric accelerometers as their sensor unit send data to a local storage unit, e.g., Personal Computer (PC), via wire connections. Further, with the emergence of Internet of Things (IoT) technology, Wireless Sensor Networks (WSN) were promoted to deploy a chain network from sensor nodes to the cloud. In such scenarios, the data collected by the sensors is transmitted to the Gateway through wire-based protocols (CAN) or wireless (Bluetooth). Further, Gateway transmits data to the cloud using more stable, long-distance, free bandwidth, and power-hungry protocols such as WiFi. Most recent SHM sensor nodes like [33] deploy NB-IoT, a low-power extension of the LTE (4G Long Term Evolution) developed for long-battery lifetime and low-cost applications. The GW is avoided in such designs since the SHM node can directly communicate with the cloud via the sim card and NB-IoT module. In our design, we examined two widely utilized transmission methods in Sensor Networks for the Internet of Things, which are also addressed in the literature:

- Wire-based Transmission (CAN): This method involves a wired connection, explicitly utilizing the Controller Area Network (CAN). CAN facilitates the close connection of multiple nodes, enabling seamless communication. This configuration is particularly advantageous when deploying sensor nodes in a typical Wireless Sensor Network (WSN) environment.

- Wireless Transmission (NB-IoT): The second approach involves wireless communication using the Narrowband Internet of Things (NB-IoT). This wireless unit allows the sensor node to operate as a stand-alone unit, providing the capability for both static and dynamic analysis. NB-IoT's wireless nature enhances flexibility and independence, making it suitable for scenarios where a wired connection may be impractical or restrictive.

By considering both wire-based (CAN) and wireless (NB-IoT) transmissions, our work aims to explore and leverage the strengths of each method to enhance the overall capabilities and adaptability of sensor nodes for data transmission.

CAN

The Controller Area Network (CAN) protocol is a widely used communication protocol to collect data in a short distance. Despite its initial development for automotive applications, it has been utilized in various other industries. CAN benefit from a multi-master bus, allowing any node on the network to act as a sender or receiver. Further, its asynchronous operation calls for no master-slave connection between nodes. Moreover, the nodes on the network can send messages independently when they have data to transmit. Such communication is event-driven, with nodes responding to messages as they occur. CAN robustly detect and handle transmission errors, reporting them back to the node so that the affected message can be re-transmitted. Another major key that suits CAN as a communication unit for SHM applications is high data rate support from $125\,\mathrm{kbps}$ to $1\,\mathrm{Mbps}$, ensuring a real-time transition.

NB-IoT

NB-IoT, short for Narrowband Internet of Things, is a low-power extension of LTE (4G Long Term Evolution), specifically designed for applications requiring a long battery lifetime and

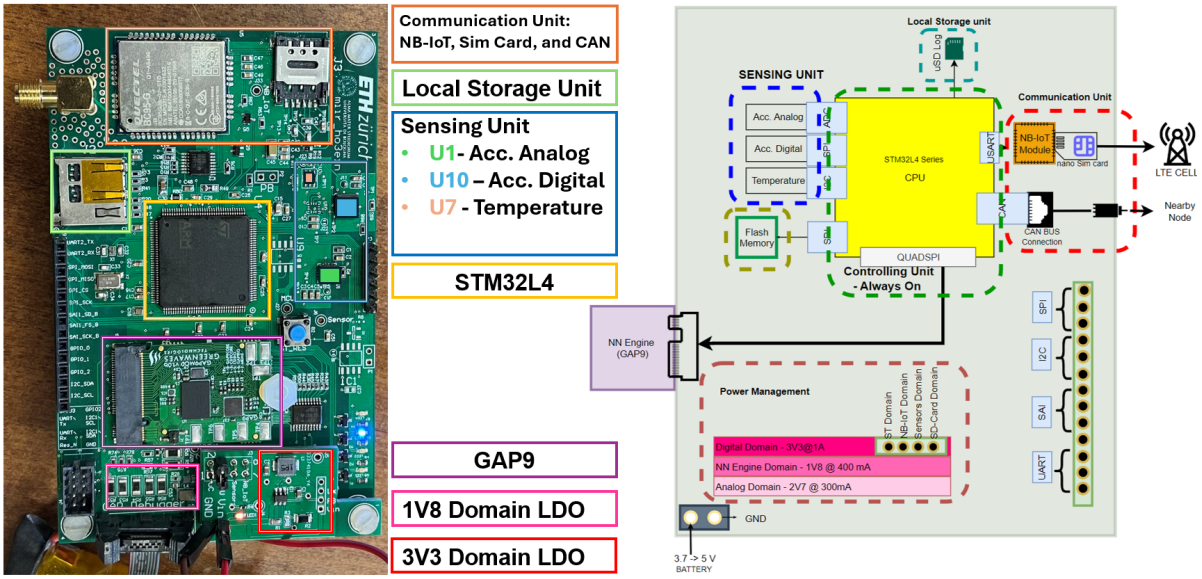


Figure 4.2: The final block diagram schematic of the SHM sensor-node.

cost-effectiveness. The primary features of NB-IoT include low power consumption, long coverage, and backward compatibility [110]. One of the significant advantages of NB-IoT is its ability to operate with low power consumption, ensuring battery-based systems' long lifetime. Additionally, it offers extended coverage, making it suitable for applications in diverse environments, e.g., Agriculture and Structure monitoring. The standardization of NB-IoT by 3GPP for Low Power Wide Area Networks (LPWANs) enhances its widespread adoption. The power consumption of NB-IoT devices can vary based on environmental factors such as country-specific regulations and network operator settings. These variations can significantly impact the overall performance of end devices. The adaptability of NB-IoT makes it a compelling choice for a wide range of applications, contributing to the growth of the Internet of Things ecosystem.

A comparison between different wireless connectivity is introduced in the work presented by Polonelli et al. [14]. Further, in [33], they investigated wireless connectivity methods, providing an exploration of the most suitable NB-IoT module and configuration. Further, this work [33] characterizes the energy consumption of two different NB-IoT modules in good ($-95dBm < RSSI$[1]), medium ($-110dBm < RSSI < -95dBm$), and bad ($RSSI < -110dBm$) coverages, concluding that *BC95-G* is a preferable node for SHM systems, ensuring a long lifetime. Therefore, we use it for our design as the NB-IoT module.

The BC95-G is a high-performance NB-IoT module that supports multiple frequency bands, all while maintaining low power consumption. It is designed to be compatible with the Quectel GSM/GPRS, which provides a flexible and scalable platform. This feature allows for a smooth migration from GSM/GPRS to NB-IoT networks, enhancing adaptability. Utilizing surface-mounted technology, the BC95-G ensures durability and ruggedness, making it suitable for challenging environments. Its compact form factor, ultra-low power consumption, and extended temperature range make it an optimal choice for IoT applications. The main applications of BC95-G deployment include smart metering, bike sharing, smart parking, smart city infrastructure, security and asset tracking, home appliances, and agricultural and environmental monitoring. The module can provide a comprehensive range of data transmission services to fulfill various client-side demands.

[1](Received Signal Strength Indicator)

4.2 Final Framework

Fig. 4.2 presents the final design of the proposed SHM sensor node. The final design includes a sensing unit, a communication unit, two computational units, and a power management replicating a standard SHM node. In this section, we finalize the hardware design by exploring different configurations of the designed node. In the power management unit, the input accepts two different power modes, one with the battery, as shown, and one with a micro USB. The node can be battery- and grid-based as the DC-DC converters accept up to $5\,\mathrm{V}$. Further, the digital domain is split into four different sections with different headers, whereas each header can be used to detach a specific unit from the system's digital domain. Further, these headers can be used to measure the whole system or a specific part.

In the sensing unit, we included a low-noise, analog MEMS accelerometer reaching an acceptable acquisition accuracy as shown in Sec. 4.3. Moreover, we also considered the IMU accelerometer to enable processing not only at the MCU level but also at the sensor level. The deployability of these sensors is a user-defined configuration, as one can record acceleration using both or either. The footprint used for the IMU is standard; hence, one can use all the commercial IMUs from STM32 company. This allows us to quickly replace a new version of IMUs with more computational power than the older ones. A humidity and temperature sensor is also embedded in the system to monitor how weather can affect the structure under study.

The acquired data by sensors can either be stored locally on the SD Card, which accepts maximum $32\,\mathrm{MB}$ of data, or sent to the communication unit to be transmitted to the cloud. The communication unit benefits from a wireless NB-IoT module that can transmit data directly to the cloud, avoiding the gateway. Moreover, a couple of sensors can be connected by wire via a CAN connection. Since using NB-IoT continuously to transmit data is energy-costly, we added the CAN connection, allowing sensor-fog-cloud paradigms.

The computational unit has two main components: a single-core $STM32L4R5ZI$ and a multi-core PULP-based $GAP9$. The single-core MCU is the always-on unit, whereas the GAP9 is activated only when an inference is required. When the single-core MCU collects an informative data window, an interrupt will trigger on $GAP9$ to switch to the total operating mode. Afterward, $GAP9$ replies with an acknowledgment pin to receive and start processing the window. At the end of processing, the results are sent back to the single-core device to be transmitted to the cloud. The single-core MCU handles the system by running an RTOS with five tasks: two for acquisition, one for data processing, and two for data transmission to the cloud. The acquisition task handles data from the two sensors with a ping-pong topology, benefiting the best use of the Direct Memory Access (DMA) unit embedded in the single-core device. Furthermore, a low-pass filter is applied to the acquired signals to reduce the noise level of the acquired data. The filtered data are forwarded to the processing task to be processed by the GAP9 processor or stored in the system's SD CARD. Finally, the communication tasks are in charge of reading and transmitting the stored data in the SD CARD, which are either raw filtered data or extracted features, which is the outcome of the processing unit. The transmission task is highly reliable against packet loss as it re-transmits the packet in case of package loss or broken link.

4.3 Experimental Results

In this section, our main focus is on analyzing the evaluation of the two MEMS used in the SHM node with the accurate piezoelectric accelerometer employing the three domains proposed in Sec. 4.1.3. Utilizing a lab-scale shaker, we first evaluate the performance of each

	PCB	LIS	$\mathrm{ISM_H}$	$\mathrm{ISM_L}$
$\mu g/\sqrt{Hz}$	10.28	23.98	65.86	436.20
mg_{RMS}	0.06	0.12	0.35	2.19

Table 4.2: Estimation of Noise in the time and frequency domain

Experiment		Signal Frequency [Hz]				Signal Amplitude [mg]			
		PCB	LIS	$\mathrm{ISM_H}$	$\mathrm{ISM_L}$	PCB	LIS	$\mathrm{ISM_H}$	$\mathrm{ISM_L}$
LAB	$30\,\mu m$	10.0	10.0	10.0	9.9	10	9	10	24
	$250\,\mu m$	10.0	10.0	10.0	9.9	91	86	100	120
Experiment		Signal Frequency [Hz]				Decay Factor [1e-3]			
BEAM		5.50	5.54	5.59	5.57	8.3	8.6	8.3	11.6

Table 4.3: In-time Analysis for the Lab and Beam experiments.

sensor by sinusoidal excitations with a small and large amplitude. Then, we study the results achieved in the in-lab experiments with a real-life scenario on a concrete beam.

4.3.1 Noise Analysis

Ambient noise is a consistent, non-zero element present in acquisition systems. A low noise level is essential in designing analog and digital devices such as sensors to avoid the inference of small valuable signals and noise.

Since the advanced processing methodologies that assess a structure's condition deploy both the time and frequency domain, we performed noise analysis in the time and frequency domain. Table 4.2 indicates that the piezoelectric sensor, PCB393B12, benefits from the lowest noise level with only $0.06mg_{\mathrm{RMS}}$ and $10.28\mu g/\sqrt{Hz}$ for the time and frequency domain, respectively. The second best place is analog MEMS, where the noise level is approximately double that of the PCB sensor. On the contrary, the noise level of the digital MEMS accelerometer increases drastically where Table 4.2 reports that ISM in low-power suffers from $2.2\,mg_{\mathrm{RMS}}$ to $436.2\,\mu g/\sqrt{Hz}$, i.e. one order of magnitude higher than the piezoelectric sensor in both time and frequency domain. The former difference is due to the low power consumption of the digital sensors since there is a trade-off between noise level and power consumption. To conclude, the results in Table 4.2 show that the costly piezo sensor benefits from a low noise level; thus, it could be the best choice for monitoring systems with no constraint on power consumption. However, analog MEMS can provide a similar noise level for dense scalable monitoring systems with constraints on budget and power.

4.3.2 Time Analysis

Recent studies have demonstrated the feasibility of deploying raw time-series signals as input features to monitor large-scale structures, especially damage detection methodologies. Therefore, we fit a model based on the formulation described in Sec. 4.1.3. For the in-lab experiments, we fit the acquired data of a sinusoidal impulse where the two critical parameters are signal amplitude and frequency. Next, we study the signal frequency and decaying exponential factors for the real-life experiment. Table 4.3 reports the result of the fitted models for the former parameters.

Experiment		Peak Frequency [Hz]				Peak Amplitude [mg]				Peak Width [Hz]			
		PCB	LIS	ISM_H	ISM_L	PCB	LIS	ISM_H	ISM_L	PCB	LIS	ISM_H	ISM_L
LAB	$30\,\mu m$	10	10	10	10	0.05	0.04	0.05	0.05	0.1	0.1	0.1	0.1
	$250\,\mu m$	10	10	10	10	4.80	4.23	4.50	4.03	0.1	0.1	0.1	0.1
Experiment		Peak Frequency [Hz]				Peak Amplitude [μg]				Peak Width [Hz]			
BEAM		5.5	5.5	5.5	5.5	5.39	3.53	5.60	6.79	0.4	0.4	0.4	0.4

Table 4.4: In-frequency Analysis for the Lab and Beam experiments.

For the in-lab experiment, the small amplitude stimuli, i.e., $30\mu g$, Table 4.3 reports that it is more complicated for digital MEMS in LP mode to fit the exact amplitude of the stimuli compared to the piezo one. However, the digital sensor in HP mimics the piezo by zero error. In low-power mode, ISM deviates from the nominal value because the signal with a small amplitude is equal to the noise level; hence, the fitting algorithm is not capable of finding the optimum solution for the data. By increasing the nominal value to $250\mu g$, ISM in low-power mode error is less than $30\,\%$, while for the other two cases, the error diminishes to less than $10\,\%$ compared to the piezo one, indicating that MEMS operates better with larger signals than smaller ones. Furthermore, all the sensors can perfectly fit the signal's frequency with a slight error for the ISM in LP mode compared to the nominal value. For the real-life experiment, we consider the value obtained by the piezo as the most accurate one, with a $5.5\,\text{Hz}$ signal frequency and $8.3\,\text{m}$ decaying factor. The bottom part of Table 4.3 reports that ISM in HP and LIS can mimic the behavior of the piezo with the same decaying factor, whereas the ISM in LP decays faster by $41.20\,\%$. Next, the signal frequency section of Table 4.3 reports that LIS has the closest estimation to the piezo with only $0.04\,\text{Hz}$ difference, while ISM has a similar error with $0.09\,\text{Hz}$ and $0.07\,\text{Hz}$ for HP and LP modes compared to the piezo sensor, respectively. In conclusion, MEMS-based sensors imitate the piezo manners in acquiring frequency and amplitude signals in the time domain. However, the ISM in low-power decays faster than the other two sensors due to low sampling rate and high noise level.

4.3.3 Frequency Analysis

Frequency Analysis is one of the primary methods civil engineers use to monitor structures' modification and identify damages by deploying metrics like peak frequency, amplitude, and width. We utilize the PSD method described in Sec. 4.1.3 to extract former frequency parameters. For the in-lab experiment, we translated 3 minutes of time-series acquired acceleration into smaller non-overlapping 10-second windows, resulting in 18 windows ($\frac{3\times60}{10} = 18$). Furthermore, for the real-life experiment, we deployed only one event with $15\,\text{s}$ captured by all the sensors to avoid any heterogeneity in the frequency domain evaluation. Consider that a window size of more than 10 seconds does not impact the result of PSD since no structural response lasts more than 10 seconds. However, windows of less than 5 seconds cause a drop in the accuracy of frequency analysis.

The first part of Table 4.4 reports the results obtained for the LAB experiments. The excitation stimuli applied by the shaker have a frequency of 10 Hz, which is captured as the first natural frequency by all the sensors in all scenarios. Notice that ISM in LP mode works as accurately as the piezoelectric sensor, even in the smallest input range. Compared to the piezoelectric sensor, the most accurate sensor, the results reported in the peak amplitude section of Table 4.4 indicate $11.5\,\%$, $4.0\,\%$, and $3.4\,\%$ deviation for LIS, ISM in HP, and ISM in LP mode in peak frequencies' amplitude identification, respectively. Considering the width of the natu-

ral frequency, the last section of Tab. IV indicates a narrow window for all the peak frequencies, fixed at $0.1\,\mathrm{Hz}$ for all the sensors. Furthermore, the second part of Table 4.4 reports the result of the real-life event. Considering the peak frequency, the piezo sensor identifies $5.5\,\mathrm{Hz}$, further characterized by the MEMS sensor. The peak amplitude section of Table 4.4 demonstrates that similar to the Lab experiment, the analog MEMS deviates the most by $1.86\mu g$, i.e., $34.5\,\%$ compared to the piezo one. Notice that digital MEMS in the LP mode also reports $1.4\mu g$ ($25\,\%$) mismatch compared to the accurate piezo sensors and with dissimilarities close to analog ones. This mismatch is due to the low sampling rate of both MEMS, which cannot capture the whole amplitude of the exciting stimuli. Finally, the peak width section in Table 4.4 provides similar results to that of peak amplitude, in the sense that digital MEMS, ISM in HP mode, is capable of following piezo sensor, whereas analog and digital MEMS in LP mode deviates from piezo by $0.03\,\mathrm{Hz}$. The former results mainly characterize two outcomes. The former is both analog and digital MEMS sensors are feasible candidates to replace the costly, power-hungry piezo sensors, whereas the latter is the fact that low sampling frequency can achieve reasonable accuracy modal frequency analysis and peak width up to maximum $7\,\%$ error rates. Although deploying a lower sampling rate impacts parameters like network traffic, it performs less accurately than a high sampling rate, given the peak amplitudes parameter.

CHAPTER

5

APPLIED SIGNAL PROCESSING

This chapter is dedicated to explaining the three applications that have been developed, showcasing the feasibility of shifting partially or fully processing workloads from the cloud to the edge near the sensors. In the first application, two key challenges, automating anomaly detection and doing so in a scalable manner, are discussed. An anomaly detection pipeline is developed and deployed at three levels: cloud, edge-cloud, and entirely on the edge. Further, in the following section, the data reduction algorithm is discussed to show the pros and cons of parallelizable tasks in single-core and multi-core scenarios supporting our sensor node design, including an ultra-low power parallel computational engine described in Chap. 4. Finally, the last section focuses on one of the latest fields in SHM systems, which is to deploy a vibration-based system as a traffic load estimator to cluster vehicles based on their gross weight. The solutions proposed for this application are among lightweight signal processing approaches already present in embedded system applications.

5.1 Data-driven Vs Model-Based Algorithm

This section describes the main contribution of this application. A novel SHM system, pipeline, and its deployment to augment the SHM installations to automatically raise *integrity* alarms is discussed. First, the installed SHM systems and the viaduct, along with the benchmark acquired from the bridge, are described. Then, the complete pipeline, comprising a step of initial training, the in-field estimation, and the possibility of an online update of the models, is discussed. Finally, the proposed solutions to efficiently embed the pipeline inside the existing system, reducing energy consumption and network traffic while maximizing the system's scalability for large SHM installations is presented.

5.1.1 SHM Installation

Bridge Structure

The vibration data analyzed in this work comes from a viaduct located in northern Italy on the ss335 state highway, which is composed of 18 different sections. In year 2019 , the viaduct underwent a technical intervention to strengthen the structure of a viaduct section, with a corresponding change in the vibration signal produced by its structure. Before the intervention, this section has been instrumented with five SHM nodes to monitor viaduct vibration, as illustrated in Fig. 5.1. For this reason, in this application, we analyze the unique situation of accelerations gathered before and after this strengthening intervention. We use these data as a proxy for an abrupt change in the viaduct structure caused, for example, by external factors such as an earthquake. After the intervention, we consider the vibrations raw data as the normal data produced by a 'sane' viaduct. Conversely, the accelerations measured before the intervention are used as the 'anomaly', given the high degradation of the bridge's structure.

SHM Network

The depicted installation is a vibration-based SHM system, which exploits acceleration gathered from the sensors to detect damages and monitor the viaduct's health condition. Fig. 5.1 shows the installation composed of five nodes connected via CAN-BUS to transmit data to a gateway: in the baseline setting, no computation is performed on the nodes or the gateway. Nodes gather and transmit data to the gateway. The gateway sends the sensor's data to the cloud for storage purposes. All the data processing is then carried out daily on the cloud.

The gateway is a Raspberry Pi 3 module B (RPi3), an edge computer actively employed in many fields such as robotics, smart sensors, or SHM. It includes a Broadcom BCM2837 SoC, with 64-bit 4-core Cortex-A53 running at $1.2\,\mathrm{GHz}$ and $1.2\,\mathrm{GB}$ of DDR2 RAM. The gateway supports the Linux operating system, allowing for typical Python applications deployment for either communication (e.g., an MQTT broker [53]) and in-field machine learning (e.g., Keras [102], scikit-learn [94]).

The sensor node is represented in the lower part of Fig. 5.1. It is composed of the LIS344ALH analog tri-axial accelerometer [99], the HTS221 temperature and humidity sensor [100], and an STM32L476VGTx microcontroller as a computational core. The core features a floating-point unit and a digital signal processing (DSP) library, which has been used in our work to optimize the algorithm deployment. The microcontroller unit is an ARM 32-bit Cortex-M4 running at $80\,\mathrm{MHz}$, with $96\,\mathrm{kB}$ of SRAM and $1\,\mathrm{MB}$ of Flash memory. This node samples the acceleration with the internal ADC at a frequency of $25.6\,\mathrm{kHz}$. For increasing the bit-resolution, windows of 256 samples are filtered with a 6-state FIR filter and reduced to a single

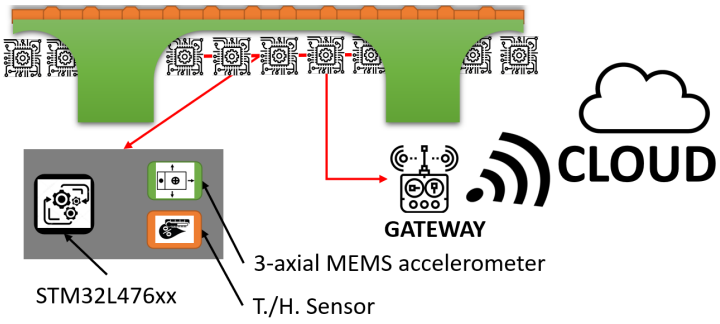


Figure 5.1: Overview of the installed monitoring system on the viaduct. Five sensors are linked to a gateway for streaming data to the cloud. The grey box showcases the main components of a sensor node.

value, thus obtaining a final sampling rate of $100\,\mathrm{Hz}$.

Benchmark

Our test dataset comprises $25\,\mathrm{days}$ of continuous monitoring of the viaduct with $5\,\mathrm{sensors}$ described in Sec. 5.1.1. For our analysis, we consider the central sensor of the chain, which is most influenced by the viaduct vibration. Note that using a higher number of sensors does not improve the accuracy in this case, but it is still feasible. The data are composed of $5\,\mathrm{days}$ before the maintenance intervention, labeled as anomalies, and $20\,\mathrm{days}$ after, labeled as normal data. We select as the test set all the $5\,\mathrm{days}$ of anomalies and $5\,\mathrm{days}$ of normal data to have a balanced test dataset. We divided the remaining $15\,\mathrm{days}$ of normal data into a validation set ($5\,\mathrm{days}$) and a training set ($10\,\mathrm{days}$). Note that anomalies are used neither in training nor validation datasets, given that all analyses are unsupervised. The anomalies are used in our results only to assess the classification accuracy of our approaches. To the best of our knowledge, considering the viaduct's unique condition, this is the first anomaly labeled data from a real-life viaduct.

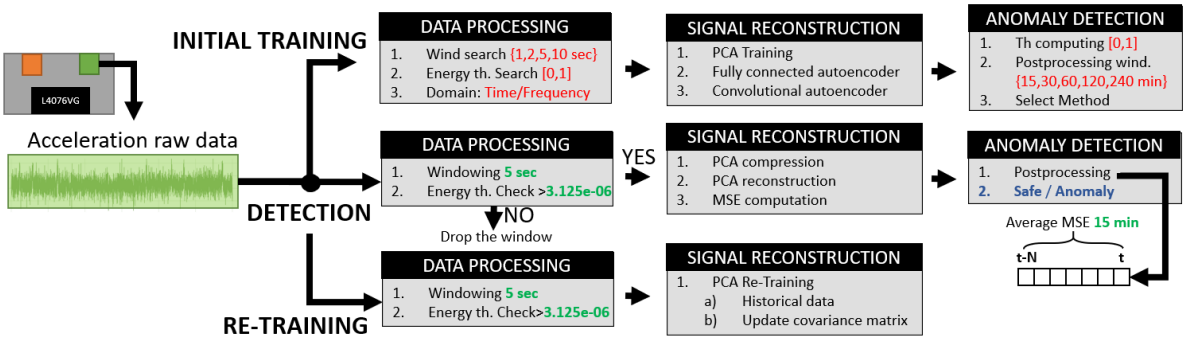


Figure 5.2: The proposed framework to analyze the condition of a viaduct starting from raw acceleration data. In the top part of the figure, we show the hyper-parameter tuning (in red) and the initial training steps done before the monitoring system was activated for the first time. In the middle, we show the inference steps to be done continuously for safe/anomaly condition assessment. In the bottom part, we show the possibility of updating the signal reconstruction algorithms after the pipeline detects an abnormal event to avoid the increase of false-positive alarms due to the bridge's static deformation caused by wind or aging.

5.1.2 Methods: Anomaly Detection in an SHM framework

As shown in Fig. 5.2, our pipeline comprises three main blocks (from left to right in the figure). First, a series of transformations, such as windowing, data filtering, and feature extraction, is applied. Then, the signal compression-decompression algorithm for anomaly detection is applied. We tested three algorithms: i) PCA, ii) a fully connected autoencoder, and iii) a convolutional autoencoder. Finally, the MSE between the decompressed and original signals is computed to detect the structural integrity of the viaduct. An average over time is calculated to smooth the damage detection, reducing false alarms.

Pre-processing

This step covers the windowing of the raw signal, the energy extraction, and, eventually, the application of the FFT if needed. We used a single acceleration axis for our analysis, namely the z-axis (i.e., vertical axis) of the sensor installed in the middle of the section since it contains the most critical information about the bridge.

As Fig. 5.2 shows, data processing starts by dividing acceleration raw data into non-overlapping windows, similar to [19]. We explore window dimensions of 1 to 10 seconds to balance accuracy with algorithm complexity. Noteworthy, given the hardware-related constraints such as limited memory and hard time constraints, different window dimensions can fit different use cases.

After, we check the energy of the windowed signal. In our case, the analyzed bridge does not experience heavy traffic; hence, it often results in low-vibration windows containing only the white noise of the sensor. Therefore, we designed an energy-based window cleaning to eliminate non-informative windows. To this end, the energy of each window is extracted and compared to a trained energy threshold. Windows with an energy lower than the trained threshold are removed from further analysis.

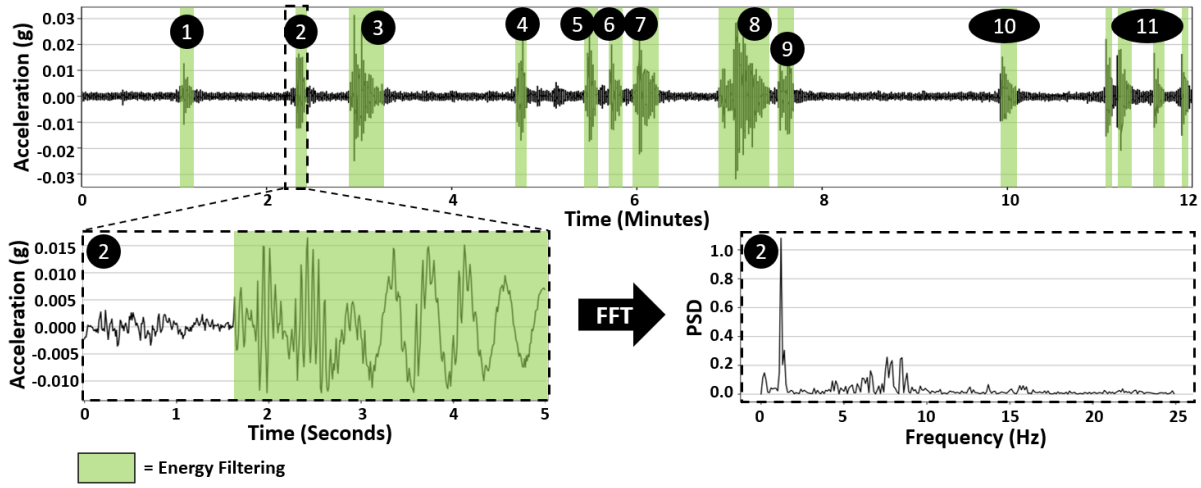


Figure 5.3: Top panel: Twelve minutes of mean-centered raw acceleration data of the $z-$ axis of the middle sensor installed on a bridge span. Peaks are associated with vehicle passages. Left panel: Zoom of a 5-second window containing the oscillation associated with the passage of a vehicle. Right panel: The frequency response of the window of the signal is highlighted with the dashed rectangle.

The energy of each window is computed as:

$$E = \sum_{i=1}^{W_d} X_i^2 \tag{5.1}$$

where W_d is the width of each window. The search for energy threshold is done by exploiting the iterative steps of Alg. 1. At each step, increasing the threshold leads to removing a higher percentage of the windows. Alg. 1 stops when the reconstructed signal of not filtered-out windows drops below a predetermined Quality of Service (QoS), namely the average reconstructed signal-to-noise ratio (RSNR), computed as $\mathrm{RSNR} = 20\log_{10}\left(\frac{\|\mathbf{x}\|_2}{\|\mathbf{x}-\hat{\mathbf{x}}\|_2}\right)$, with $\mathbf{x}$, the original signal, and $\hat{\mathbf{x}}$, the reconstructed one. Based on [19] and considering a compression factor of $15\times$ as in [19], we set this lower bound average RSNR to $16\,\mathrm{dB}$. Fig. 5.3 shows acceleration

Algorithm 1 Energy Filtering

Input: $\mathbf{X_{train}}, \mathbf{X_{val}}$
$th = 10^{-10}$
do
    $th += 2^{-8}$
    $\mathbf{X_{train}}, \mathbf{X_{val}} \leftarrow$ filter $(\mathbf{X_{train}}, \mathbf{X_{val}}, th)$
    $W \leftarrow pca(\mathbf{X_{train}})$
    $X_r \leftarrow \mathbf{X_{val}}\mathbf{W}\mathbf{W}^\intercal$
    $S \leftarrow RSNR(\mathbf{X}, \mathbf{X_r})$
while $S < 16$ dB
Output: th

data and highlights the portion of the signals selected by the tuned energy threshold with a

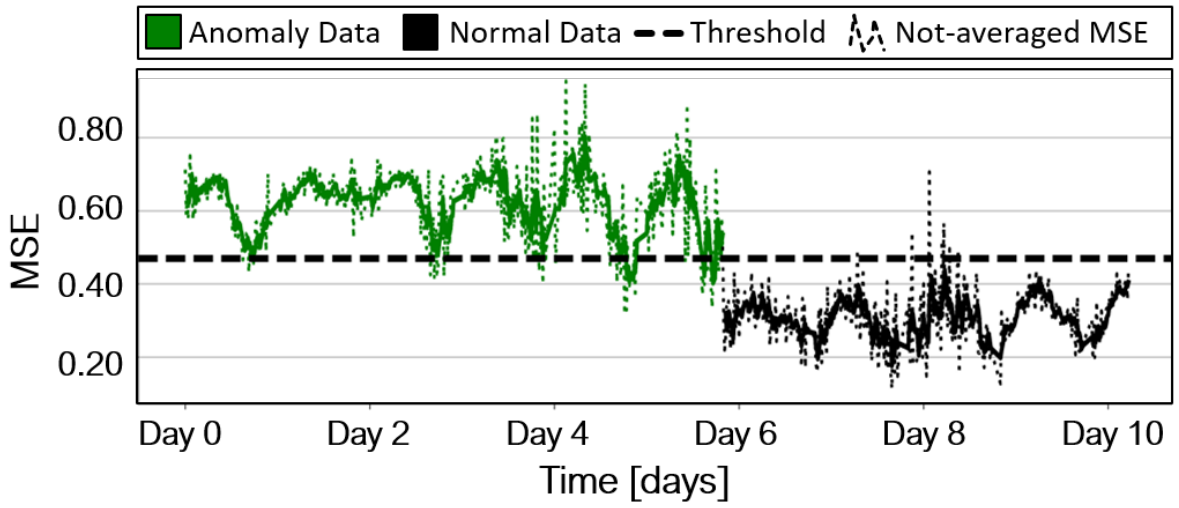


Figure 5.4: PCA output mean square error (MSE) on the test dataset. The input window dimension is set to 5 seconds. The solid line is obtained by applying the post-processing with window dimension = 1 h.

green background. Fig. 5.3-B and Fig. 5.3-C show a zoom of peak ❷ in the time and frequency domain. As detailed in Sec. 5.2.2, applying this energy filtering improves the accuracy of all our analyses.

Signal Reconstruction

We process the non-discarded windows with different compression-decompression models. The similarity of the original with the reconstructed signal is then used to detect anomalies.

This phase is split into two steps: i) compression and ii) reconstruction of input pre-processed signals. We test one model-driven method, namely the PCA, and two data-driven approaches, a fully connected autoencoder and a convolutional autoencoder, as anomaly detectors. We impose a compression factor of the input signal of $16\times$ before reconstruction. In PCA, we keep the top 16 principal components. In the fully connected autoencoder, we employed 16 neurons in the hidden layer. In the convolutional autoencoder, we utilized a stride over convolutional layers of the encoder part of 32, reducing from 500 to 16 the dimension of the signal before the transposed convolutions. The PCA and fully connected autoencoder perform the same number and type of operations (two matrix multiplications, $\mathbf{A}\times\mathbf{B}$, and $\mathbf{B}\times\mathbf{C}$, with dimensions $\mathbf{A}$ 1×500, $\mathbf{B}$ 500×16 and $\mathbf{C}$ 16×500) and only differ in the training approaches: the first one is model-based, while the second is trained via a data-driven back-propagation. The convolution autoencoder comprises 8 hidden layers followed by ReLU activations. Adam optimizer, along with 80 epochs, is used to train this model.

Anomaly Detection

We use the difference between the original and reconstructed signals as an anomaly detection score. A higher difference implies a worse reconstruction. In particular, we compute the mean

square error (MSE) as described in Eq. 5.2:

$$MSE = ||x_i - \bar{x_i}||_{L2} = \frac{1}{n}\sum_{i=0}^{n-1}(x_i - \bar{x_i})^2 \tag{5.2}$$

where x is the original signal, $\bar{x}$ is the reconstructed signal, and n is the number of samples in a window.

Our reconstruction algorithms are trained solely with normal data using an unsupervised process. Therefore, the algorithms should reconstruct normal data with low MSE, while they cannot reconstruct anomalies that show a different signal dynamic not seen during the training, leading to a higher MSE. A threshold to distinguish the two data classes can be thus statistically derived solely by normal validation data. To compute it, in our case, we compress a validation set of normal data using different compression algorithms. Then, we select the threshold as the mean of the MSE over all the data compressed plus three times its standard deviation ($th = \mu + 3 \times \sigma$). Noteworthy, we set this threshold to have only $0.01\,\%$ of statistical false positive errors. The results of this procedure are shown in Fig. 5.4, where PCA is used to compute MSE over normal and abnormal data.

We propose an average over time of the soft predictions (MSE values) to further reduce false alarms. We explore windows between $15\,\mathrm{min}$ to $4\,\mathrm{h}$, showing that a larger window positively correlates with better accuracy, increasing the gap between reconstructed normal data and reconstructed anomalies but causing larger delays in prediction.

Algorithm Phases: Train, Detect, Re-Train

Our pipeline is characterized by three main phases (Fig. 5.2, top-down), namely i) an initial algorithm selection, parameter tuning, and model training, ii) the continuous bridge monitoring, and iii) a re-training phase to adapt the model to slow modifications of the bridge dynamic.

The first phase, *training*, begins with an ablation study over the possible hyper-parameters: the input window dimension, the tuning of the energy filtering step, the anomaly detection models parameters, and the post-processing. After defining the parameters, the chosen model is trained with the normal data of the viaduct.

The second phase, *continuous monitoring*, exploits the best solution found during the training to perform a long-term online detection of the viaduct damages.

The last phase, *re-training*, involves updating the model parameters over time to adapt to the temporal-changing dynamic of the signal. This step is primary for this kind of analysis since modal analysis shows that light stresses such as wind or traffic load cause slow structural modifications, resulting in slightly different signal dynamics. Further, in SHM scenarios, false alarms can not be tolerated since they can trigger critical alarms, causing a bridge maintenance intervention with a consequently high cost.

5.1.3 Deployment: Sensor Vs. Cloud

Deploying our proposed anomaly detection pipeline (Fig. 5.2) is not trivial due to problems such as the scalability in the number of nodes or the lifetime of the nodes. Data communication costs become critical when multiple streams must be transmitted to the cloud. At the same time, the limited memory footprint of tiny edge devices is a major constraint for on-sensor computing. Therefore, we here discuss three deployment scenarios of our anomaly detection pipeline on our SHM system, composed of the sensor network installed on the viaduct and the cloud that augments the system with data storage and computation capabilities. Specifically,

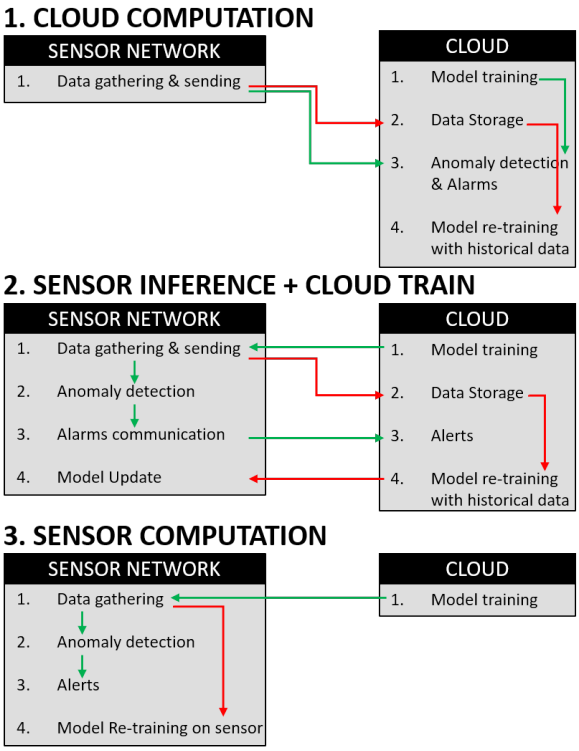


Figure 5.5: Three deployment scenarios of our anomaly detection pipeline. Green arrows highlight the inference steps, while red ones highlight the re-training and updating of the model over time.

we discuss the trade-offs of performing the three phases (i.e., training, anomaly detection, and re-training) of our algorithm either on the cloud, on the nodes, or as a mix of them. Noteworthy, these reasonings also hold for much bigger SHM installations of hundreds of nodes, where the scalability issues and the data storage can be the real bottleneck of the system.

Cloud Computing

As shown in Fig. 5.5-1, transmitting all the data to the cloud while having no processing in the sensor network causes i) a high data communication cost, ii) the necessity of cloud data storage, and iii) a daily cloud computation of anomalies, alarms, and, less frequently, the iv) re-training of the model.

Data transmission to the cloud is the first issue in this scenario. Although several cost minimization techniques, such as new communication paradigms [32,76] or edge data-reduction [19], have been introduced recently, data communication still represents the highest installation cost over months in terms of energy. Using one of the most efficient standard protocol stacks available today, the Narrow Band Internet of Things (NB-IoT) [121], which has demonstrated optimal performance in the SHM field, the system consumes up to 0.94 J for a typical transmission of 500 bytes in the open space, decreasing the maximum lifetime of the SHM nodes and thus needing solutions such as energy harvesting [107] or a wired sensor. Furthermore, the different cloud service providers such as Amazon, Microsoft, and Google account for data computation costs as pay-to-go, with the client paying for the computational time exploited [56], also making the money invested in this service not negligible. Therefore, a complete cloud paradigm for anomaly detection causes a higher maintenance cost and shortens the lifetime of the SHM

nodes, demanding more frequent interventions on the installation.

Sensor Interface with Cloud

Involving sensors in the computation reduces the costs of the anomaly detection pipeline. The anomaly detection model is exported to the sensor to predict the viaduct behavior, while the model re-training is still performed in the cloud. Fig. 5.5-2 shows the overall functionality of this approach. In green, we highlight the anomaly detection pipeline; in red, we highlight the model update over time. Note that while we can reduce both the traffic (streaming only data when we decide to start a re-training) and the cloud computation (only the re-training function is executed on the cloud), cloud storage and processing cost still remain an issue for this kind of scenario, making the scalability an open problem in this kind of approaches.

In our use case, we deploy the anomaly detection pipeline on the node for this scenario while keeping the data streaming to the cloud for algorithm re-training. After the on-cloud algorithm re-training, the new model is deployed on the nodes.

Sensor Computing

To also eliminate the communication costs for re-training, we propose to move both the computation of the online anomaly detection and the update of the node's model on the sensor. Using this approach, after the initial training, done once per SHM installation, no further computation is required from the cloud. Each SHM installation can be considered a standalone unit without the need for cloud communication unless an anomaly is detected. In this scenario, scalability is no longer a problem since the cloud only monitors and initializes the sensors' status and initialize them. Fig. 5.5-3 highlights the steps of this approach.

For our use case, while the porting of the anomaly detector is trivial, training PCA on a memory-constrained device entails many challenges, such as storing the covariance matrix in a memory-constrained microcontroller. Further, storing many data on local nodes is impossible, given the low FLASH memory. Thus, we employ streaming PCA, previously deployed on a sensor node in [19], which aims at finding a compression matrix sequentially to avoid i) storing lots of data at the edge and ii) computing the entire covariance matrix [113]. Compared to [19], instead of employing the PCA only for data compression, we also use it to perform anomaly detection at the edge of the sensor network.

5.2 Results: Data-Driven Vs Model-Based

This section mainly focuses on analyzing the proposed framework in Fig. 5.2. Using grid search over different hyperparameters and framework elements, we explore our pipeline's performance while changing its blocks (e.g., anomaly detection techniques) and the block's parameters (e.g., by presence or absence of the energy filtering while tuning its threshold). After, we examine our best detector's robustness, artificially changing the severity of the anomaly in the dataset and correlating severity with algorithm performance. Then, we compare the proposed anomaly detectors with state-of-the-art ones. To be fair, we reproduced the state-of-the-art algorithms and applied them to our data using the same input window dimension and post-processing.

The second phase of this section will analyze several deployments of the best algorithm found in the performance analysis, namely, the PCA, on the processing unit introduced in Sec. 5.1.1, the STM32L476VGTx. All pipeline steps, including data processing, signal reconstruction, and anomaly detection, have been deployed using optimized C code and the FreeRTOS operating system. Initially, we tuned the CF of PCA with multiple input dimensions against memory constraints to realize the utmost limit of PCA deployment with a floating-point compression matrix. We further address each case's energy consumption and execution time to report its pros and cons. We then fix CF for the best performance and perform interference with MCU to compare its performance with the offline version. Finally, we present a comparison of the best solutions to show the pros and cons of the three scenarios discussed in Sec. 5.1.3.

5.2.1 Algorithm Exploration

Notations & Benchmark

First, we introduce the notations and metrics we use to evaluate this application's different methods and hyperparameters. We use three metrics for performance assessment:
i) accuracy, the total correctly classified windows

$$Acc. = \frac{TP + TN}{TP + FP + TN + FN}$$

ii) sensitivity, the percentage of correctly detected anomalies

$$Sens. = \frac{TP}{P} = \frac{TP}{TP + FN}$$

iii) specificity, the percentage of correctly classified normal windows

$$Spec. = \frac{TN}{N} = \frac{TN}{FP + TN}$$

Where P are the positives, N the negatives, TP are the true positives, TN are the true negatives, FP are the false positives, and FN are the false negatives. Furthermore, we use Area Under Curve (AUC) to assess the performance of our models. For our purpose, we consider the "anomalies" as positives prior to the intervention, while the negatives are windows of "normal" data after the intervention. With Compression Factor (CF), we point to the ratio between high-dimensional original space and algorithms-reduced data space, i.e., the projected PCA data and the latent autoencoder data. Finally, we define the input dimension, the length of each non-overlapping window in the data processing step, and the output dimension, the total time considered after averaging multiple windows before final classification.

Algorithm	Domain	Acc.	Spec.	Sens.
PCA	Raw	98.8 %	100 %	97.33%
	FFT	77.22 %	99.20 %	50.63 %
	DWT	84.36 %	96.79 %	74.44 %
FC Autoencoder	Raw	68.75 %	99.73 %	44.04 %
	FFT	56.02 %	96.54 %	23.61 %
	DWT	69.99 %	97.87 %	47.66 %
Conv. Autoencoder	Raw	50.6 %	67.2 %	37.1 %
	FFT	56.30 %	85.28 %	32.66%
	DWT	52.12 %	100 %	13.83 %

Table 5.1: Performance of our pipeline changing anomaly detection algorithm with discrete wavelet transform, frequency, and time data as input space domain.

Model selection & Data domain

Modal analysis is the gold standard used to analyze the dynamic characteristics of large-scale buildings [101]. On the other hand, previous studies have demonstrated the feasibility of using raw time series for anomaly detection [19]. Hence, both time and frequency domains are promising directions to analyze. Therefore, we test three anomaly detectors fed with frequency and time inputs. We selected PCA and Autoencoders as detectors given their already demonstrated success in anomaly detection and, more precisely, on SHM tasks [70]. For this comparison, we fix the input window dimension to $5\,\mathrm{s}$ of accelerometer output samples and the output dimension to $60\,\mathrm{min}$. The compression factor is fixed to 16 ; therefore, we select the most significant 32 principal components for PCA while ensuring the innermost latent dimension of both fully connected and convolutional autoencoders has a dimension of 32

Table 5.1 and Fig. 5.6 report the evaluation results on the three models using the $10\,\mathrm{days}$ of the test set, using both the input data domains. As previously described, to report accuracy, sensibility, and specificity, we use a threshold on the output MSE of $\mu + 3 \times \sigma$. On the other hand, the Receiver Operating Characteristics (ROC) curve is threshold-independent. Using time-domain input, PCA outperforms both the other two models, reaching $98.8\,\%$, $100\,\%$ and $97.33\,\%$ of accuracy, specificity, and sensitivity, respectively, and an approximate $1.00\,\mathrm{AUC}$. Notice that PCA is the only method to remove all the false alarms in the system, preventing sending false alarms to bridge maintainers.

Although the fully connected autoencoder mimics the PCA model (i.e., with the same matrix multiplications of the PCA algorithm), it shows a lower performance ($\simeq 30.00\,\%$ drop of accuracy) than PCA due to two factors. First, given the small size of our training set, which negatively affects the data-driven model's performance, it reaches an AUC of only 0.97 Further, the threshold chosen while analyzing only normal data does not permit high accuracy by favoring the specificity. MSE achieved by both anomalies and normal data is very near the test set using frequency domain input data. Therefore, a small modification in the threshold can also impair the accuracy, leading to low sensitivity. Note that we choose this threshold with statistical consideration on the validation dataset, ensuring a specificity $> 99.9\,\%$ on the validation set, but without any assumptions on the sensibility. On the other hand, the convolutional autoencoder does not show promising results, with a very low sensitivity of $37.10\,\%$. This low sensitivity is probably due to the high number of parameters that overfit the training dataset, not allowing it to reach the performance of the other methods.

Comparing frequency and time domains, we first visually analyze the input data. We notice

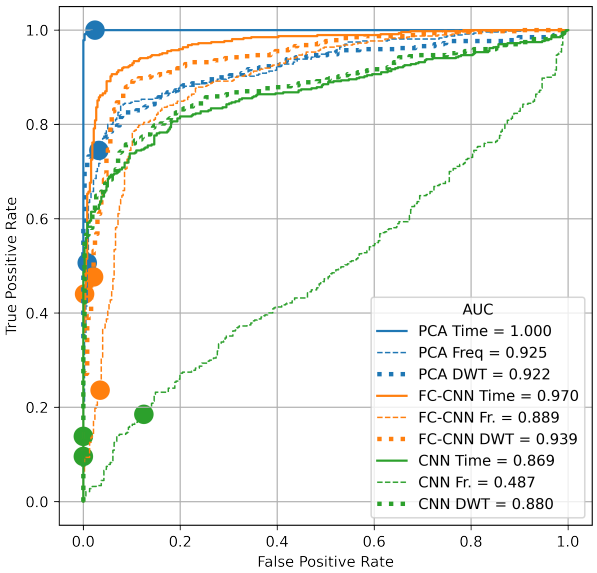


Figure 5.6: ROC curve for different input signal domains, i.e., time, frequency, and time-frequency.

a slightly different waveform between anomalies and normal data in the time domain, given by higher variations in amplitude and lower frequencies in anomalies. These changes are also noticeable in the power spectrum, with a slight deviation in the first natural component of the viaduct. Therefore, we feed our algorithm with either raw data or FFT of each input window. Since the viaduct's natural frequency is relatively low, we cut the frequency spectrum between $0-25\,\mathrm{Hz}$. Although we can reach high AUCs of $0.92\,\mathrm{and}\ 0.88\,\mathrm{for}$ our best models with the FFT pre-processing, we see an improvement using time-domain data. Moreover, our unsupervised threshold training does not allow us to reach a satisfactory accuracy on frequency data. Even though FFT shows a slight difference, it is prone to spectrum leakage due to the measured signal's non-stationarity or non-linearity [61]. To avoid possible spectrum loss, we also evaluate Discrete Wavelet Transform (DWT) approximation coefficients to represent different time and frequency resolutions simultaneously. Thus, we use the DWT coefficients of each 5s window as one other possible input to our anomaly detectors. DWT results reveal that we can reach as high AUC as FFT with $0.92\,\mathrm{and}\ 0.93\,\mathrm{for}$ the superior models in the pipeline. Similarly to FFT, due to unsupervised threshold training, DWT does not reach an adequate accuracy, with only $84\,\%$ and $69.99\,\%$ for the former algorithms. Table 5.1 and Fig. 5.6 summarize the time (Raw data), frequency (FFT), and time-frequency (DWT) results. At the end of this exploration, we select the PCA and the time domain as the best competitors, and we, therefore, use them in subsequent analysis and deployment on edge nodes.

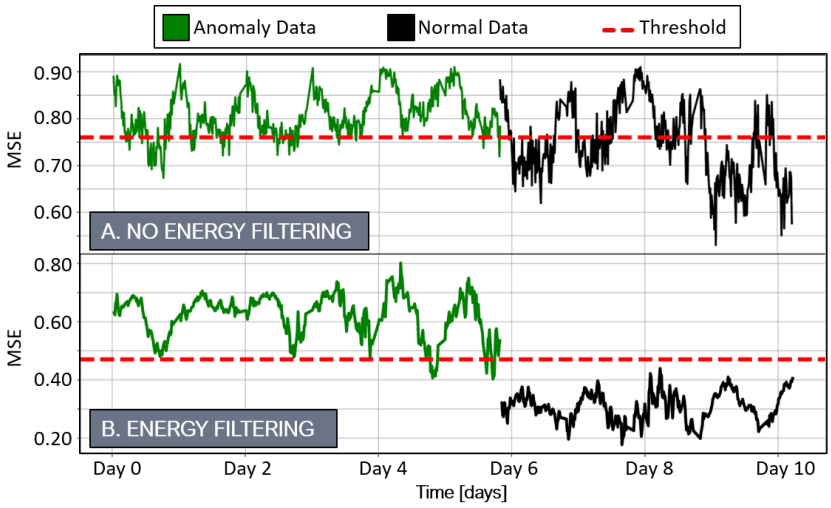


Figure 5.7: The energy filtering step impacts the PCA output MSE. In the top panel, we show the MSE when the energy filtering is not applied. In the bottom panel, we show the improved result with its application.

5.2.2 Hyperparameters exploration

Energy Filtering

In Sec. 5.1.2, we propose filtering non-informative windows to train models with only the most energetic windows, thus removing windows where the viaduct does not vibrate under the passage of vehicles. Our hypothesis is that including all the windows leads to higher reconstruction errors of normal and abnormal data while the gap between the two errors is reduced. Fig. 5.7 quantifies this claim, showing classification with and without the energy filtering block. We can observe that the PCA is strongly affected if we omit this filtering step, with a severe drop of specificity/sensitivity (up to $\simeq 41\,\%$). Notably, the PCA's poor performance is due to the aforementioned increase of MSE of normal windows, whose average move from 0.31 to 0.70 This experiment confirms our initial idea, given that non-energetic windows only contain white noise, which is not autocorrelated. Thus, it is impossible to compress and reconstruct with PCA, leading to high reconstruction errors, similar to anomalies. Therefore, adding this block allows for a strong improvement in the detector performance. The energy filtering is not only beneficial for accuracy but also for computation, reducing the total number of processed windows by $\sim 17\,\%$ on average, thus reducing the total consumed energy.

Input & Output Dimension Exploration

Fig. 5.8 shows the tuning of input and output dimensions, with twenty combinations of four input dimensions and five output dimensions. Input dimension variation is not positively/negatively correlated with algorithm performance. We notice that using $5\,\mathrm{s}$ (grid search between $1\,,2\,,5\,,$ and $10\,\mathrm{s}$) outperforms the other input dimension values from Fig. 5.8. On the other hand, using smaller windows reduces the computation and, thus, the energy consumption of the algorithm execution, leading to a trade-off between energy consumption vs. accuracy. We will better study this trade-off in the following sections.

Contrary to the input dimension, an increase in the output dimension positively correlates with the framework's performance, resulting in a trade-off in delay vs accuracy. However,

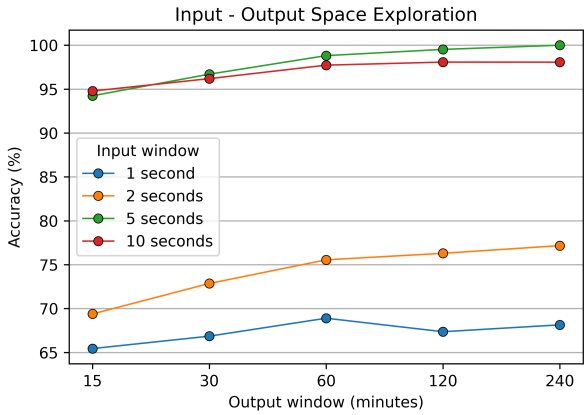


Figure 5.8: Effect of input-output dimensions sweep on the performance of the best detector (PCA).

since viaduct structure modifications are slow, very low delays are not required. Therefore, we decided to use $60\,\mathrm{min}$ of output dimension, which almost saturates performance for 5-second windows while having a reasonable delay. Increasing the output dimension to 120 and $240\,\mathrm{min}$ impressively provides better accuracy. However, the choice of the output dimension, which strongly affects the delay in detecting the status of an anomaly in the viaduct, is related to the specific use case or necessity of the system. For instance, choosing $240\,\mathrm{min}$ as a dimension leads to the perfect distinction of anomalies and safe time slots ($100\,\%$ accuracy) but causes a delay of $4\,\mathrm{h}$ in the notification of the damage alarm.

Compression factor

We also explore different compression factors for PCA to analyze its effect on the framework's overall performance. Intuitively, preserving more high-dimensional space elements does not guarantee enhancement in overall performance since they can improve the reconstruction of both normal and abnormal data. For this reason, starting from our initial value of 16, we further explore CFs = 4, 8, 24, and 32

Fig. 5.9 shows the distribution of MSE values of anomalies and normal data with four values. As expected, a lower compression factor leads to an overall better reconstruction of all the data ($0.05 - 0.30\,\mathrm{MSE}$ with CF = 4) while using a higher CF (CF = 24) causes a higher reconstruction error ($0.4 - 0.9\,\mathrm{MSE}$). On the other hand, none of these conditions implies higher accuracy, given that the critical metric that leads to high classification accuracy is the gap between the two data distributions. Exploring the different values, we find that the original CF value and similar CF = 16 is the sweet spot in this trade-off, leading to a reasonable reconstruction of normal data ($0.1 - 0.4\,\mathrm{MSE}$) and a poor reconstruction of anomalies ($0.4 - 0.8\,\mathrm{MSE}$). As visually noted, the distance between the means of the two distributions is maximized for CF = 16, with a value of 0.30 Simultaneously, other CFs, 4, 8, and 24, only present 0.05, 0.09, and 0.21 distances, respectively. Similar to the input data dimension, this parameter affects both the computation and memory footprint of the algorithm and will be further explored in the following sections.

Synthetic Experiments

Lastly, we artificially generated degradations to monitor the robustness of the best-trained detector,i.e., PCA. To inject different sets of anomalies, we modified the distance between the

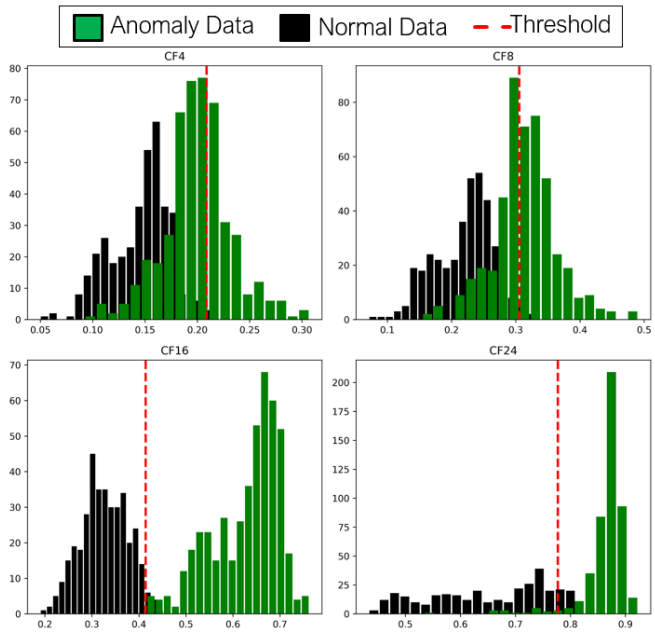


Figure 5.9: MSE distribution while changing CF parameter

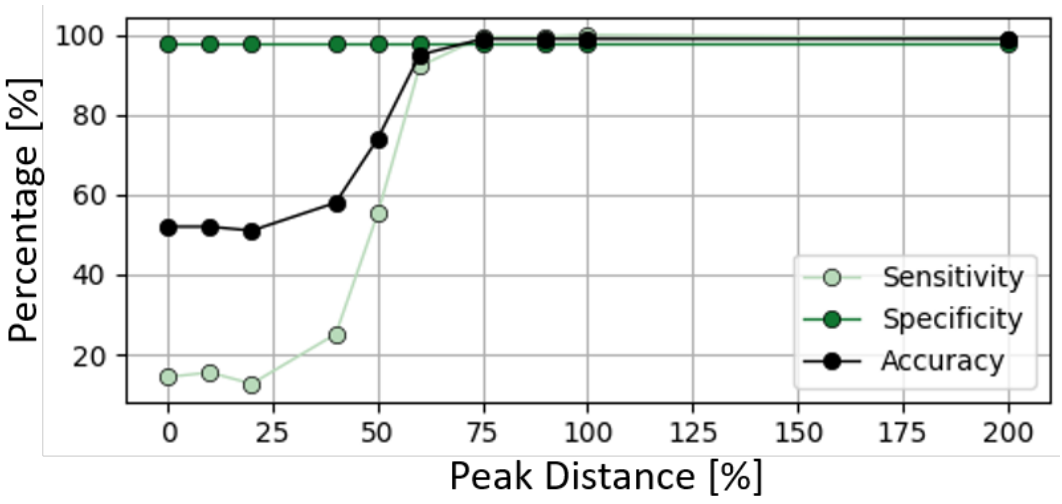


Figure 5.10: Performance of the PCA classifier while sweeping over several severities of anomalies w.r.t real case scenario of the bridge.

two peaks of normal and anomaly in spectrum density and transformed them back to the time domain. We take each $15\,\mathrm{min}$ of the dataset, process with FFT, and gradually close the two peaks of first natural frequency between anomalies and normal data between 0 to $200\,\%$ of the actual distance in the dataset. Note that $100\,\%$ corresponds to the original real-life anomaly. Finally, we transform the data back to the time domain prior and use these new data to test the algorithm. Reducing the gap between the two peaks allows the detector to produce data that is more similar to normal data, implying a harder task for the detector. Fig. 5.10 shows the result of this experiment. Reducing the distance to lower than $75\,\%$ of the original distance causes the detector to reduce its sensitivity, starting to classify anomalies as normal cases. Note that specificity is constant since the anomaly threshold does not change, given that it is computed only with unmodified normal data.

Model Comparison

In this section, we compare our anomaly detectors with methods presented in Sec. 2.3. To do this, we reproduce the pipeline shown in Fig. 5.2, substituting the anomaly detection algorithm with the state-of-the-art ones but keeping the pre-processing and post-processing steps unchanged.

We compare the PCA with four other statistical-based approaches based on frequency peak detection [35], Multivariate Gaussian Distribution [105], and AutoRegressive models [21]. We do not add supervised deep learning methods to the comparison since they require labels for normal and anomaly cases, which are unavailable at training time in normal SHM use cases. Table 5.2 showcases the comparison in terms of accuracy, specificity, and sensitivity.

The literature about anomaly detection in SHM shows that autoregressive moving average (ARMA) residuals are damage-sensitive features of structures [21, 48]. Entezami et al. [39] propose a novel approach to extract these features for big data (GBs). We reproduced their approach on our data, training two different statistical distances, L1 distance and Mahalanobis Square Distance (MSD), to distinguish the normal and abnormal data. Table 5.2 shows that L1 achieves an overall better accuracy (+22.78 %) than Mahalanobis Square Distance (MSD). Santos et al. [35] propose to extract frequency information from the signal and perform the classification based on the position of the main peak of the spectrum. However, in our use case, this method achieves an accuracy of only $67.79\,\%$. This result further proves that frequency features are unsuitable for distinguishing safe and anomalous time windows on our dataset. Finally, we investigated a recent study that targets edge computing [105], exploiting seven

Method	Acc.	Spec.	Sens.
State-of-the-art algorithms			
FFT + peaks detection [35]	67.79 %	99.2%	43.09 %
MGD [105]	59.48 %	95.66%	10.25 %
AR features + MSD [21]	58.43 %	85.90%	36.82%
AR features + L1	81.11 %	88.49%	71.80%
Our Work			
Raw + PCA	98.80%	100%	97.33%
DWT + FC Autoencoders	69.99%	97.87 %	47.66%
FFT + 1D-CNN Autoencoders	56.30%	85.28%	32.66%

Table 5.2: Comparison of our proposed solution with state-of-the-art methods applied to our dataset. The $60\,\mathrm{min}$ post-processing is identically applied to all methods.

statistical features (i.e., mean, mean square, variance, standard deviation skewness, kurtosis, and crest factor) together with a multivariate Gaussian model to predict anomalies in vibrating systems. However, this method also fails in our use case, with a drop in accuracy to $\approx 60\,\%$.

In a nutshell, the results in Table 5.2 show that correlation and autocorrelation of 1-D vibrations are promising solutions (exploited by both PCA and AR models) to detect anomalies in viaducts.

Input dim.	FLASH [kB]	RAM [kB]	Time [ms]	Energy [uJ]
1	32.82	11.12	0.754	3.35
2	40.63	19.95	1.568	12.9295
5	**91.04**	**77.55**	**6.428**	**73.96**
10	276.54	Overflow	-	-

Table 5.3: Deployment metrics of PCA algorithm with CF = 16, output dimension = 60 minutes, and variable input dimension. Time and Energy are only for one inference.

5.2.3 Implementation

CF tuning Vs. Metric Figures

Starting from the accuracy results shown in Sec. 5.2.2 and Sec. 5.2.1, we extensively explore the trade-off between accuracy, memory, and energy consumption by modifying both the CF and the input dimension, with a fixed output dimension of $60\,\mathrm{min}$. We show the results of our exploration in Fig. 5.9. As previously mentioned, CF = 16 results in the best accuracy, with $67.34\,\%$, $76.29\,\%$, $98.82\,\%$, and $97.33\,\%$ for input dimensions of 1 ,2 ,5 , and $10\,\mathrm{s}$, respectively. Despite the higher accuracy of CF = 16 , increasing the CF allows for reducing both the memory footprint and energy consumption. For instance, from the first graph of Fig. 5.9, we can notice that using CF = 24 with a window of $5\,\mathrm{s}$ still allows us to reach an acceptable accuracy of $92.97\,\%$. Contrarily, using a lower CF causes i) higher energy, ii) higher memory consumption, and iii) lower accuracy, excluding these CF values from the trade-off choice. Therefore, We fix the search space to CF $\in [16\,,32\,]$. We also remove the $10\,\mathrm{s}$ input dimension since its compression matrix does not fit the small $96\,\mathrm{kB}$ RAM (dotted line in the second graph of Fig. 5.11). In this region, we found that the only points that reach an accuracy $> 80\,\%$ are achieved for CF = 16 or CF = 24 and input dimension = $5\,\mathrm{s}$. Target application and deployment scenarios can choose the best trade-off between former parameters. Given our pipeline, we found that the largest model that fits the MCU memory is the one with CF = 16 and an input dimension of $5\,\mathrm{s}$, achieving $98.82\,\%$ accuracy with a $73.96\,\mu\mathrm{J}$ energy consumption per inference.

Table 5.3 underlines memory footprint, latency, and energy consumption with a fixed CF of 16 and different input dimensions. Since increasing input dimension corresponds to a more extensive PCA compression matrix, a higher input dimension requires more FLASH space (e.g., $91.04\,\mathrm{kB}$ for $5\,\mathrm{s}$). Although the compression matrix is not a problem (roughly $10\,\%$ of total FLASH for $5\,\mathrm{s}$), the reconstruction procedures occupy up to 77.55 kB (81 %) of RAM for 5 seconds. Such a high usage area puts a solid constraint for embedding the PCA for larger input dimensions, given the option to run other tasks for data gathering. Despite the optimal solution obtained with $5\,\mathrm{s}$, reducing the input dimension to 1 second allows us to maintain an accuracy of $67.34\,\%$, with a latency reduction of $8.5\times$ and $9.5\times$ lower energy consumption. While the former factor brings no obstacle to the system due to the sampling rate ($100\,\mathrm{Hz}$), the latter causes a shorter node lifetime, creating a trade-off between accuracy vs. energy consumption.

Cloud vs. Node costs

Narrowband IoT (NB-IoT) is a recent protocol standardized by 3GPP for Low Power Wide Area, an extension of LTE (4G Long Term Evolution) designed for long battery and low-cost applications where it can virtually work everywhere [14].

NB-IoT consumes more energy per payload packet than other similar technologies. How-

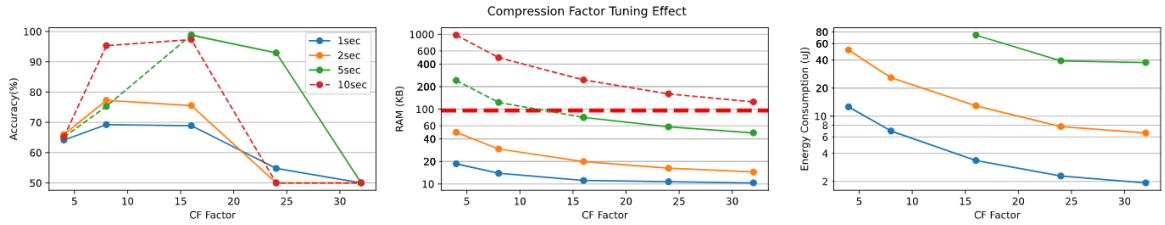


Figure 5.11: CF tuning versus accuracy, memory, and energy. Horizontal red line points to the limit of MCU memory. Dotted lines represent not deployable solutions.

ever, since it has no limitation on the number of bytes sent in a single connection to the cell, it is a prominent transmission protocol in the LPWAN category [14]. NB-IoT deployment of nationally licensed connectivity (e.g., LTE bands) implies no band usage limitation and no latency for streaming acquired data to the cloud. Since in the SHM field, data streaming need not be continuous, and data can be grouped in big batches and sent with a single connection to the cloud, NB-IoT can be an ideal communication option for SHM systems. Therefore, we use NB-IoT to gauge the benefits of the different scenarios introduced in [32].

Fig. 5.2 shows all the options for training and inference. The cloud-based method continuously streamlines the data to the cloud for training and detection phases. In contrast, sensor computation only reports the structure's status to the cloud hourly. We want to quantify these scenarios regarding energy consumption and transition costs to develop a scalable solution for SHM applications.

Table 5.3 reports the deployment results of model inference at the node. The best solution in terms of accuracy, i.e., the PCA with $5\,\mathrm{s}$ input window dimension, consumes $73.96\,\mu\mathrm{J}$. Exploiting a smaller input window dimension, i.e., $1\,\mathrm{s}$, only consumes $3.35\,\mu\mathrm{J}$. Although smaller input windows are $3\times$ more energy efficient, the degradation in terms of accuracy compared to bigger ones is too critical. Furthermore, compared to the cloud paradigm analysis presented in Table 5.4, the energy consumption of the processing unit is negligible. With this in mind, energy consumption is the only counter-effect of larger input dimensions, while other factors like memory footprint and execution time are satisfied. Hence, we keep $5\,\mathrm{s}$ of input dimensions to preserve the performance.

The node installed on the viaduct works with an output sampling rate of $100\,\mathrm{Hz}$; thus, it generates 100 16-bit samples per second. Therefore, the node generates $200\,\mathrm{B}$ per second, leading to $720\,\mathrm{kB}$ per hour. To estimate this node's energy consumption with the NB-IoT protocol, we use the estimations provided in [32], where diversity in the payload for each packet affects the node's power consumption. We decided to use a payload of $1300\,\mathrm{B}$ for this experiment. This selection of payload can send $650\,(13002\,)$ samples per packet. Hence, we need 554 packets to transmit $720\,\mathrm{kB}$ of hourly data. Notice that we send one hour of acquired data all at the same time to leave NB-IoT in the power sleep mode (PSM) for most of the time, reducing the total power consumption. However, storing an hour of data further adds a cost of storage ($\approx 1\,\mathrm{J/h}$) to an off-chip memory (e.g., a micro SD card). Table 5.4 summarizes the energy consumption regarding different sections of both the training and inference parts. It shows that exploiting the full deployment of the cloud computing approach consumes $312.84\,\mathrm{J/h}$, which is reduced to $63.50\,\mathrm{J/h}$ for the localized sensor deployment of our pipeline. An approximate $5\times$ drop in energy consumption for the latter case is due to the low traffic load transmitted to the cloud.

On the other hand, if we bring all the computation to the node where the node only sends the structure's status to the cloud, we can reduce the traffic of the system to only $3\,\mathrm{B}$ (i.e.,

Scenario	Network Traffic (B/H)	NB-IoT E. [J] (1h)	Node Comp. E. [J] (1h)	Gathering E. [J] (1h)
Inference				
Cloud Computation	780 kB	248.85 + E_{sleep}	1.208	62.4
Sens. Inference + Cloud Train	3 B	0.7130 + E_{sleep}	0.005	62.4
Sensor Computation	**3B**	**0.7130 +** E_{sleep}	**0.005**	**62.4**
Train				
Cloud Computation	780 kB	248.85 + E_{sleep}	1.208	62.4
Sens. Inference + Cloud Train	780 kB	248.85 + E_{sleep}	1.208	62.4
Sensor Computation	**0 B**	E_{sleep}	**0.00162**	**62.4**

Table 5.4: NB-IoT deployment cost for the scenarios of our pipeline
$E_{sleep} = 390$ (mJ) is the energy consumption of the node in PSM.
$E_{acq} = 52.596$ (mJ) is the energy consumption to acquire 1 second of data

"OK" or "NOK"). Given the small number of generated data by the node, for this case, we can tune the payload of the NB-IoT module to only $10\,\mathrm{B}$ (the smallest possible number) for each packet. Then, we only transmit one packet to the cloud, leading to less than $1\,\mathrm{J}$ energy consumption. Although our solution reduces transmission costs to the cloud, allowing scalable solutions for large-scale structures, it consumes a high energy rate at the node. Table 5.4 also reports different sides of the transmission vs. node energy consumption trade-off for both training and inference phases. The high traffic rate during inference ($\sim 780\,\mathrm{kB}$/hour) for a complete cloud-based approach prohibits its utilization for large-scale SHM scenarios. On the contrary, our solution reduces the traffic of only $10\,\mathrm{B}$/h to the cloud and is extendable to large-scale systems. On the other hand, the node computation energy is always negligible compared to the energy required to gather the acceleration data, thanks to i) the initial energy-filtering of the windows and ii) the lightweight algorithm employed (PCA).

5.3 System Identification

This section provides a novel data reduction technique for vibration-based systems by estimating the acquired signal's power spectrum density (PSD) on a multi-core device at the extreme edge to overcome memory limitation while optimizing the energy consumption of the in-situ sensor node for the SHM system. Initially, we briefly discuss the case study of this work and the benchmark used to validate the implementation of the multi-core device at the edge. Further, we present the pipeline used to solve system identification models on resource-limited devices, comparing its parallel and sequential types of implementations.

5.3.1 Case Study

Vibration data collected from a four-storey frame structure under white noise base excitation with a sampling rate of $50\,\mathrm{Hz}$ were used for testing and validating the implementation on the target device, i.e., GAP9 Sec. 4.1.4. All the possible combinations of N_p (model order of the AR model) and $N_{s/1p}$ (number of samples per each model order) values were explored by varying the former quantity between 9 and 57 (step size equal to 8), whereas the latter was swept between $\{25\,,\,30\,,\,35\}$. The search space for N_p has been selected to model a wide range of target structures, considering that the number of parameters typically settles below a couple of dozens, even for the most complicated systems [25]. Besides, the choice for $N_{s/1p}$, which is responsible for the length of time series to be processed, has been selected to meet the storage capabilities of the computing unit even in the most cumbersome configurations. We show in Sec. 5.4.1 that the selected upper boundaries are large enough for robust modeling of the PSD of the input signal, which is, however, large enough for robust modeling.

Output-only SysId models for vibration analysis

Two output-only SysId models have been implemented on the target platform since they are among the most effective for processing ambient-excited vibration data: the Autoregressive (AR) and Autoregressive with Moving Average (ARMA) [89]. Given an N-long discrete time-series sampled at regular intervals kT_s (k being the generic time index), the mathematical formulation of the AR and ARMA models are described by Eq. 3.2 and Eq. 3.3, respectively in Sec. 3.1.1. In these expressions, $N_p = p + q + 1$ is defined as the model order, while $e[k]$ is assumed equal to a zero–mean white noise Gaussian term with prescribed variance, serving as a proxy of the unknown input force. The length N is conveniently selected proportionally to N_p according to $N = N_p N_{s/1p}$, $N_{s/1p}$ being the number of time samples necessary to identify one single model parameter accurately. Therefore, SysId aims at computing the N_p model parameters, a task that can be fulfilled by means of ordinary least-squares (OLS) applied to the linear regression form of Eq. 3.2 and Eq. 3.3, which generically reads as

$$Y = \Psi\Theta \tag{5.3}$$

with $Y \in \mathbb{R}^{N\times 1}$ and $\Psi \in \mathbb{R}^{N\times N_p}$ being the measured vibration response and the regression matrix, respectively[1].

[1] For the definition of Ψ, see [123]

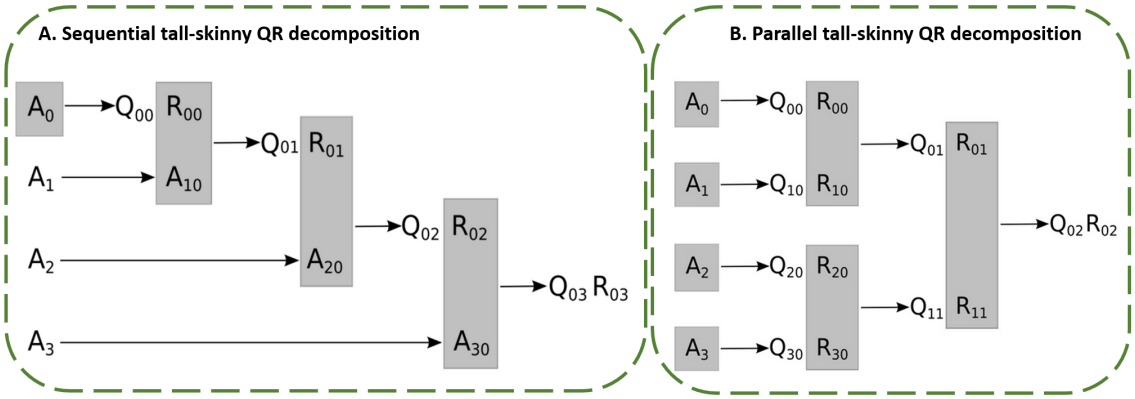


Figure 5.12: Two different implementations of the QR decomposition deployed to solve the System Identification model parameters. Panel A: The sequential implementation of QR decomposition at the edge processing each chunk of data sequentially. Panel B: The parallel implementation of QR decomposition at the edge processing multiple chunks of data simultaneously.

5.3.2 Methods: Sequential Vs. Parallel Tall Skinny QR

Factorizing the regression matrix via QR decomposition, i.e., $\Psi = QR$, is required to reduce the phenomena of numerical instability and rounding effects of the OLS algorithm employed in conventional SysId solutions. This yields the solution of (5.3) to be computed as

$$\Theta = R^{-1}Q^{T}Y \tag{5.4}$$

However, the memory requirements for standard QR decomposition make it inapplicable in most extreme-edge computing domains. Thus, two approaches are studied in the literature to tackle memory limitations at the edge. The State-of-the-Art approach [123] is deemed to solve the problem sequentially since it targets a single-core device at the edge while we introduce the parallelized version of the same approach in this chapter. In Fig. 5.12-A, the sequential implementation of the QR is presented, which focuses on splitting the computation of the input matrix $\Phi \in \mathbb{R}^{N \times N_p}$ into the subsequent decomposition of smaller size $\hat{\Phi} \in \mathbb{R}^{N_r \times N_p}$, with $N_r = N/N_c$, N_c the number of partitions (chunks) the matrix is divided in. *Sequential Tall-Skinny QR*, a procedure based on the iterative QR factorization of the vertical concatenation of the previous-step R matrix and the next chunk [123].

Parallel tall-skinny QR for SysId

The *parallel tall-skinny QR decomposition* (P-TSQR), a different QR decomposition algorithm suited for big data processing frameworks [13], is proposed in this work. P-TSQR has to be preferred over other QR decomposition methods since its parallel implementation scheme (doable for a multi-core processing framework) allows to significantly speed up the computation time over sequential (S-TSQR) variants, such as those exploited in [123] to accomplish the same task.

Given a generic input matrix $A \in \mathbb{R}^{M \times N}$ ($M \gg N$) to be decomposed and supposing that N_c chunks (or cores) are available for parallelization, P-TSQR follows a binary tree implementation requiring $L = \log_2 N_c$ iterations. The latter is depicted in Fig. 5.12-B and can be described as:

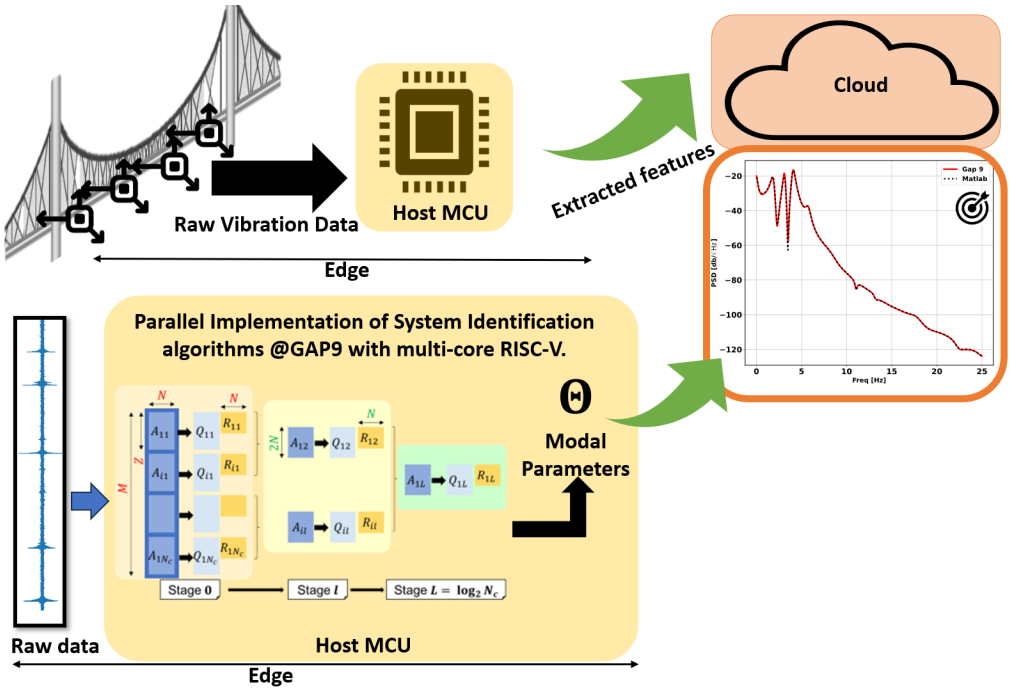


Figure 5.13: Workflow of the proposed P-TSQR-based SysId approach proposed in this application in which merely modal parameters are transmitted to the cloud replacing the raw data.

- *stage 1*: A is divided into N_c partitions of dimension $Z = M/N_c$. For each i-th chunk $A_{i1} \in \mathbb{R}^{Z\times N}$, the QR decomposition is computed independently, leading to $A_{i1} = Q_{i1}R_{i1}$, with $R_{i1} \in \mathbb{R}^{N\times N}$ and $Q_{i1} \in \mathbb{R}^{Z\times N}$ the factorizing matrices.
- *next stages*: at generic stage $l \in [2, \dots, L]$, the R_{i1} matrices are vertically concatenated two-by-two into N_c/l matrices $A_{il} = [R_{i(l-1)} \quad R_{(i+1)(l-1)}] \in \mathbb{R}^{2N\times N}$ which are then decomposed by a new step of QR factorization.

The whole computation ends for $l = L$ resulting in a single $R_{L1} \in \mathbb{R}^{N\times N}$ matrix, that is indeed the R matrix that would have been obtained by directly decomposing the whole initial matrix A, and a certain number of Q_{il} matrices, that can reconstruct the original Q matrix.

Implementation comparison of S-TSQR & P-TSQR

The first noticeable difference between the STSQR and the PTSQR lies in the dimension of the matrices to be decomposed: indeed, considering the same starting matrix $\Phi \in \mathbb{R}^{N\times N_p}$ divided into N_c chunks, the STSQR computes a single QR decomposition of a $N_r \times N_p$ matrix, then N_c-1 factorizations of a $2N_r \times N_p$ matrices, for a total of N_c decompositions (reminding $N_r \gg N_p$). Instead, the PTSQR performs N_c factorizations of $N_r \times N_p$ matrices in the first iteration, while the other steps involve much smaller $2N_p \times N_p$ matrices. Even though the number of QR decompositions is more remarkable for the PTSQR, the difference in the dimension of the matrices to be elaborated resulted in a considerable difference in the computation time. Further, the structure of the TSQR algorithm enables parallel computing, making it a superior solution for the new generation of edge devices that benefit from multi-core system architectures.

Another consideration regards the memory limitations for the entire computation. Parallel and Sequential QR decompositions themselves do not tackle the memory obstacles of the *QR*

factorization because all the intermediate *Q* matrices must be stored for the final parameter estimation; thus, the total *Q* matrix has the same size as the original Φ matrix. An additional step at the end of each QR decomposition addresses the aforementioned memory issues. Specifically, the vector coefficients are updated at each decomposition of Q. This is feasible due to Q^T and observing the particular structure of the Q matrix built on every PTSQR step, the computation of $Q^T\mathbf{Y}$ equals to multiply over time every individual Q to the respective section of the vector coefficient. This avoids storing every Q matrix at each iteration nor computing the total Q matrix.

Data Compression

One may wonder how the system identification methods can act as a data compression technique to mitigate the data traffic load to the cloud. The pipeline developed in this application is shown in Fig. 5.13 indicates that only modal parameters, i.e., θ and γ are sent to the cloud instead of the raw data. By doing so, system identification can reach up to $50\,\text{or}$ even more compression ratio. As it is shown in Sec. 5.4, $35\,\text{samples}$ are sufficient to estimate one single coefficient with the current PTSQR implementation efficiently. Thus, the length of the entire signal is $35 \times$ NO. of parameters. The number of parameters is usually below $20\,,$ even for the most complicated geometries is dependent on the number of relevant frequencies in the spectrum.

5.4 Results: System Identification Parametric Methods

In this section, we focus on analyzing the results of the proposed pipeline in Fig. 5.13 on a multi-core device, namely GAP9. We explore all combinations of N_p and $N_{s/p}$ values presented in Sec. 5.3.1 to find the optimum solution balancing the PSD estimation accuracy vs. memory limitation trade-off at the multi-core source-constrained device at the edge. The first section describes the PSD estimation accuracy of the GAP9 vs. the ground-truth results obtained from the MATLAB running on the Intel core device using an evaluation metric called Itakura Saito Spectral Divergence (ISD).

The second section provides a comparison between the SToA model, i.e., Sequential Tall Skinny QR [123], and our implementation on GAP9, which parallelizes the QR showcasing the feasibility of deploying such a system on the multi-core devices for the two output-only models, namely AR and ARMA with the configurations presented in Sec. 5.3.1. We discuss three main aspects of our implementation, i.e., execution time, memory footprint, and energy consumption of the current solution, showing the optimizations compared to the SToA model.

5.4.1 Data Compression

The Itakura Saito Spectral Divergence (ISD) has been exploited to evaluate the level of spectral superimposition between the Power Spectral Densities (PSD) obtained via built-in MATLAB utilities ($PSD_{MAT}(f)$) and the one estimated from the GAP9 coefficients ($PSD_{GAP9}(f)$):

$$ISD = \frac{1}{N}\sum_{f=1}^{N}\left[\frac{PSD_{GAP9}(f)}{PSD_{MAT}(f)} - \log\left(\frac{PSD_{GAP9}(f)}{PSD_{MAT}(f)}\right) - 1\right] \tag{5.5}$$

PSD curves can be easily computed by moving Eq. 3.2) and Eq. 3.3) in the frequency domain, depending on the selected AR/ARMA model. ISD ranges between 0 and 1 , with ISD =0 a perfect match between the curves, and ISD = 1 indicates the worst case between the curves. Fig. 5.14 highlights one of the cases (($N_p, N_{s/p}$) =(24 ,35)) of the end-to-end application to estimate PSD of the input signal. Notice that the difference between the two cases is that the modal parameters of the system identification model are computed on GAP9 and MATLAB. Fig. 5.14 imposes that the two cases are identical to one another in identifying the main eigen frequencies to detect damages due to the work presented in [123].

Further, the numerical values of ISD are reported in Table 5.5 supports the visual comparison provided in Fig. 5.14. As can be observed, all the values are stably below $1.05 \cdot 10^{-2}$ even for the worst-performing configuration, proving the spectral accuracy of the deployed SysId models. Additionally, it is worth observing that these results compare favorably with respect to the benchmark solution in [123], in which a maximum ISD of $0.93 \cdot 10^{-2}$ was scored when working with the same time series.

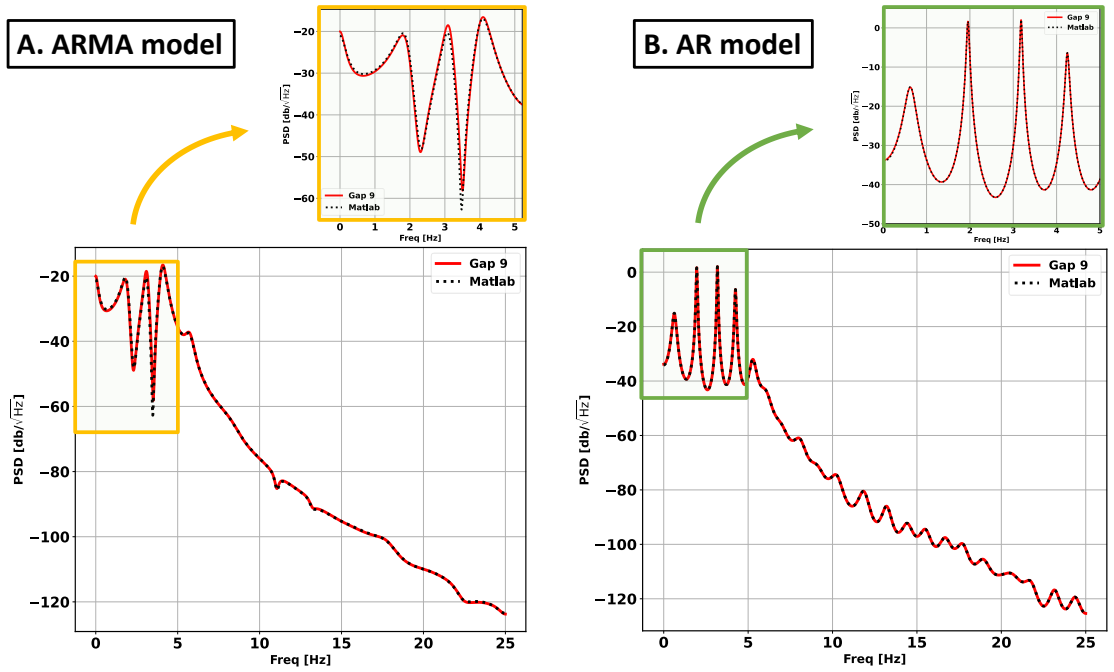


Figure 5.14: PSDs of the AR/ARMA models in a particular configuration (Np = 25 Ns1p = 35), comparison between GAP9 and Matlab

5.4.2 Implementation

Execution Time

Minimizing the algorithmic latency is fundamental to deploying near-sensor streaming data analysis. Accordingly, the execution time necessary to process one batch of SysId model parameters has been quantified by measuring the number of cycles N_{cycles} given the operating frequency F_{ck} of the processor, i.e., Exe. Time $= N_{cycles}/F_{ck}$. In order to perform a fair comparison with the setup in [123], GAP9 was clocked at $F_{ck} = 110\,\mathrm{MHz}$. Results are summarized in Table 5.5.

As expected, the highest computational time is associated with the maximum $N_p = 57$ and $N_{s/1p} = 35$ parameters, i.e., those yielding to the longest vector to be processed (corresponding to 1995 time samples). More in detail, the largest AR and ARMA model requires $0.83\,\mathrm{ms}$ and $1.65\,\mathrm{ms}$, respectively, which are almost $47\times$ and $24\times$ smaller than the time taken to acquire the input signal, the latter amounting to $1995/50\mathrm{Hz} \approx 40\,\mathrm{s}$. Consequently, the devised pipeline proves to be compatible with the real-time execution of SysId at the sensor level.

More importantly, these outcomes significantly outperform the sequential SysId implementation in [123], characterized by worst-case execution times of $52.80\,\mathrm{s}$ and $99.12\,\mathrm{s}$ for the same parameters. Evidence is proved in Fig. 5.15, which showcases the time gain when moving from the sequential to the parallel implementation, expressed as the ratio between the execution time required when running in the STM32L5 board with respect to the one scored by the GAP9 platform. Independently from the considered $N_{s/1p}$, the gain achieved by the parallel solution follows a linear trend, moving from a minimum of $5.44\times$ ($N_p = 9$, $N_{s/1p} = 25$) to a maximum of $60.11\times$ ($N_p = 57$, $N_{s/1p} = 35$) in case the ARMA model is considered. The same speed-up ranges from $6.75\times$ to $63.85\times$ for the AR counterpart. Three main reasons can motivate this significant improvement: i) hardware accelerators for matrix multiplication embedded in GAP9, ii) switch of paradigm from the sequential implementation of QR decom-

		ISD [a.u.]			E [mJ]			Memory [KBytes]			Execution time [ms]		
		$N_{s/1p}$			$N_{s/1p}$			$N_{s/1p}$			$N_{s/1p}$		
		25	**30**	**35**	**25**	**30**	**35**	**25**	**30**	**35**	**25**	**30**	**35**
ARMA	**9**	2.18e-05	3.21e-05	8.07e-06	0.74	0.82	0.75	18.44	22.46	26.88	15.80	17.50	16.00
	17	5.33e-03	9.04e-04	2.43e-03	2.75	3.21	3.18	70.80	86.58	103.93	58.60	68.10	67.70
	25	1.36e-03	2.39e-05	9.30e-03	6.94	8.29	8.77	157.19	192.46	231.25	147.00	176.00	186.00
	33	5.29e-04	4.76e-03	1.13e-02	13.80	16.30	17.20	277.61	340.11	408.86	290.00	345.00	360.00
	41	1.05e-02	8.79e-04	9.25e-03	25.90	28.40	30.40	432.07	529.53	636.75	545.00	600.00	635.00
	49	2.93e-03	5.64e-03	6.42e-03	41.20	46.00	51.50	595.71	733.06	884.46	863.00	965.00	1070.00
	57	9.8e-04	4.55e-03	3.03e-03	62.90	69.50	80.10	843.07	1033.65	1243.38	1300.00	1440.00	1650.00
AR	**9**	2.68e-04	2.41e-04	3.70e-05	0.33	0.43	0.40	15.18	17.93	20.88	6.80	9.10	8.40
	17	8.74e-04	9.20e-04	8.02e-04	1.26	1.50	1.70	57.74	68.44	79.93	26.80	31.80	36.10
	25	8.23e-04	7.88e-04	7.12e-04	3.41	4.52	4.71	127.80	151.65	177.25	72.10	95.90	100.00
	33	5.51e-04	5.42e-04	4.71e-04	7.03	8.28	8.76	225.36	267.55	312.86	148.00	175.00	183.00
	41	13.40e-04	11.70e-04	1.16e-04	12.70	13.60	15.30	350.43	416.15	486.75	264.00	286.00	320.00
	49	9.13e-04	1.63e-04	3.72e-04	20.80	24.30	25.60	502.99	597.44	698.93	434.00	508.00	535.00
	57	1.35e-04	3.87e-04	3.87e-04	32.60	34.30	39.40	683.05	811.43	949.38	681.00	721.00	827.00

Table 5.5: Performance indicators for varying N_p and $N_{s/1p}$ when parallelizing the AR and ARMA SysId models on the GAP9 platform.

position to the parallel version of it, i.e., moving from one core to 8 cores deployment, and iii) in the case of sequential implementation, a large matrix should be segmented to multiple chunks to execute a matrix multiplication, whereas the large L2 memory in GAP9 (compared with a maximum of $256\,\mathrm{KBytes}$ of RAM for the STM32L522 board) allows for the large matrix multiplications to be executed without chunking.

Memory footprint

The precision of SysId routines increases when working with longer time series; however, limitations have to be respected when dealing with memory-constrained devices. Hence, the memory footprint of the models has been evaluated to find the maximum SysId configuration (N_p,$N_{s/1p}$) compatible with the available GAP9 storage capabilities. The memory column in Table 5.5 specifies the space occupied in the L2 memory for each combination, showing that the ARMA model utilizes, in general, nearly $1.3\times$ more memory than the corresponding AR model. This result is coherent with the inherently more complex nature of the ARMA routines [123]. Further, Table 5.5 reports that $Np = 57$ and $N_{s/1p} = 35$ put a tight constraint for embedding the ARMA model in GAP9 when precision is set to `float32`, as this configuration requires $1.2\,\mathrm{MBytes}$. Nevertheless, this hardware constraint is compatible with the majority of civil and industrial facilities [25].

Energy Consumption

Low energy consumption plays a crucial role in a sustainable battery-based system in long-term monitoring. Table 5.5 reports energy consumption for one run of the SysId deploying AR and ARMA model. Noticeably, the energy demanded by ARMA is, on average, double the energy consumed by AR due to the two-step nature of the adopted ARMA algorithm. This is mainly due to the longer execution time of the ARMA models; however, notice that since both models computationally use merely matrix multiplication, they yield similar power consumption of $48.3\,\mathrm{mW}$.

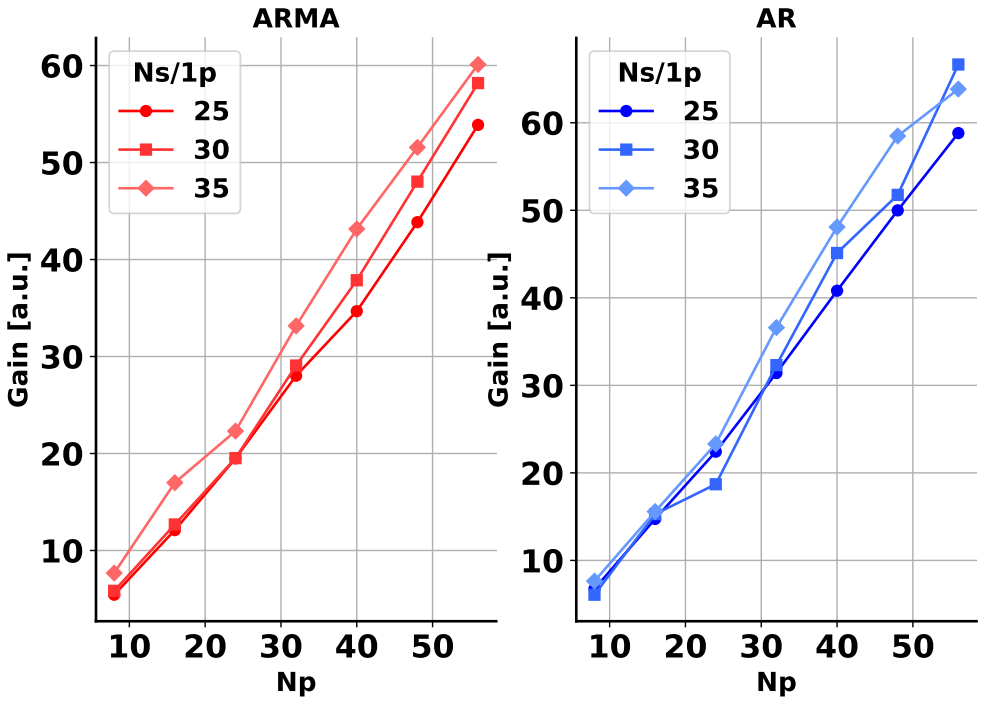


Figure 5.15: Time gain achieved by moving from sequential to parallel implementation.

5.5 Vehicle Classification

5.5.1 Case Study

Bridge Structure

The case study is a roadway bridge in Italy, made of 18 spans, 2 layers, and 583 m long. The structure is a girder bridge in reinforced concrete with an isostatic statical scheme. All the spans have the same length, equal to 20 m, except for the first span, which is 10 m long, and the 10^{th} span, which is 29.5 m long.

SHM Framework

The data acquisition systems deployed for this work are divided into two sections: i) sensor nodes and ii) Weight in Motion (WiM). Sensor nodes are massively installed over the bridge to acquire acceleration data. In contrast, WiM captures metrics like Gross Weight, Velocity, and number of vehicle axles. Fig. 5.16-A shows the block diagram scheme of how the WiM system and viaduct are distanced from each other. Further, Fig. 5.16-B displays the front look of the viaduct where only one lane of the bridge with the installed SHM system is shown.

Sensor Nodes

The SHM system consists of 282 MEMS biaxial clinometers, 142 MEMS triaxial accelerometers, and three gateways that transmit the data to the cloud. Sensor data is collected by the gateways and transmitted to the cloud through an LTE modem embedded in the gateway. Fig. 5.16-C highlights the in-situ sensor node used to acquire data. The MEMS accelerometers are connected via CAN-BUS to the gateway. They are three axes linear accelerometers

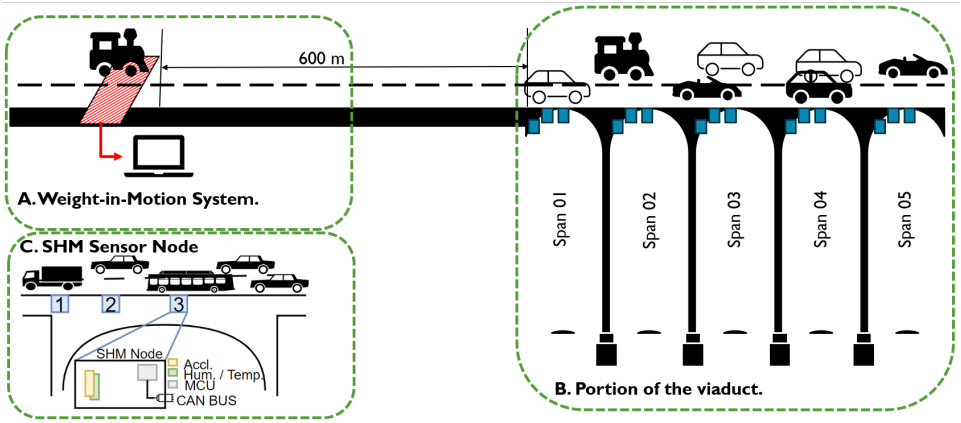


Figure 5.16: The real-life case study used for vehicle classification application. Panel A: The weight-in-motion (WiM) system stores several metrics, such as weight and velocity. Panel B: Block diagram of the 5 spans of the viaduct under study. Panel C: SHM Framework for data acquisition installed under the viaduct.

Day	Time Interval [AM]	Type of dataset
1	1:00 - 6:00	Trainning Set
2	1:00 - 4:00	
3	1:00 - 4:30	
4	1:00 - 8:00	
5	1:00 - 4:30	Validation Set

Table 5.6: Time intervals of each day for the dataset.

with ± 2 g full scale and 100 Hz sample rate. Accelerometers are equally distributed between the 18 spans. On each span, the accelerometers monitor the external two beams, and, for each beam, three sensors were installed at the quarter, the third, and the midspan of the beam.

WiM

The WiM sensors are placed about 600 m before the viaduct in a section without highway exits or parking areas. The WiM measurement system is directly connected to a laptop to store the acquired data. It can provide several features about the traffic on the bridge, including the lane of the detected vehicle, its length, weight and speed, and the number of vehicle axles.

Dataset

The sensor node and WiM acquisition systems captured data for five days over the bridge; hence, our dataset comprises four training days and one validation day. Table 5.7 reports the time interval of each day in the dataset, which is mostly focused on the night since the bridge experiences low traffic volume. Further, we could label the data from four bridge spans in each time interval of Table 5.7. It should be noted that the quantity of samples differs across different intervals due to potential lane changes or velocity reductions made by vehicles.

As a result, vibrations may not be discernible from the accelerometer's perspective and can therefore have minimal impact. Considering the time interval of Table 5.7 for the four spans, four scenarios are constructed. Each span yields a T_i and a V_i $\forall i \in [1, 2, 3, 4]$, corresponding training and validation set of each span, respectively. Further, we can define $T_{tot} = \sum_{i=1}^{4} T_i$, and $V_{tot} = \sum_{i=1}^{4} V_i$, which include all the spans together for training and validation of this work's framework. In this context, we can define four different scenarios for the training and validation set, which are:

- SC 1: a T_i as the training set and a V_i as the validation set
- SC 2: T_{tot} for the training set and V_i as the validation set.
- SC 3: T_{tot} for the training set and V_{tot} as the validation set.
- SC 4: T_{tot} for the training set and V_i as the validation set with redundancy in classification.

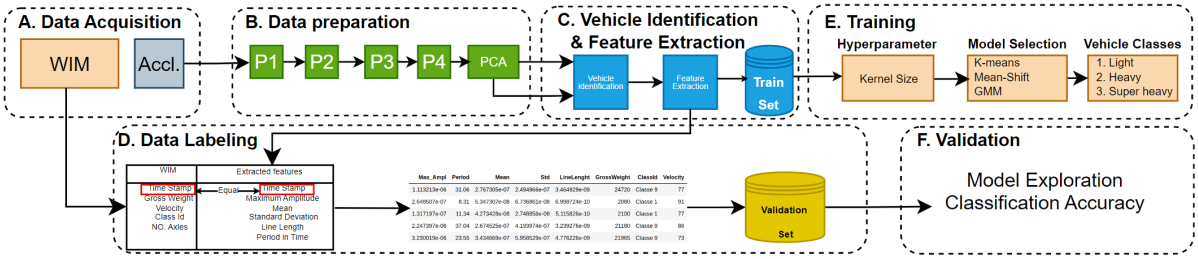


Figure 5.17: The proposed framework of this work: (A) Two data acquisition systems of our system, namely, Accelerometer and Weight-In-Motion device. (B) Data pre-processing chain applied to 2D raw vibration to extract a 1D trace showcases the raw data. P1 to P4 are the L2-norm of the x-z plane, band pass Butterworth filter, overlap windowing, and energy smooth trace, respectively. Finally, the smooth traces are applied to PCA to extract two thresholds for vehicle identification. (C) Vehicle identification and feature extraction. (D) Labeling the extracted features by coinciding with WIM data. (E) Training and (F) Different validation studies.

5.5.2 Methodology: Vehicle Classification in SHM context

This part describes the main contribution of this section, which is a framework to classify the vehicles based on their gross weights. Similar to [18], the raw vibration data are fed to a preprocessing chain to extract a smooth trace that eases identifying vehicle passage over the viaduct. Next, Principal Component Analysis (PCA) is applied to the smooth traces to identify the vehicle's passage. Furthermore, for an individual vehicle's passage, five different features, namely, Maximum Amplitude, Time duration, Standard Deviation (std), Mean, and Line Length, are computed to represent each vehicle. Then, the labeling step is performed to label the extracted features. WiM data are aligned with the extracted features to label the data. Finally, K-means, Mean Shift, and GMMs are deployed to cluster data into three clusters,i.e., Light, Heavy, and Super-heavy classes. Fig. 5.17 illustrates the main framework of this work, distributing it in Data Acquisition (Sec. 5.5.1), Data preprocessing (Sec. 5.5.2), Vehicle identification and Feature extraction (Sec. 5.5.2), Vehicle labeling (Sec. 5.5.2), and vehicle classification (Sec. 5.5.2).

Pre-Processing

The preprocessing stage is split into two primary sections. The initial section receives a 2D plane of raw vibration data, specifically along the $x-z$ axis, and derives informative 1D traces from it. These traces are coupled with Principal Component Analysis (PCA) to calculate two thresholds for boxing a vehicle passage event. Subsequently, a vehicle detection phase is performed utilizing the 1D traces joined with the two thresholds of the PCA analysis. Fig. 5.18 showcases the different stages of the data preprocessing chain. In the following, we will discuss each step of this chain.

L_2 Normaliztion

To combine the information of two bridge axes, i.e., x and z axis, an L_2 normalization is performed to convert 2D information into 1D. L_2 normalization can be extracted as follows:

$$|\cdot|_{L_2} = \sqrt{(x-\overline{x})^2 + (z-\overline{z})^2}$$

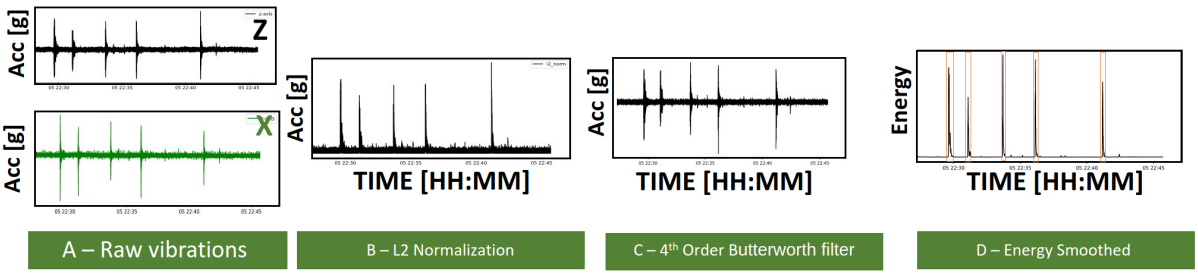


Figure 5.18: The preprocessing chain applied to the raw data: A) Acceleration Raw data of z axis on top and x axis at the bottom, B) Result of the L_2 normalization of $x - z$ axes, C) Result of the 4^{th} order filter applied to the normalized values at the former step. D) Extracted energy values as the signature of each vehicle passage.

Where $\overline{x}$ and $\overline{z}$ stand for the mean of the axis during a reference period of 5 minutes free of peaks. Fig. 5.18-B highlights the difference between the raw data and the L_2 normalization This step is more beneficial for low-energy vibration caused by light vehicles that are non-trivial to identify.

4^{th} order Butterworth filter

Time [Hour]	4	12	18	22
Frequency Mode [Hz]	4.34	4.07	4.27	4.55

Table 5.7: Frequency analysis results of 15 minutes in 4 different time zones of the day

Structures Oscillate in relatively low frequencies in the range of a few Hz. Therefore, a Butterworth filter is applied to the normalized 2D data in order to separate low-frequency signals from high-frequency noise. Moreover, when it comes to a viaduct where the passage of vehicles may intertwine, shortening the damping time is advantageous in detecting all peaks and preventing any overlap among vehicle passages. This measure proves critical for efficient traffic flow management in such structures. Consequently, a 4^{th}-order Butterworth filter is employed to maintain the desired spectral range in the viaduct's scenario,i.e., $0 - 15\,\mathrm{Hz}$. Finally, Table 5.7 reports the spectrum analysis over several vehicle passages over the day, indicating that the natural frequency of the viaduct is placed approximately at $4\,\mathrm{Hz}$.

Energy Extraction

The energy of the filtered vibrations can be computed as follows:

$$E = \sum_{t=0}^{100} S_i^2$$

Where S_i is overlapped shifting windows of 1-second data, notice that we take overlapping windows to avoid data loss. [18] shows that a duration of $1\,\mathrm{second}$ is sufficient to ensure the detection of a vehicle vibration trigger signal.

Energy Smoothing

Finally, we apply exponential smoothing to the computed energies. It aids in decreasing the oscillation amplitude damping, considering the history of the signal. Consider that energetic windows corresponding to the traces with larger amplitude could cause multiple informative vehicle passages to be missed. To mitigate this variability, an energy smoothing technique is employed. This energy filtering can be mathematically formulated as

$$E[t] = \alpha * E[t] + (1 - \alpha) * E[t - 1]$$

with E the energy and α the smoothing factor. In this work, we set $\alpha = 0.7$. Finally, panel D in Fig. 5.18 depicts the final 1-D traces deployed for vehicle identification of our work.

Energy Threshold

According to Fig. 5.18, each peak's damping times differ; hence, it is necessary to customize the identification process for individual vehicle vibration traces. To achieve this objective, the algorithm described in Sec. 5.5.2 has been developed by incorporating two distinct energy levels: a high threshold for initiating and a low threshold for terminating a vehicle passage. To discriminate between an informative window, i.e., vehicle passage, and a non-informative one, i.e., white noise, Alg. 1 is a promising solution. Hence, such a solution is deployed to determine the high threshold value for the vehicle identification algorithm, which is $2.56E-7$ in our case study. In a nutshell, the energy level is computed via Principal Component Analysis (PCA). Alg. 1 presents the algorithm's pseudocode, where the reconstruction error, namely RSNR[2], determines the breaking point of the algorithm. Initially, an infinitesimal threshold is set for window cancellation, which is ascended at each iteration. Thus, it remains more informative at each iteration and has fewer non-informative windows. According to [19], $RSNR = 16dB$ is sufficient to reconstruct the original signal's structure. The high threshold computed for our case is $2.56E - 7$. Further, the low threshold is empirically chosen as 0.1 of the high threshold coinciding with the noise level of the accelerometer sensors deployed for acquisition.

Vehicle Identification & Feature Extraction

Initially, this module details the approach for defining a bounding box around each vehicle and subsequently presents the statistically extracted features for the classification of vehicles. These extracted characteristics are established attributes in time-series data classification domains such as EEG [5] and vibration signals preprocessing [18].

Vehicle Identification

The literature [18] suggests using only one threshold for triggering and ending the vehicle passage event; however, we decided to deploy two thresholds to capture the whole passage time of the vehicle. Further, given the unique structure of the viaduct understudy, having one threshold would lead to very small windows for light vehicle passage, which would cause a loss of information for the given event. While the high threshold initiates the vehicle passage event, the low threshold is set at the noise level ending a vehicle passage event. Empirically, it is determined as an order of magnitude less than the high threshold. This ensures that any energy

[2]Reconstructed Signal to Noise Ratio = $20 \log_{10} \left(\frac{\|\mathbf{x}\|_2}{\|\mathbf{x}-\hat{\mathbf{x}}\|_2} \right)$

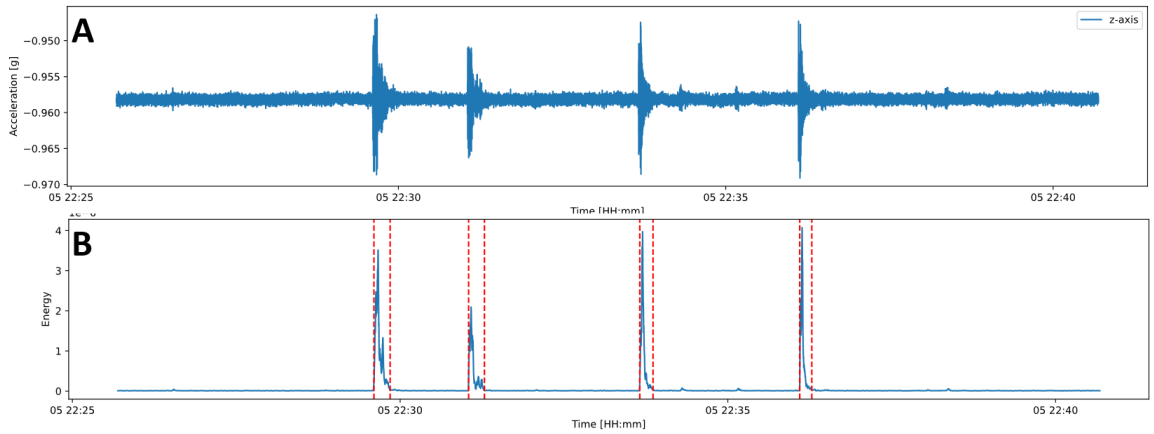


Figure 5.19: From raw data to the boxed version of each vehicle passage. Panel A) 15 minutes of raw data B) Boxed vehicles.

level below this threshold is considered noise and not part of a vehicle passage event. Finally, Fig. 5.18-D showcases the result of vehicle boxing for 45 minutes of data. As mentioned, since the damping time of a vehicle passage event is more than the duration of one window, i.e., 1 second, we define two thresholds to box a vehicle passage. The high threshold is set for initiating, and the low threshold is for terminating a vehicle passage event. Sec. 5.5.2 describes the algorithm to compute the high threshold.

Let us define an informative window of a vehicle passage as V and a non-informative window as W. Thus, two consecutive windows can be $V-W$, $W-W$, $W-V$, and $V-V$. Initially, the tracking flag is set high, indicating we are waiting to trigger a vehicle passage event. Monitoring two consecutive windows of 1 second, whenever the algorithm experiences a $W-V$ case, it considers it a vehicle passage and sets the flag to low to search for the stopping point. Similar to the triggering event, when we have a consecutive sequence of $V-W$, the algorithm sets it as the end point of the vehicle passage. This algorithm is repeated for the whole dataset to extract all the vehicle passages. Fig. 5.19 shows 15 minutes of the raw data indicating the transition from the raw vibration to energy-smooth signals, where each vehicle passage event is boxed. Finally, when the borders of the vehicle are defined, we extract the features described in Sec. 5.5.2 for each vehicle passage event.

Feature Extraction

The algorithm described in Sec. 5.5.2 results in different windows of time; thus, to characterize all vehicles with the same basis, we extracted features vastly deployed in the literature [cite a few]. Four macro statistical features for each vehicle passage event are considered: Maximum Amplitude, Mean, Standard Deviation, and Line Length [41]. Let's consider $X_k = [x_0, x_1, ..., x_{N-1}]$ a time series array corresponding to the vehicle event k, where N is the number of samples in the box of the detected vehicle. The extracted features can be defined as follows:

1. Maximum Amplitude = $max(X_k)$
2. Time = $T@(N-1) - T@(0)$
3. Mean = $\frac{1}{N} * \sum_{i=0}^{N-1} x_i$
4. Standard Deviation = $\sqrt{\sum_{i=0}^{N-1} \frac{(x_i-\mu)^2}{N}}$

5. Line Length = $\frac{1}{N} * \sum_{i=1}^{N-1} |x_i - x_{i-1}|$

In Sec. 5.6.1, we show that only four features are useful for vehicle classification with our dataset.

Data Labelling

To assess the performance of vibration-based systems and obtain the accuracy of classifying a vehicle, vision-based systems are deployed at the field to label each vehicle passage. However, data extracted from vision-based systems help allocate the vehicle type, leaving out the gross weight and velocity of an individual vehicle passage, which are essential features of bridge integrity. A data labeling step is performed to establish a link between the extracted features and the gross weights of a given vehicle to assign WiM metrics to extracted features from the SHM system.

We employed timestamps generated from the WiM and SHM systems to establish a link between the two sets of data. Given that the WiM system is positioned within 600 meters of the bridge, we deemed it appropriate to consider a time interval of 2 minutes between vehicle event times and corresponding WiM data. Due to the bridge's massive weight, certain light vehicles (less than $2\,\mathrm{tons}$) were only captured by the WiM system rather than accelerometers. Consequently, more vehicle passage events are recorded by the former compared to the latter. In each instance of labeling vehicle events in time slots, we connected heavy vehicles with passages identified by accelerometers. Furthermore, when there was more than one occurrence per minute, k number of heavy cars were assigned sequentially with k different vehicle events as captured through accelerometers' readings. Sec. 5.6.1 details the result of the link between the WiM and SHM system features. Alg. 2 showcases the labeling algorithm's pseudocode, where the number of the captured vehicles by the SHM system, i.e., accelerometer, determines the number of the vehicles we are linking between the accelerometers and WiM system's metrics.

Algorithm 2 Data Labelling

Input: $\mathbf{Data}_{WiM}$, $\mathbf{Data}_{Acc.}$
for ts $\in$ Unique Ts of $\mathrm{Data}_{Acc.}$ **do**
    $\mathrm{Accl}_{ts} \leftarrow$ all vehicles $\in$ **ts** of $\mathrm{Data}_{Acc.}$
    $\mathrm{WiM}_{ts} \leftarrow$ all vehicles $\in$ [**ts-2**, **ts**] of Data_{WiM}
    $\mathrm{Accl}_{Num} \leftarrow$ Number of Vehicles in Accl_{ts}
    $\mathrm{Vehicle}_{WiM} \leftarrow$ Append(heaviest $\in \mathrm{Wim}_{ts}$)$\times \mathrm{Accl}_{Num}$
    $\mathrm{Data}_{WiM} \leftarrow$ Remove($\mathrm{Vehicle}_{WiM} \in \mathrm{Data}_{WiM}$)
end for
Output: $\mathrm{Vehicle}_{WiM}$, $\mathrm{Data}_{Acc.}$

Vehicle Classification

The features obtained from the accelerometer sensors are utilized in unsupervised classification algorithms to classify vehicles into three categories. The classification process is based on a 4D space consisting of the aforementioned extracted features: Maximum Amplitude, Mean, Std, and Line Length for each vehicle passage event. In this study, we have categorized vehicles into three macro clusters according to their gross weight: light class (less than $10\,\mathrm{tons}$), heavy class (between 10 to $30\,\mathrm{tons}$), and super-heavy class (above $30\,\mathrm{tons}$). This categorization has

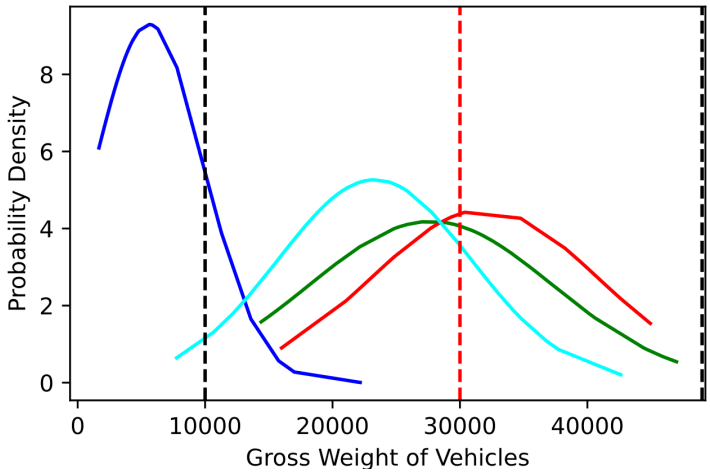


Figure 5.20: Distribution of clusters for a portion of SHM dataset showing 4 micro classes in light blue, dark blue, green, and red and thresholds to classify vehicles into three macro classes, i.e., Light, heavy, and super heavy.

been established due to the bridge maintenance constraint that requires alarming in case of massive dynamic weight over the bridge, which may result in its collapse or severe damage. The classification is done in two steps. The first is unsupervised classification, followed by a weight assigned to each vehicle passage. In the phase of unsupervised classification, three methods (K-means, mean shift, and GMMs) are utilized to cluster vehicles, as detailed in Sec. 3.2. K-means and GMMs necessitate a predetermined number of clusters for grouping input data. Conversely, the mean shift algorithm identifies the required number of clusters based on its kernel size. The intuitive choice of classes to start for K-means and GMMs is three because a vehicle passage would be in one of three categories: light, heavy, and super-heavy. However, we explore a range between 3-8 microclusters to obtain an optimal number for K-means and GMMs while keeping macro classes limited up to 3 in our final classifier. Meanwhile, we evaluate various neighborhood bandwidth sizes $[0.1 - 0.25]$ through the mean shift algorithm, automatically determining optimal numbers for clusters by adjusting kernel size accordingly. For the weight assignment stage, we deploy statistical features of training sets to decide the cluster of each class. Fig. 5.20 shows the pdf of the microclusters in the training set with 6 centroids and the macro clusters' boundaries. In order to come back to 3 macro clusters from the higher micro clusters, we considered the statistical metrics of the training set. If the mean of the Gross weight of a vehicle in a micro class fits within the boundary of a macro class,i.e., $10\,\text{tons}$, $30\,\text{tons}$, and above $30\,\text{tons}$, we consider that micro class as light, heavy, or super-heavy, respectively.

5.6 Results: Vehicle Classifications

This section is dedicated to validating the best features, classifiers, and labeling metrics for clustering vehicles into the three classes with the pipeline presented in Sec. 5.5.2.

In the first section, we justify the choices for the extracted features in the time domain to select a set of statistical features for the classification problem. Next, we report the result of the labeling procedure described in Sec. 5.5.2. Then, we explore different vehicle classification methods to find our framework's optimum configuration (Fig. 5.17).

In the second part of this chapter, we examine the hyperparameters of the unsupervised

vehicle classification pipeline, namely, the number of classification clusters, exploring different spans and the final classification described in Sec. 5.5.2 to explore all the aspects of the system.

5.6.1 Feature Extraction & Data Labelling

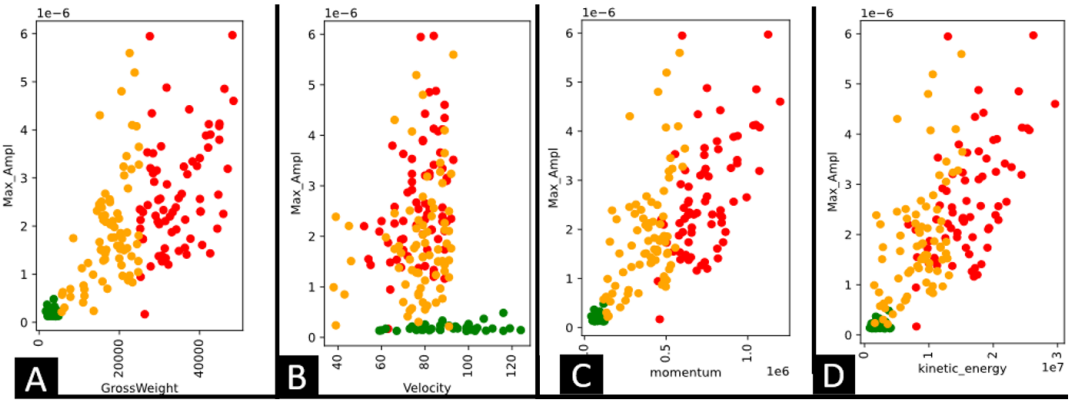


Figure 5.21: Four different vehicle identicator metrics, namely Grossweight (panel A), Velocity (panel B), Momentum (panel C), and Kinetic energy(panel D) vs. the maximum amplitude of each vehicle event. In green the light class, in orange the heavy class, and in red super heavy class vehicles.

The standard European laws for vehicle classification are based on the number of axles of a vehicle, grouping them into 13 different classes. However, these laws do not consider the gross weight and velocity of a vehicle passage, which could affect the viaduct infrastructure's safety, maintenance, and durability. Considering the WiM dataset, Fig. 5.22 depicts the statistics of each class in terms of gross weight, which is a critical metric for bridge dynamic weight load. Fig. 5.22 presents that classes with more axles are not necessarily the heaviest and most energetic vehicles to imperil the viaduct's integrity. Thus, a new metric must be deployed to classify vehicles as harmful or harmless to the bridge. Fig. 5.21 showcases one of the extracted features, namely the maximum amplitude of each vehicle passage for four different energetic metrics, i.e., Gross weight, Velocity, Momentum, and Kinetic energy, correlating the extracted feature to each metric over the second span of the bridge. For instance, lighter vehicles tend to have lower maximum amplitudes, while heavier ones possess more energy, resulting in higher maximum amplitudes. Fig. 5.21 shows that gross weight and maximum amplitude increase together as the light vehicles have less maximum amplitude and heavier vehicles contain more energy; hence, they tend to have higher maximum amplitude. In contrast, an examination of velocity reveals that the maximum amplitude of vehicles exploring the whole space and clustering is not feasible for vehicle classification in terms of the harmfulness to bridge infrastructure. Further, an evident result of velocity is that vehicles with higher velocities are the light ones with a small maximum amplitude. Further, Momentum and Kinetic energy mimic the Gross weight metric, indicating a similar or less interesting correlation to the maximum amplitude; thus, we deploy the gross weight of each vehicle to label them, which is the critical energetic metric to cluster each vehicle passage. As a result, in this application, we deploy gross weight as a new metric instead of the number of axles to monitor the dynamic motion over the bridge.

As described in Sec. 5.5, we extracted 5 statistical features for an individual vehicle trace. Fig. 5.23 shows a positive correlation between the four features, i.e., Maximum Amplitude,

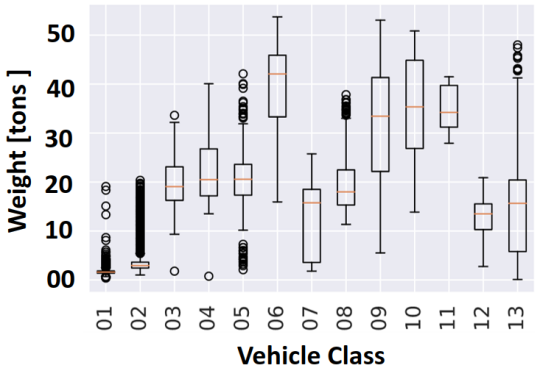


Figure 5.22: Gross weight distribution of EU - laws

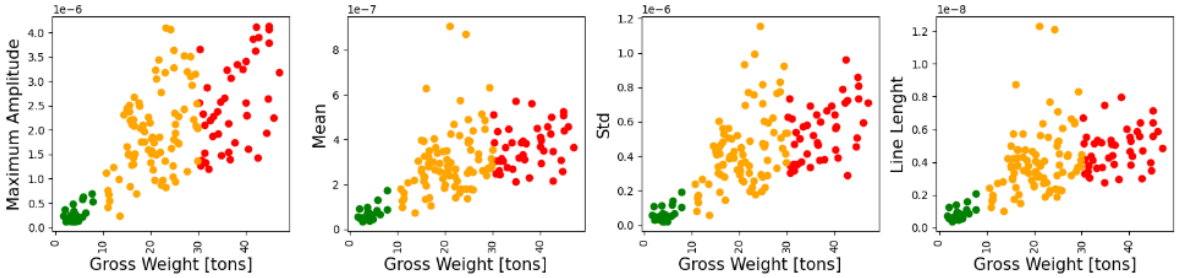
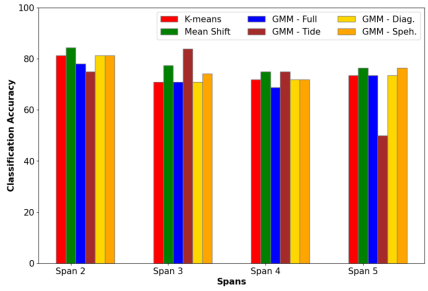
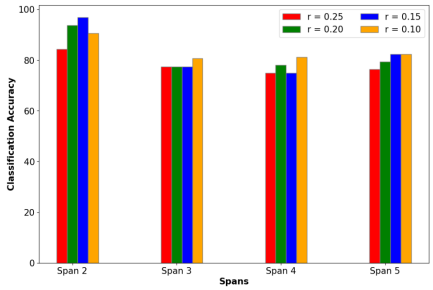


Figure 5.23: Distribution of the extracted features over the gross weight. In green, the light class; in orange, the heavy class; and in red, super heavy class vehicles.

Mean, standard deviation, and line length, with the gross weight. However, the time duration of each vehicle passage does not contain any information in distinguishing vehicles as light vehicles could cause damping for a long period and gradually reach the noise level of the sensors; hence, we discard the time duration for the unsupervised classification.



(a) Classification accuracy of the three unsupervised clustering models over different spans of the viaduct.



(b) Effect of modifying mean-shift kernel size in clustering.

Figure 5.24: Classification Accuracy for the two scenarios of the unsupervised classification of vehicles

5.6.2 Algorithm Exploration

We have investigated three unsupervised clustering methods for categorizing vehicles into three distinct classes: light, heavy, and super-heavy. Deploying scenario 01 in Sec. 5.5.1 with all the datasets' spans, the clustering results show promising classification based on the extracted features. Fig. 5.24a shows that classification accuracy differs from $50\,\%$ worse case to $84.37\,\%$, the best case considering only 3 clusters. Further, the mean shift method surpasses the other classification approaches regarding classification accuracy with $84.37\,\%$ of classification accuracy. Furthermore, the mean shift algorithm exhibits superior robustness and adaptability to diverse data distributions compared to other methods. Notice that while the span moves from 02 to 05, there is a fluctuation of up to $9\,\%$ for the mean shift, whereas other classification algorithms display variations ranging between $10\,\%$ to $35\,\%$, changing the training set from one span to another. Finally, Fig. 5.24a depicts that GMMs share similarities with mean shift and K-means, making them highly comparable in classification accuracy. One can achieve similar results to either of the aforementioned methods by simply altering the variance parameter.

In conclusion, the mean shift method is preferable for analyzing and clustering vehicles based on their impact on bridge infrastructure.

Mean Shift Kernel Sweep

As mentioned, we labeled four spans of the bridge in our dataset. Further, since the mean shift is the promising method for classifying the vehicles, we delved into mean shift by performing an exploration over the kernel size of the mean shift, sweeping it from 0.1 to 0.25 According to Fig. 5.24b, the kernel size of 0.1 is the best size for the mean shift algorithm varying the spans of the bridge with a classification accuracy of above $80\,\%$. However, span number 2 shows a noticeable performance with a kernel size of 0.15 , resulting in a classification accuracy of $96\,\%$, which is due to the training and validation set distributions of that span. Further, the hyperparameter exploration over the mean-shift classifier indicates the sensitivity of each span and chosen parameters in the pipeline; hence, deploying a robust classification requires an in-depth knowledge of the structure.

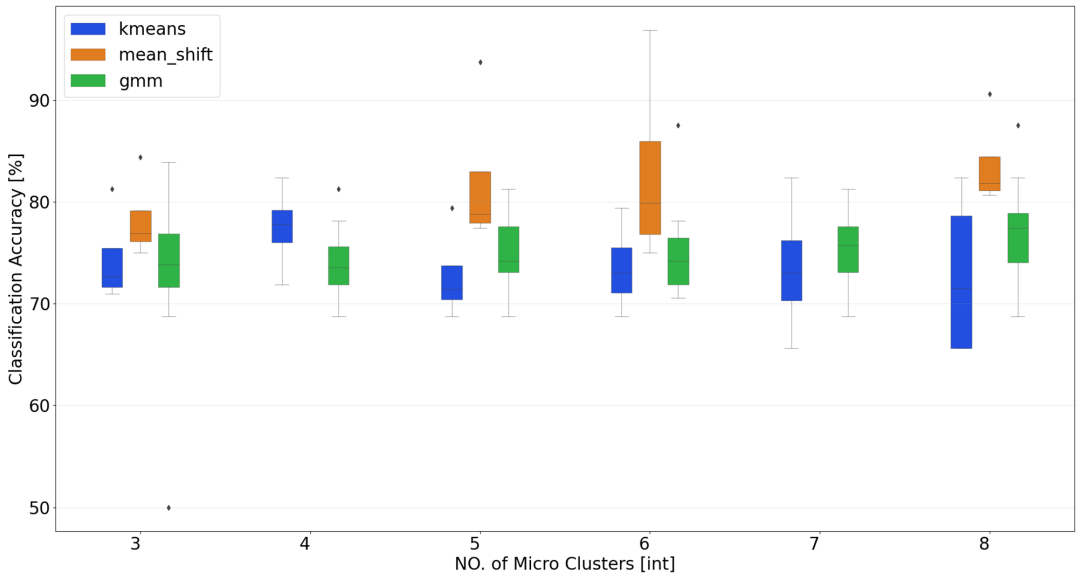


Figure 5.25: Result of varying clusters of clustering methods.

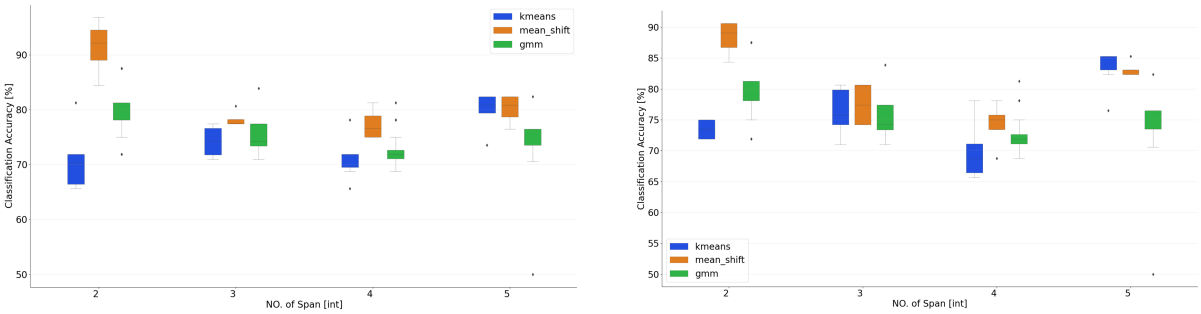
5.6.3 Hyperparameter Exploration

In this section, we examine the hyperparameters of the unsupervised vehicle classification pipeline, namely, the number of classification clusters, exploring different spans and the final classification described in Sec. 5.5.2 to explore all the aspects of the system.

Micro-Clustering

To achieve better classification accuracy, we increased the number of clusters for methods like k-means clustering and GMMs with the intuition of enhancing the trained model. Additionally, exploring different bandwidths for the mean-shift algorithm could also contribute to improving classification accuracy. Thus, this section explored a range of clusters from 3 to 8 while keeping the final classification classes as 3. Note that we are still considering scenario 1 for training and validating our methods while increasing each clustering method's clusters.

Fig. 5.25 presents the impact of increasing the number of microclusters on classification accuracy, highlighting a significant improvement in most cases with an increase of up to $12\,\%$. Nonetheless, it is noteworthy that this relationship is not monotonic due to variations observed while we increase the number of clusters. For instance, mean-shift reaches an accuracy of $96\,\%$ with a bandwidth of 0.15 for span 2 and then reaches the best accuracy with a bandwidth of 0.1 for the other spans with an accuracy of $80.64\,\%$, $81.25\,\%$, and $82.35\,\%$, respectively. Further, considering K-means clustering, the accuracy reaches its highest value with 3 clusters for span 2; however, this method behaves better as we increase the number of clusters in the other spans. The reason for these fluctuations is our limited dataset and the different distribution of the data during the different spans, which might not always produce the most accurate classification with increasing clusters. Considering GMMs, having 8 clusters leads to the highest classification accuracy across the majority of the spans. In conclusion, it is essential to note that increasing the number of clusters may not always result in improved classification accuracy. Therefore, it is essential to carefully consider the optimal number of clusters for a given dataset before applying any clustering algorithm.



(a) Span Exploration - SC01

(b) Span Exploration - SC02

Figure 5.26: Span Exploration Results in the two scenarios described in Sec. 5.5.1

Span exploration

As described in Sec. 5.5.2 and Sec. 5.5.1, we labeled four bridge spans to make the dataset for vehicle classification. Further, by deploying these spans, we end up having four scenarios for training and validating the proposed pipeline. Thus, we described different training and validation scenarios with these four spans.

Fig. 5.26b highlights the result of scenarios 2 and Table 5.8 reports for scenario 3 described in Sec. 5.5.1, where we combine all the spans together to define the training set. Notice that we perform such an idea to increase training samples in the given 4D space with the intuition to enhance the trained model. Considering Fig. 5.26, we notice that increasing the training dataset size impacts the k-means progressively, where we observe an increase in span 03 and 04 by $3\,\%$ and $7\,\%$, respectively, whereas it causes a decrease of $6\,\%$ for span 02, considering the best number of micro-clusters for each model. Despite K-means, an increase in training set size is ineffective for the mean shift algorithm, where we face a drop of accuracy in spans 02 and 04 by $6\,\%$ and $3\,\%$, respectively, while the classification accuracy faces an insignificant alteration for the other two spans.

Further, Table 5.8, which reports the result of merging the test sets together to form a unique test set, and it can be seen that the models lose their robustness to larger test sets with different samples where $7.76\,\%$ and $5.45\,\%$ drop in accuracy observed by k-means and mean-shift, respectively. In conclusion, while increasing the training set could benefit training a

K - Means	NO. micro Clusters	SC 03	SC 04 Equal	SC 04 Weighted
	3	77.51	75.00	79.16
	4	74.41	83.33	75
	5	76.74	79.16	75
	6	75.96	75	70.83
	7	75.19	75	70.83
	8	75.19	75	70.83
Mean Shift	NO. Quantile			
	0.25	77.51	79.16	79.16
	0.2	79.84	83.33	83.33
	0.15	78.29	91.66	91.66
	0.1	78.29	87.5	87.5

Table 5.8: Comparison between k-means and mean-shift for considering all the spans together for the training set while sweeping over different micro-clusters

Scenario 02	Spans	K-Means	Mean-Shift
	02	75.00	90.62
	03	80.64	80.64
	04	78.12	78.12
	05	85.29	85.29
Scenario 03	-	77.53	79.84
Scenario 04	Voting Paradigm		
	Equal	83.33	91.66
	Weighted	79.16	91.66

Table 5.9: Comparison between the best case Classification Accuracy for different scenarios

robust model, considering an individual node to classify a vehicle passage is more promising than taking a cluster of nodes to classify a vehicle passage.

Redundant Vs. Non-redundant clustering

To combine the outcome of each span for the classification of a vehicle, we adopted two voting paradigms, namely equal weight and weighted. We described in Sec. 5.6.3 that although a larger training set affects the robustness of the trained model, a combination of spans as a unique classifier drops the classification accuracy. Thus, we decided to explore another direction in utilizing several sensor outcomes to make a unique decision. In the equal-weighted scenario, all the spans hold the same weight. i.e., $25\,\%$, for the final decision of the vehicle's class. Regarding an equal vote for a vehicle passage, we refer to the decision of span 02 since it is the closest span to our golden model, i.e., the WIM system. In the weighted voting paradigm, we give more merit to the spans closer to the golden model, i.e., span 02, by assigning it a weight of $50\,\%$, while the remaining spans have $25\,\%$, $12.5\,\%$, and $12.5\,\%$, respectively, for the final decision of a vehicle's class. Notice that this allocation of weights is based on the assumption that spans closer to the golden model would have a higher level of accuracy. However, when there is no WIM present in the scenario, weight can be assigned by taking into account other relevant factors, such as the precision of individual sensors or the distance of each span from the entrance/exit of the roadway. Table 5.8 displays the classification accuracy of the two voting paradigms and training and validation scenario number 4. Comparing the two voting paradigms, equal voting surpasses weighted one for the k-means algorithm by $4.17\,\%$, while the voting topology does not affect the mean shift. Further, Considering the best cases of scenarios 3 and 4, where we train and validate on spans together with different paradigms, we observe that voting paradigms effectively increase the classification accuracy by $5.80\,\%$ and $1.03\,\%$ for k-means and mean shift, respectively.

Despite that, varying the perspective toward a cluster of spans to classify a vehicle passage improves the classification accuracy, comparing the results of single node classification, i.e., scenario 2, and a cluster of nodes classification. i.e., scenarios 3 and 4, Table 5.9 reports that no absolute approach surpasses the other one. However, considering K-means, we face that single-node classification excels by $1.96\,\%$, whereas in mean-shift, the story is reversed, and voting paradigms report better classification accuracy with a minimal increase of $1.00\,\%$ compared to the single-node evaluation. Thus, scenarios 02 and 04 are selected based on the application and the available dataset.

CHAPTER

6

CONCLUSION

This work presented a paradigm shift from cloud to edge computing for SHM systems on real-life case studies utilizing homogenous big data sets featuring MEMS-based accelerations data acquired from long-span viaducts.

In the node design phase, we provided an experimental evaluation of analog and digital MEMS accelerometers for structural health monitoring applications to justify replacing the traditional costly piezoelectric accelerometer with the latest generation of MEMS accelerometer. We discussed the two criteria by which sensors are evaluated, which are the time and frequency domain. Noise analysis of the sensors shows that the piezoelectric sensor is the less noisy device, with only $10.28\mu g/\sqrt{(}Hz)$. While in-lab experiments show that MEMS low-cost sensors mimic piezo sensors for modal frequency analysis, we show in the real-life experiment that MEMS-based accelerations diverge maximum by 1.6% error.

In the first signal processing application, we proposed an efficient damage detection solution at the edge, simultaneously reducing network traffic and energy consumption while anomaly detection accuracy is not adversely altered compared to cloud-based systems. This application initially discussed a new damage detection pipeline comprising a pre-processing step, an anomaly detection algorithm, and a post-processing step. Comparing PCA and two different autoencoders, we show that PCA outperforms the other two methods by approximately $30\,\%$ and $48\,\%$ on the SHM dataset collected on a real-standing Italian bridge. We show that by tuning the hyperparameters of the proposed pipeline, we further improve the accuracy in detecting anomalies by $20\,\%$. Additionally, we demonstrate embedding our tuned pipeline on a tiny low-power device, moving the damage detection to the edge of the network. By doing so, we reduce the data traffic by a factor of $\approx 8 \cdot 10^5\times$, from $780\,\mathrm{KBytes/hour}$ to $10\,\mathrm{Bytes/hour}$, compared to a cloud-based anomaly detection solution. Further, we reduce the power computation for the node by $5\times$.

Secondly, we studied an unconventional data reduction algorithm, namely system identification, to reduce its latency and energy consumption by moving from sequential to parallel implementation. This application presented a parallelized implementation of a vibration compression algorithm based on system identification, which is necessary to decrease the volume

of data transmitted to the cloud, using an ultra-low-power multi-core platform, namely GAP9, as the computing platform. We showed that by shifting the paradigm from sequential to parallel implementation, an improvement up to $\approx 60\times$ could be attained in terms of execution time, which is fundamental to avoid the long latency of the sequential version and enables the in-field deployment of the SysId algorithm for streaming vibration data processing. Further, we showed that the most power-hungry deployment of the model consumes $48.3\,\mathrm{mW}$ per each run of the algorithm, making it suitable for long-term self-sustainable battery-based monitoring systems.

Lastly, we proposed an unsupervised clustering framework to classify vehicles into three classes of light, heavy, and super heavy deploying accelerometer data acquired from an 18-span viaduct in the North of Italy. This application discussed a new vehicle classification pipeline divided into a pre-processing step, a feature extraction step, and an unsupervised classification step. we compared 3 state-of-the-art algorithms named K-means, mean shift, and GMMs for clustering the vehicles, showing that mean shift is more robust along all the spans with a classification accuracy over 77% in the worst case and 84.37% in the best-case scenario. Moreover, we performed a hyperparameter exploration over the radius of the mean shift kernel, and we showed that the best result is with $0.15\,\mathrm{cases}$ since it reaches a classification accuracy of 96.87% with span number 2. Further, in a more detailed comparison, we compared the performance of the K-means, mean shift, and GMMs algorithm varying their hyperparameters, such as the number of clusters and the bandwidth, showing that the mean shift increases its performances from 6% up to 10%.

BIBLIOGRAPHY

[1] O. Abdeljaber, O. Avci, S. Kiranyaz, M. Gabbouj, and D. J. Inman. Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks. Journal of Sound and Vibration, 388:154–170, 2017.

[2] 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, 19(3):693–735, 2020.

[3] C. Acar and A. M. Shkel. Experimental evaluation and comparative analysis of commercial variable-capacitance mems accelerometers. Journal of micromechanics and microengineering, 13(5):634, 2003.

[4] E. Akintunde, S. E. Azam, A. Rageh, and D. Linzell. Full scale bridge damage detection using sparse sensor networks, principal component analysis, and novelty detection. In Proceedings, volume 42. MDPI, 2019.

[5] A. S. Al-Fahoum and A. A. Al-Fraihat. Methods of eeg signal features extraction using linear analysis in frequency and time-frequency domains. International Scholarly Research Notices, 2014, 2014.

[6] A. Alamri, W. S. Ansari, M. M. Hassan, M. S. Hossain, A. Alelaiwi, and M. A. Hossain. A survey on sensor-cloud: architecture, applications, and approaches. International Journal of Distributed Sensor Networks, 9(2):917923, 2013.

[7] A. Albarbar, A. Badri, J. K. Sinha, and A. Starr. Performance evaluation of mems accelerometers. Measurement, 42(5):790–795, 2009.

[8] A. Albarbar, A. Badri, J. K. Sinha, and A. Starr. Performance evaluation of mems accelerometers. Measurement, 2009.

[9] S. S. Arslan, R. Jurdak, J. Jelitto, and B. Krishnamachari. Advancements in distributed ledger technology for internet of things, 2020.

[10] D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics.

[11] O. Avci, O. Abdeljaber, S. Kiranyaz, B. Boashash, H. Sodano, and D. J. Inman. Efficiency validation of one dimensional convolutional neural networks for structural damage detection using a shm benchmark data. In Proc. 25th Int. Conf. Sound Vib.(ICSV), pages 4600–4607, 2018.

[12] M. R. Azim and M. Gül. Data-driven damage identification technique for steel truss railroad bridges utilizing principal component analysis of strain response. Structure and Infrastructure Engineering, 17(8):1019–1035, 2021.

[13] G. Ballard, J. Demmel, L. Grigori, M. Jacquelin, H. D. Nguyen, and E. Solomonik. Reconstructing householder vectors from tall-skinny qr. In 2014 IEEE 28th International Parallel and Distributed Processing Symposium, pages 1159–1170. IEEE, 2014.

[14] M. Ballerini, T. Polonelli, D. Brunelli, M. Magno, and L. Benini. Nb-iot versus lorawan: An experimental evaluation for industrial applications. IEEE Transactions on Industrial Informatics, 16(12):7802–7811, 2020.

[15] F. Barchi, L. Zanatta, E. Parisi, A. Burrello, D. Brunelli, A. Bartolini, and A. Acquaviva. Spiking neural network-based near-sensor computing for damage detection in structural health monitoring. Future Internet, 2021.

[16] C. Bedon, E. Bergamo, M. Izzi, and S. Noè. Prototyping and validation of mems accelerometers for structural health monitoring—the case study of the pietratagliata cable-stayed bridge. Journal of Sensor and Actuator Networks, 7(3):30, 2018.

[17] C. M. Bishop and N. M. Nasrabadi. Pattern recognition and machine learning, volume 4. Springer, 2006.

[18] A. Burrello, D. Brunelli, M. Malavisi, and L. Benini. Enhancing structural health monitoring with vehicle identification and tracking. In 2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), pages 1–6. IEEE, 2020.

[19] A. Burrello, A. Marchioni, D. Brunelli, S. Benatti, M. Mangia, and L. Benini. Embedded streaming principal components analysis for network load reduction in structural health monitoring. IEEE Internet of Things journal, 8(6):4433–4447, 2020.

[20] A. Burrello, G. Zara, L. Benini, D. Brunelli, E. Macii, M. Poncino, and D. J. Pagliari. Traffic load estimation from structural health monitoring sensors using supervised learning. Sustainable Computing: Informatics and Systems, 35:100704, 2022.

[21] E. P. Carden and J. M. Brownjohn. Arma modelled time-series classification for structural health monitoring of civil infrastructure. Mechanical systems and signal processing, 22(2):295–314, 2008.

[22] G. Carvalho, B. Cabral, V. Pereira, and J. Bernardino. Edge computing: current trends, research challenges and future directions. Computing, 103:993–1023, 2021.

[23] P. E. E. R. Center. Home page, 1998.

[24] H.-F. Chang and T.-K. Lin. Real-time structural health monitoring system using internet of things and cloud computing. arXiv preprint arXiv:1901.00670, 2019.

[25] E. N. Chatzi and C. Papadimitriou. Identification methods for structural health monitoring, volume 567. Springer, 2016.

[26] Q. Chen, J. Cao, and Y. Xia. Physics-enhanced pca for data compression in edge devices. IEEE Transactions on Green Communications and Networking, 6(3):1624–1634, 2022.

[27] Y. Cheng. Mean shift, mode seeking, and clustering. IEEE transactions on pattern analysis and machine intelligence, 17(8):790–799, 1995.

[28] P. Croce. Impact of road traffic tendency in europe on fatigue assessment of bridges. Applied Sciences, 10(4):1389, 2020.

[29] I. Crossbow Technology. MICAz datasheet.

[30] Z. Cui, F. Li, and W. Zhang. Bat algorithm with principal component analysis. International Journal of Machine Learning and Cybernetics, 10(3):603–622, 2019.

[31] H. V. Dang, H. Tran-Ngoc, T. V. Nguyen, T. Bui-Tien, G. De Roeck, and H. X. Nguyen. Data-driven structural health monitoring using feature fusion and hybrid deep learning. IEEE Transactions on Automation Science and Engineering, 18(4):2087–2103, 2020.

[32] F. Di Nuzzo, D. Brunelli, T. Polonelli, and L. Benini. Structural health monitoring system with narrowband iot and mems sensors. IEEE Sensors Journal, 21(14):16371–16380, 2021.

[33] F. Di Nuzzo, D. Brunelli, T. Polonelli, and L. Benini. Structural health monitoring system with narrowband iot and mems sensors. IEEE Sensors Journal, 2021.

[34] H. Dong, X. Wang, C. Zhang, R. He, L. Jia, and Y. Qin. Improved robust vehicle detection and identification based on single magnetic sensor. Ieee Access, 6:5247–5255, 2018.

[35] I. L. Dos Santos, L. Pirmez, É. T. Lemos, F. C. Delicato, L. A. V. Pinto, J. N. De Souza, and A. Y. Zomaya. A localized algorithm for structural health monitoring using wireless sensor networks. Information Fusion, 15:114–129, 2014.

[36] K. Dragos and K. Smarsly. A comparative review of wireless sensor nodes for structural health monitoring. In Proc. of the 7th International Conference on Structural Health Monitoring of Intelligent Infrastructure. Turin, Italy, volume 1, page 2015, 2015.

[37] S. J. Dyke, D. Bernal, J. Beck, and C. Ventura. Experimental phase ii of the structural health monitoring benchmark problem. In Proceedings of the 16th ASCE engineering mechanics conference, 2003.

[38] A. D'Alessandro, S. Scudero, and G. Vitale. A review of the capacitive mems for seismology. Sensors, 19(14):3093, 2019.

[39] A. Entezami, H. Sarmadi, B. Behkamal, and S. Mariani. Big data analytics and structural health monitoring: A statistical pattern recognition-based approach. Sensors, 20(8):2328, 2020.

[40] A. Entezami, H. Shariatmadar, and S. Mariani. Fast unsupervised learning methods for structural health monitoring with large vibration data from dense sensor networks. Structural Health Monitoring, 19(6):1685–1710, 2020.

[41] R. Esteller, J. Echauz, T. Tcheng, B. Litt, and B. Pless. Line length: an efficient feature for seizure onset detection. In 2001 Conference Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, volume 2, pages 1707–1710. IEEE, 2001.

[42] C. R. Farrar and K. Worden. Structural health monitoring: a machine learning perspective. John Wiley & Sons, 2012.

[43] E. Figueiredo and J. Brownjohn. Three decades of statistical pattern recognition paradigm for shm of bridges. Structural Health Monitoring, 21(6):3018–3054, 2022.

[44] P. F. Giordano, S. Quqa, and M. P. Limongelli. The value of monitoring a structural health monitoring system. Structural Safety, 100:102280, 2023.

[45] A. Girolami, D. Brunelli, and L. Benini. Low-cost and distributed health monitoring system for critical buildings. In 2017 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems (EESMS), pages 1–6. IEEE, 2017.

[46] A. Girolami, D. Brunelli, and L. Benini. Low-cost and distributed health monitoring system for critical buildings. In 2017 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems (EESMS), pages 1–6, 2017.

[47] Y. Goi and C.-W. Kim. Damage detection of a truss bridge utilizing a damage indicator from multivariate autoregressive model. Journal of Civil Structural Health Monitoring, 7:153–162, 2017.

[48] M. Gul and F. N. Catbas. Damage assessment with ambient vibration data using a novel time series analysis methodology. Journal of Structural Engineering, 137(12):1518–1526, 2011.

[49] Y. Guo, B. Li, M. D. Christie, Z. Li, M. A. Sotelo, Y. Ma, D. Liu, and Z. Li. Hybrid dynamic traffic model for freeway flow analysis using a switched reduced-order unknown-input state observer. Sensors, 20(6):1609, 2020.

[50] R. Hassan, F. Qamar, M. K. Hasan, A. H. M. Aman, and A. S. Ahmed. Internet of things and its applications: A comprehensive survey. Symmetry, 12(10):1674, 2020.

[51] T. Haugen, J. R. Levy, E. Aakre, and M. E. P. Tello. Weigh-in-motion equipment–experiences and challenges. Transportation Research Procedia, 14:1423–1432, 2016.

[52] S. HekmatiAthar, M. Taheri, J. Secrist, and H. Taheri. Neural network for structural health monitoring with combined direct and indirect methods. Journal of Applied Remote Sensing, 14(1):014511–014511, 2020.

[53] HiveMQ. HiveMQ documentation v4.5, 2019.

[54] F. Iasha and P. A. Darwito. Design of algorithm control for monitoring system and control bridge based internet of things (iot). In 2020 International Conference on Smart Technology and Applications (ICoSTA), pages 1–6. IEEE, 2020.

[55] Intel. Intel® core™ i7-4910mq processor specification page.

[56] Z. Juhasz. Quantitative cost comparison of on-premise and cloud infrastructure based eeg data processing. Cluster Computing, 24(2):625–641, 2021.

[57] A. Kamariotis, E. Chatzi, and D. Straub. A framework for quantifying the value of vibration-based structural health monitoring. Mechanical Systems and Signal Processing, 184:109708, 2023.

[58] S. Kamkar and R. Safabakhsh. Vehicle detection, counting and classification in various conditions. IET Intelligent Transport Systems, 10(6):406–413, 2016.

[59] S. Kavitha, R. J. Daniel, and K. Sumangala. High performance mems accelerometers for concrete shm applications and comparison with cots accelerometers. Mechanical Systems and Signal Processing, 66:410–424, 2016.

[60] J. Kim and J. P. Lynch. Autonomous decentralized system identification by markov parameter estimation using distributed smart wireless sensor networks. Journal of Engineering Mechanics, 138(5):478–490, 2012.

[61] P. Kuwałek, P. Otomański, and K. Wandachowicz. Influence of the phenomenon of spectrum leakage on the evaluation process of metrological properties of power quality analyser. Energies, 13(20):5338, 2020.

[62] Q. Ling, Z. Tian, Y. Yin, and Y. Li. Localized structural health monitoring using energy-efficient wireless sensor networks. IEEE Sensors Journal, 9(11):1596–1604, 2009.

[63] H. Liu, J. Ma, T. Xu, W. Yan, L. Ma, and X. Zhang. Vehicle detection and classification using distributed fiber optic acoustic sensing. IEEE Transactions on Vehicular Technology, 69(2):1363–1374, 2019.

[64] J. Liu, S. Chen, M. Bergés, J. Bielak, J. H. Garrett, J. Kovačević, and H. Y. Noh. Diagnosis algorithms for indirect structural health monitoring of a bridge model via dimensionality reduction. Mechanical Systems and Signal Processing, 136:106454, 2020.

[65] Y. Liu, T. Voigt, N. Wirström, and J. Höglund. Ecovibe: On-demand sensing for railway bridge structural health monitoring. IEEE Internet of Things Journal, 6(1):1068–1078, 2018.

[66] S. Lloyd. Least squares quantization in pcm. IEEE transactions on information theory, 28(2):129–137, 1982.

[67] L.-A. Louizos, P. G. Athanasopoulos, and K. Varty. Microelectromechanical systems and nanotechnology: A platform for the next stent technological era. Vascular and endovascular surgery, 46(8):605–609, 2012.

[68] S. Lu, J. Lu, K. An, X. Wang, and Q. He. Edge computing on iot for machine signal processing and fault diagnosis: A review. IEEE Internet of Things Journal, 2023.

[69] J. P. Lynch, A. Sundararajan, K. H. Law, H. Sohn, and C. R. Farrar. Design of a wireless active sensing unit for structural health monitoring. In Health Monitoring and Smart Nondestructive Evaluation of Structural and Biological Systems III, volume 5394, pages 157–168. SPIE, 2004.

[70] J. Mao, H. Wang, and B. F. Spencer Jr. Toward data anomaly detection for automated structural health monitoring: Exploiting generative adversarial nets and autoencoders. Structural Health Monitoring, 20(4):1609–1626, 2021.

[71] M. Mishra, P. B. Lourenço, and G. V. Ramana. Structural health monitoring of civil engineering structures by using the internet of things: A review. Journal of Building Engineering, 48:103954, 2022.

[72] A. A. MOALLEMI, L. ZANATTA, A. BURRELLO, M. SALVARO, M. LONGO, P. DARO, F. BARCHI, D. BRUNELLI, L. BENINI, and A. ACQUAVIVA. Unsupervised vehicle classification using a structural health monitoring system. STRUCTURAL HEALTH MONITORING 2023, 2023.

[73] M. Morgese, F. Ansari, M. Domaneschi, and G. P. Cimellaro. Post-collapse analysis of morandi's polcevera viaduct in genoa italy. Journal of Civil Structural Health Monitoring, 10:69–85, 2020.

[74] L. E. Mújica, M. Ruiz, F. Pozo, J. Rodellar, and A. Güemes. A structural damage detection indicator based on principal component analysis and statistical hypothesis testing. Smart materials and structures, 23(2):025014, 2013.

[75] M. Muttillo, V. Stornelli, R. Alaggio, R. Paolucci, L. Di Battista, T. de Rubeis, and G. Ferri. Structural health monitoring: An iot sensor system for structural damage indicator evaluation. Sensors, 20(17):4908, 2020.

[76] C. Negru, F. Pop, O. C. Marcu, M. Mocanu, and V. Cristea. Budget constrained selection of cloud storage services for advanced processing in datacenters. In 2015 14th RoEduNet International Conference-Networking in Education and Research (RoEduNet NER), pages 158–162. IEEE, 2015.

[77] H. X. Nguyen, S. Zhu, and M. Liu. A survey on graph neural networks for microservice-based cloud applications. Sensors, 22(23):9492, 2022.

[78] Z. Nie, E. Guo, J. Li, H. Hao, H. Ma, and H. Jiang. Bridge condition monitoring using fixed moving principal component analysis. Structural Control and Health Monitoring, 27(6):e2535, 2020.

[79] E. Odat, J. S. Shamma, and C. Claudel. Vehicle classification and speed estimation using combined passive infrared/ultrasonic sensors. IEEE transactions on intelligent transportation systems, 19(5):1593–1606, 2017.

[80] E. Parisi, A. Moallemi, F. Barchi, A. Bartolini, D. Brunelli, N. Buratti, and A. Acquaviva. Time and frequency domain assessment of low-power mems accelerometers for structural health monitoring. In 2022 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0&IoT), pages 234–239, 2022.

[81] C. S. N. Pathirage, J. Li, L. Li, H. Hao, W. Liu, and P. Ni. Structural damage identification based on autoencoder neural networks and deep learning. Engineering structures, 172:13–28, 2018.

[82] C. S. N. Pathirage, L. Li, W. Liu, and M. Zhang. Stacked face de-noising auto encoders for expression-robust face recognition. In 2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA), pages 1–8. IEEE, 2015.

[83] A. Pedrocchi, S. Hoen, G. Ferrigno, and A. Pedotti. Perspectives on mems in bioengineering: A novel capacitive position microsensor [and laser surgery and drug delivery applications]. IEEE Transactions on Biomedical Engineering, 47(1):8–11, 2000.

[84] T. Polonelli, A. Bentivogli, G. Comai, and M. Magno. Self-sustainable IoT wireless sensor node for predictive maintenance on electric motors. In 2022 IEEE Sensors Applications Symposium (SAS), pages 1–6. IEEE, 2022.

[85] T. Polonelli, D. Brunelli, M. Guermandi, and L. Benini. An accurate low-cost crackmeter with lorawan communication and energy harvesting capability. In 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), volume 1, pages 671–676. IEEE, 2018.

[86] T. Polonelli, J. Deparday, I. Abdallah, S. Barber, E. Chatzi, and M. Magno. Instrumentation and measurement systems: Aerosense: A wireless, non-intrusive, flexible, and mems-based aerodynamic and acoustic measurement system for operating wind turbines. IEEE Instrumentation & Measurement Magazine, 26(4):12–18, 2023.

[87] T. Polonelli, H. Müller, W. Kong, R. Fischer, L. Benini, and M. Magno. Aerosense: a self-sustainable and long-range bluetooth wireless sensor node for aerodynamic and aeroacoustic monitoring on wind turbines. IEEE Sensors Journal, 23(1):715–723, 2022.

[88] P. Ragam and N. Devidas Sahebraoji. Application of mems-based accelerometer wireless sensor systems for monitoring of blast-induced ground vibration and structural health: a review. IET Wireless Sensor Systems, 9(3):103–109, 2019.

[89] E. Reynders. System identification methods for (operational) modal analysis: review and comparison. Archives of Computational Methods in Engineering, 19:51–124, 2012.

[90] R. R. Ribeiro and R. d. M. Lameiras. Evaluation of low-cost mems accelerometers for shm: Frequency and damping identification of civil structures. Latin American Journal of Solids and Structures, 16, 2019.

[91] D. Rossi, F. Conti, M. Eggiman, A. Di Mauro, G. Tagliavini, S. Mach, M. Guermandi, A. Pullini, I. Loi, J. Chen, et al. Vega: A ten-core soc for iot endnodes with dnn acceleration and cognitive wake-up from mram-based state-retentive sleep mode. IEEE Journal of Solid-State Circuits, 57(1):127–139, 2021.

[92] A. Sabato, C. Niezrecki, and G. Fortino. Wireless mems-based accelerometer sensor boards for structural vibration monitoring: A review. IEEE Sensors Journal, 17(2):226–235, 2016.

[93] S. S. Saidin, S. A. Kudus, A. Jamadin, M. A. Anuar, N. M. Amin, A. B. Z. Ya, and K. Sugiura. Vibration-based approach for structural health monitoring of ultra-high-performance concrete bridge. Case Studies in Construction Materials, 18:e01752, 2023.

[94] scikit-learn developers. scikit-learn user guide, 2017.

[95] C. Scuro, F. Lamonaca, S. Porzio, G. Milani, and R. Olivito. Internet of things (iot) for masonry structural health monitoring (shm): Overview and examples of innovative systems. Construction and Building Materials, 290:123092, 2021.

[96] P. Shah, A. K. Jain, T. Mishra, and G. Mathur. Iot-based big data storage systems in cloud computing. In Proceedings of Second International Conference on Smart Energy and Communication: ICSEC 2020, pages 323–333. Springer, 2021.

[97] A. Sivasuriyan and D. S. Vijayan. Performance of rc beams utilizing various sensors under fundamental static loading. International Journal of System Assurance Engineering and Management, pages 1–8, 2022.

[98] STMicroelectronics.

[99] STMicroelectronics. LIS344ALH datasheet, 2008.

[100] STMicroelectronics. HTS221 datasheet, 2016.

[101] J. Su, Y. Xia, and S. Weng. Review on field monitoring of high-rise structures. Structural Control and Health Monitoring, 27(12):e2629, 2020.

[102] K. D. Team. Developers guide documentation, 2020.

[103] G. Technologies. GreenWaves Technologies gap9 official description, 2014.

[104] P. Thaprasop, K. Zhou, J. Steinheimer, and C. Herold. Unsupervised outlier detection in heavy-ion collisions. Physica Scripta, 96(6):064003, 2021.

[105] R. K. Verma, K. Pattanaik, P. Dissanayake, A. Dammika, H. Buddika, and M. R. Kaloop. Damage detection in bridge structures: An edge computing approach. arXiv preprint arXiv:2008.06724, 2020.

[106] R. Vidal, Y. Ma, and S. S. Sastry. Principal Component Analysis, pages 25–62. Springer New York, New York, NY, 2016.

[107] H. Wang, A. Jasim, and X. Chen. Energy harvesting technologies in roadway and bridge for different applications–a comprehensive review. Applied energy, 212:1083–1094, 2018.

[108] Q. Wang, J. Zheng, H. Xu, B. Xu, and R. Chen. Roadside magnetic sensor system for vehicle detection in urban environments. IEEE Transactions on Intelligent Transportation Systems, 19(5):1365–1374, 2017.

[109] X. Wang, W. Wu, Y. Du, J. Cao, Q. Chen, and Y. Xia. Wireless iot monitoring system in hong kong–zhuhai–macao bridge and edge computing for anomaly detection. IEEE Internet of Things Journal, 2023.

[110] Y.-P. E. Wang, X. Lin, A. Adhikary, A. Grovlen, Y. Sui, Y. Blankenship, J. Bergman, and H. S. Razaghi. A primer on 3gpp narrowband internet of things. IEEE communications magazine, 55(3):117–123, 2017.

[111] R.-T. Wu, A. Singla, M. R. Jahanshahi, E. Bertino, B. J. Ko, and D. Verma. Pruning deep convolutional neural networks for efficient edge computing in condition assessment of infrastructures. Computer-Aided Civil and Infrastructure Engineering, 34(9):774–789, 2019.

[112] R. Xi, Q. He, and X. Meng. Bridge monitoring using multi-gnss observations with high cutoff elevations: A case study. Measurement, 168:108303, 2021.

[113] P. Yang, C.-J. Hsieh, and J.-L. Wang. History pca: A new algorithm for streaming pca. arxiv (2018). arXiv preprint arXiv:1802.05447, 1802.

[114] Z. Ye, H. Xiong, and L. Wang. Collecting comprehensive traffic information using pavement vibration monitoring data. Computer-Aided Civil and Infrastructure Engineering, 35(2):134–149, 2020.

[115] L. Yi, X. Deng, L. T. Yang, H. Wu, M. Wang, and Y. Situ. Reinforcement-learning-enabled partial confident information coverage for iot-based bridge structural health monitoring. IEEE Internet of Things Journal, 8(5):3108–3119, 2020.

[116] J. Yiu. The Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors. Newnes, 2013.

[117] L. Yu and J.-H. Zhu. Nonlinear damage detection using higher statistical moments of structural responses. Struct. Eng. Mech, 54(2):221–237, 2015.

[118] L. Zanatta, F. Barchi, A. Burrello, A. Bartolini, D. Brunelli, and A. Acquaviva. Damage detection in structural health monitoring with spiking neural networks. In 2021 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4. 0&IoT), pages 105–110. IEEE, 2021.

[119] P. S. Zarrin, C. Martin, P. Langendoerfer, C. Wenger, and M. Diaz. Vibration analysis of a wind turbine gearbox for off-cloud health monitoring through neuromorphic-computing. In IECON 2021–47th Annual Conference of the IEEE Industrial Electronics Society, pages 1–5. IEEE, 2021.

[120] X. Zhao, W. Lin, and Q. Zhang. Enhanced particle swarm optimization based on principal component analysis and line search. Applied Mathematics and Computation, 229:440–456, 2014.

[121] N. Zhou, W. Lin, W. Feng, F. Shi, and X. Pang. Budget-deadline constrained approach for scientific workflows scheduling in a cloud environment. Cluster Computing, pages 1–15, 2020.

[122] L. Zhu, Y. Fu, R. Chow, B. F. Spencer Jr, J. W. Park, and K. Mechitov. Development of a high-sensitivity wireless accelerometer for structural health monitoring. Sensors, 18(1):262, 2018.

[123] F. Zonzini, V. Dertimanis, E. Chatzi, and L. De Marchi. System identification at the extreme edge for network load reduction in vibration-based monitoring. IEEE Internet of Things Journal, 9(20):20467–20478, 2022.

[124] F. Zonzini, A. Girolami, L. De Marchi, A. Marzani, and D. Brunelli. Cluster-based vibration analysis of structures with gsp. IEEE Transactions on Industrial Electronics, 68(4):3465–3474, 2021.

[125] F. Zonzini, M. M. Malatesta, D. Bogomolov, N. Testoni, A. Marzani, and L. De Marchi. Vibration-based shm with upscalable and low-cost sensor networks. IEEE Transactions on Instrumentation and Measurement, 69(10):7990–7998, 2020.

[126] F. Zonzini, F. Romano, A. Carbone, M. Zauli, and L. De Marchi. Enhancing vibration-based structural health monitoring via edge computing: A tiny machine learning perspective. In Quantitative Nondestructive Evaluation, volume 85529, page V001T07A004. American Society of Mechanical Engineers, 2021.

[127] F. Zonzini, M. Zauli, M. Mangia, N. Testoni, and L. De Marchi. Model-assisted compressed sensing for vibration-based structural health monitoring. IEEE Transactions on Industrial Informatics, 17(11):7338–7347, 2021.