

# Exploring Scalable, Distributed Real-Time Anomaly Detection for Bridge Health Monitoring

Amirhossein Moallemi, Alessio Burrello, *Graduate Student Member, IEEE*,  
Davide Brunelli, *Senior Member, IEEE*, and Luca Benini, *Fellow, IEEE*

**Abstract**—Modern real-time Structural Health Monitoring systems can generate a considerable amount of information that must be processed and evaluated for detecting early anomalies and generating prompt warnings and alarms about the civil infrastructure conditions. The current cloud-based solutions cannot scale if the raw data has to be collected from thousands of buildings. This paper presents a full-stack deployment of an efficient and scalable anomaly detection pipeline for SHM systems which does not require sending raw data to the cloud but relies on edge computation. First, we benchmark three algorithmic approaches of anomaly detection, i.e., Principal Component Analysis (PCA), Fully-Connected AutoEncoder (FC-AE), and Convolutional AutoEncoder (C-AE). Then, we deploy them on an edge-sensor, the STM32L4, with limited computing capabilities. Our approach decreases network traffic by  $\approx 8 \cdot 10^5 \times$ , from 780KB/hour to less than 10 Bytes/hour for a single installation and minimize network and cloud resource utilization, enabling the scaling of the monitoring infrastructure. A real-life case study, a highway bridge in Italy, demonstrates that combining near-sensor computation of anomaly detection algorithms, smart pre-processing, and low-power wide-area network protocols (LPWAN) we can greatly reduce data communication and cloud computing costs, while anomaly detection accuracy is not adversely affected.

**Index Terms**—Structural Health Monitoring, IoT, PCA, NB-IoT, Sensors Network.

## I. INTRODUCTION

The constant growth of large-scale civil infrastructures built worldwide in recent years [1], [2] is sustained by increasing investments of the top economies in the world for renewing urban areas and roads. Furthermore, civil infrastructures are

A. Burrello, A. Moallemi, D. Brunelli, and L. Benini are with the Department of Electrical, Electronic and Information Engineering, University of Bologna, 40136 Bologna, Italy.

E-mail: alessio.burrello@unibo.it, amirhossein.moallem2@unibo.it

D. Brunelli is also with the Department of Industrial Engineering, University of Trento, 38123 Trento, Italy.

E-mail: davide.brunelli@unitn.it

L. Benini is also with the Department of Information Technology and Electrical Engineering at the ETH Zurich, 8092 Zurich, Switzerland.  
E-mail: lbenini@iis.ee.ethz.ch

This work was supported by the Italian Ministry for University and Research (MUR) under the program “Dipartimenti di Eccellenza (2018-2022)” and partially supported by the EU H2020-ECSEL project Arrowhead Tools (g.a. 826452). Moreover the authors would like to thank Sacertis S.r.l. for the dataset.

Manuscript received October, 2021.

Copyright (c) 20xx IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org

Link to the github repository of this paper: [https://github.com/MiirHo3eIN/Real\\_Time\\_BHM](https://github.com/MiirHo3eIN/Real_Time_BHM)

This article has been accepted for publication in the IEEE Internet of Things Journal. doi: 10.1109/JIOT.2022.3157532

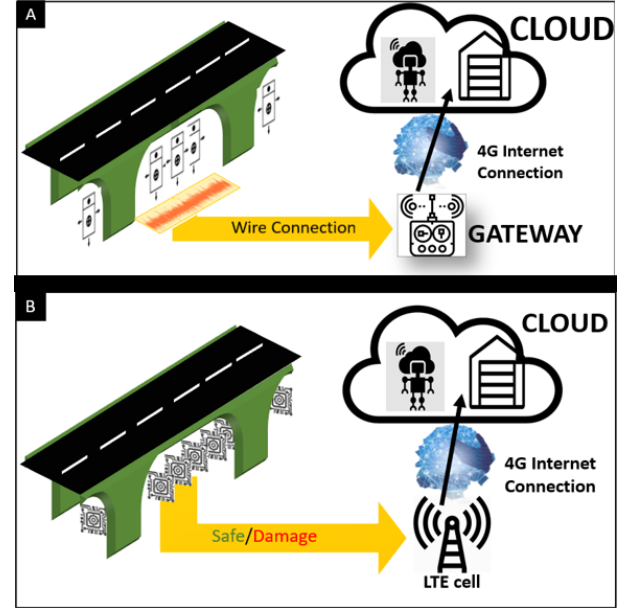


Fig. 1. IoT based SHM systems. In Panel A, the raw signal is gathered from the sensors and sent to the cloud through a gateway to analyse the condition of the structure. In Panel B, the safe/damage condition is directly computed on the node.

becoming increasingly complex. For instance, nowadays, new technologies permit the construction of long-span viaducts, long-extension undersea tunnels, and large skyscraper districts.

On the other hand, although recent structural design achievements assure robustness in these advanced buildings, vulnerability to diverse threats such as extreme weather (e.g., wind, rain), massive vehicular load, and earthquakes remain a major concern [3], [4]. The recent *Polcevera viaduct's collapse* in Genoa, Italy, with over 40 human lives lost, demonstrated that the periodic or sporadic human-assisted assessment of the structures is not enough. Therefore, a change of paradigm towards the continuous observation of structural integrity and automatic anomaly detection is becoming a key requirement for civil infrastructure maintenance [5], [6].

As a consequence, the new field of automated Structural Health Monitoring (SHM), which tracks the real-time online state of structures using dense sensor networks, is gaining prominence [7], [8]. Combined with the advancements in the Internet of Things (IoT) [9], SHM enables monitoring large structures at smaller costs than by deploying human crews [10].

A modern SHM system is composed of a series of sensor-nodes, which capture sensor data, e.g. the vibration of a

viaduct, one or more data collection and processing gateways, and centralized processing and storage resources in the cloud [11]. Fig. 1A depicts a SHM installation on a viaduct. The amount of information and data gathered by new generation SHM systems is exponentially growing, moving from a few measurements every hour from few sensors to continuous high-frequency data streams from dense sensors networks [12], [13]. Therefore, data communication and storage capabilities in the cloud have become major concerns in modern SHM systems.

This paper focuses on two key challenges: automating anomaly detection and doing so with a scalable approach that does not require communication, processing and storing raw sensor data in the cloud. For anomaly detection, different techniques have been proposed, ranging from simple regressive models [14] to deep neural networks [15]. However, they are usually tested on simulated data, not taking into account real-condition perturbations such as wind or climate fluctuations [16], [17]. Besides that, these techniques are typically deployed on cloud servers. Hence they imply comprehensive data collection from the sensor network.

In this work, we address both challenges by proposing a new pipeline for viaduct monitoring. We analyze a real highway viaduct in Italy, which underwent a pre-scheduled maintenance intervention. The viaduct has been monitored before and after the intervention, and acceleration data from five sensors has been collected. We use this real-life example of an exogenous event changing the structural properties of the bridge as a proxy for an abrupt unforeseen event such as an earthquake. Specifically, we give the following contributions:

- We compare data-driven and model-driven unsupervised anomaly detection approaches to monitor the behaviour of the viaduct: namely a Principal Component Analysis (PCA) model and two autoencoders. On our dataset, the PCA shows the best performance with 98.8% accuracy in detecting the structural changes after the interventions. Further, we assessed in depth our anomaly detection approach robustness by synthetically generating new anomalies from the original real-life one.
- We find that the best accuracy (100%) on anomaly detection is obtained by computing PCA-based reconstruction error on 5 seconds time windows and averaging errors over 4 hours, i.e., 2880 windows. Noteworthy, this result comes at the cost of a non-negligible delay in detecting the damage (4 hours). This trade-off between accuracy and delay can be tuned by the user by simply changing the length of the averaging window.
- We study the impact of energy-threshold-based pre-filtering of the raw accelerometer data, showing that this preprocessing block is essential to achieve high anomaly detection accuracy while decreasing computational effort by 17%.
- We compare the performance of the algorithms when fed with time-domain (raw) or frequency-domain (processed with FFT) data. We observe a drop of 21.58% and 12.73% of anomaly detection accuracy, respectively, for both the PCA and Fully Connected (FC) autoencoder, when using frequency domain data.

- We present the implementation of our anomaly detection pipeline on a low-power microcontroller for online inference with  $\approx 74$  uJ energy consumption for each inference. We show the trade-off between accuracy and power consumption by tuning the hyperparameters of our best performing anomaly detector, the PCA. We show that by increasing the Compression Factor (CF) of the PCA (i.e., by reducing the number of principal components utilized during the reconstruction) from 16 to 24, we still achieve 92.97% accuracy in detecting structural changes (i.e., distinguishing anomalies from normal samples) while consuming only 39 uJ per inference.
- We demonstrate our distributed approach using a node equipped with a Narrowband IoT communication unit [18], which exploits anomaly detection at the edge. We show that by performing computation on the edge and communicating only post-processed data (i.e., detected anomalies), the energy consumption of a node can be reduced by  $5.0\times$ .

The paper is organized as follows. Sec. II describes the related work. Sec. III introduces the viaduct structure, the SHM framework installed, and the background on the anomaly detection methods proposed in this work. Sec. IV describes our proposed pipeline, with different analyzed design and deployment choices. Sec. V and Sec. VI provide results of both the anomaly detector accuracy and the deployment-related metrics, i.e., memory footprint, energy consumption, and network data communication. Sec. VII concludes the paper.

## II. RELATED WORK

Structural Health Monitoring systems have become widespread in the last decade. They are usually based on networks of sensors to monitor the vibration of the structure under test [29]. A key 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.

1) *Statistical data modeling*: The first approaches in continuous SHM systems were based on modeling data distribution and extracting abnormal patterns. Ling et al. [19] 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) detections, they performed experiments with four laboratory computed datasets. Similarly, auto-regressive models are employed in e.g., [30]–[32] to extract features from raw vibration data. One of the most recent works is Entezami, et al. [14] that propose 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

TABLE I

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.

	Structure	Sensors	Data Type	Detection Model	Train Device	Test Device	Performance
<b>Statistical Data Modelling</b>							
Ling et al. [19]	Simulated Steel Frame Structure	120	Acceleration	Autoregressive model	Remote Server	Remote Server [20]	1 FP, 1 FN
Santos et al. [21]	Simulated Five Bay Structure	4×29	Acceleration	FFT + Peak Detection	N.A.	MICAz [22]	0 FN, 2 FP
Verma et al. [23]	Simulated Steel Beam Bridge	2 14	Acceleration	Features + Gaussian Model	N.A.	N.A.	85-96% 96.3-100%
<b>Deep Neural Networks</b>							
Acvi et al. [24]	BM benchmark [25]	12	Acceleration	1D-CNN	Intel core-i7 [26]	Intel core-i7 [26]	N.A.
<b>Data reduction</b>							
Liu et al. [27]	Simulated Lab-Scale Bridge	5	Acceleration	Autoencoder	N.A.	N.A.	N.A.
Nie et al. [28]	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%

occupation. Then, a k-Nearest Neighbours algorithm classifies the samples, achieving as low as 1.56% of misclassification. Despite the optimal results achieved, these work rely on a set of hand-tuned parameters, which impair the generality of the model 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 of a single cloud orchestrating unit.

The most recent study based on data modeling is [23], 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 impair 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 change in behavior has been observed.

Santos et al. [21] 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 second time window at the node. Then, computed natural frequencies are sent to sensors 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 False Positives (FP). Transmitting only natural frequencies causes a network traffic of 6.720 KB/h. In a similar

vein, we further decrease network traffic to only 10B/h by applying the Principal Component Analysis for data compression and classification directly on the node. Noteworthy, as demonstrated in section V, employing frequency features strongly impairs anomaly detection performance on our structure, making this approach unsuitable for our problem. Similar to Santos et al. [21], other works, e.g., [33]–[35], study the pros and cons of cloud computing and edge computing in the content of SHM systems.

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

2) *Deep Neural Networks*: In [24], [36], the authors present two DNN-based approaches. A 1D-CNN is used in [36] to estimate the Probability of Damage (PoD) on the BM benchmark [25]. A PoD close to 0 points to the normal case, whereas a PoD of 1 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 the conventional 3D CNNs, 1D CNNs require less computational complexity, 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 an online-training on edge-nodes.

On a totally different input data, images, Wu et al. [37] 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 MB and ResNet18 to 2MB). 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, and, more in general, labeled anomaly data is not available in typical structural health monitoring installations.

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 is the difference between the original and the reconstructed signals, the higher is the probability of damage. In this context, autoencoder (AE) neural networks are among the most popular approaches. For example, [27] uses an AE for damage diagnosis on a laboratory's synthetic bridge for indirect bridge monitoring scenarios, outperforming all other anomaly detection algorithms with  $MSE \approx 5$  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., [28], [38]–[40]). For example, [28] 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 of 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., i7 intel processor) after data transmission and collection.

In our work, 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 for 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.

### III. SHM INSTALLATION & BACKGROUND

#### A. Bridge Structure

The vibration data analyzed in this work comes from a viaduct located in northern Italy on the ss335 state highway, composed of 18 different sections. Last year, 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 2. For this reason, in this work, we analyze the unique situation of accelerations gathered before and after this strengthen 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

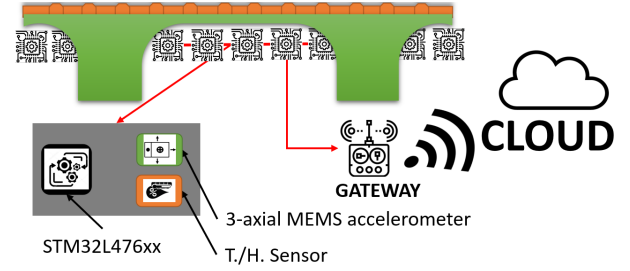


Fig. 2. Overview of the installed monitoring system on the viaduct. Five sensors are linked to a gateway for streaming data to the cloud. The grey box showcases the main components of a sensor node.

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.

#### B. 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 health condition. Fig. 2 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 neither on the nodes nor 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 processing on the data 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 bits 4-core Cortex-A53 running at 1.2 GHz and 1 GB of DDR2 RAM. The gateway supports Linux operating system, allowing for typical python applications deployment for either communication (e.g., an MQTT broker [41]) and in-field machine learning (e.g., Keras [42], scikit-learn [43]).

The sensor node is represented in the lower part of Fig. 2. It is composed of the LIS344ALH analog tri-axial accelerometer [44], the HTS221 temperature and humidity sensor [45], 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 MHz, with 96 KB of SRAM and 1MB of Flash memory. This node samples the acceleration with the internal ADC at a frequency of 25.6 kHz. For increasing the bit-resolution, windows of 256 samples are filtered with a 6-state FIR filter and reduced to a single value, thus obtaining a final sampling rate of 100 Hz.

#### C. Anomaly detection approaches

1) *Principal Component Analysis*: Principal Component Analysis (PCA) is a method to deal with high dimensional correlated data by transforming them into minimally correlated data [46]. 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

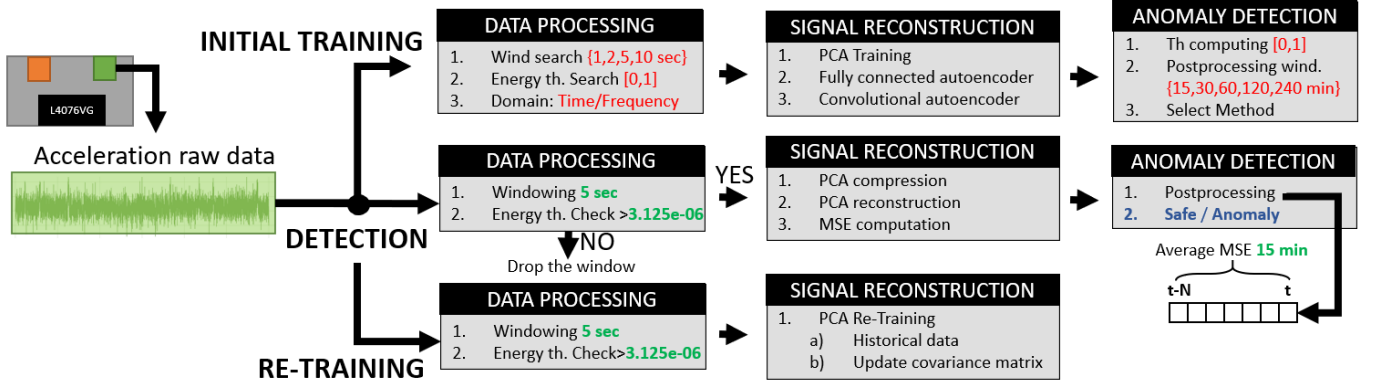


Fig. 3. The proposed framework to analyze the condition of a viaduct starting from raw acceleration data. In the top part of the figure, we show the hyper-parameter tuning (in red) and the initial training steps done before the first time that the monitoring system is activated. In the middle, we show 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 anomalous event to avoid the increase of false-positive alarms due to the bridge's static deformation caused by wind or ageing.

matrix, ordered by the value of their eigenvalues. PCA reduces data size by preserving only directions that retain most of the information [47] (the ones with the associated higher eigenvalues). Considering a  $M \times N$  dimensional data matrix  $x = [x_1, x_2, x_3, \dots, x_N]$  where  $x_k$  is a column vector of  $M$  features representing a sample, its normalized covariance matrix is

$$\Sigma = \frac{1}{N-1} \sum_{k=0}^N (x_k - \bar{x})(x_k - \bar{x})^T \quad (1)$$

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

$$\Sigma = V \Lambda V^{-1}. \quad (2)$$

where  $V$  columns represents the eigenvectors, and the principal diagonal of  $\Lambda$  contains corresponding eigenvalues. It can be proven that  $V^k \in \mathbb{R}^k$  is a basis of the sub-space of dimensions  $\mathbb{R}^k$  which retains the highest similarity with the original one.

Due to the usually high number of features,  $M$ , PCA requires a high memory footprint to store and compute the covariance  $M \times M$  matrix and extract its eigenvalues. To cope with this limitation on low-memory edge devices, the authors of [48] introduced *History PCA*, a streaming algorithm to train the PCA without storing data, which has been later deployed on edge/nodes by [49]. Compared to other streaming approaches, HPCA exploits the history of the data and new samples to update the partial covariance matrix, allowing a faster convergence and better accuracy [48], [49]. In our work, we exploit HPCA to deploy our algorithm on the sensor nodes, moving both training and detection from the gateway to the leaf nodes of the SHM sensor network.

2) *Autoencoders*: Autoencoders are neural networks composed of two or more layers used to compress data and detect anomalies [50]. 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  $h \in \mathbb{R}^k$ , exploiting one or multiple layers, either fully

connected, convolutional or recurrent [51]. An example of a single-layer encoder is

$$h = f_E(x) = \Phi(W_E x + b_E) \quad (3)$$

where  $W$  is the weight matrix, and  $\Phi$  is the activation function of a single layer. 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

$$\bar{x} = f_D(h) = \Phi(W_D h + b_D). \quad (4)$$

The model's training favours 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.

#### IV. METHODS: ANOMALY DETECTION IN AN SHM FRAMEWORK

This section describes the main contribution of this work. We discuss our novel SHM pipeline and its deployment to augment the SHM installations to raise *integrity* alarms automatically. We first show our complete pipeline, comprising a step of initial training, the in-field estimation, and the possibility for an online update of the models. Then, we describe our proposed solutions to efficiently embed our pipeline inside the existing system, reducing the energy consumption and network traffic while maximizing the system's scalability for large SHM installations.



### A. Anomaly detection pipeline

As shown in Fig. 3, our pipeline is composed of 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 decompressed and original signal is computed to detect the structural integrity of the viaduct. An average over time is calculated to smooth the damage detection, reducing false alarms.

1) *Pre-processing*: This step covers the windowing of 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. 3 shows, data processing starts by dividing acceleration raw data into non-overlapping windows, similar to [49]. 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 is often resulting 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. Energy of each window is computed as follows:

$$E = \sum_{i=1}^{W_d} X_i^2 \quad (5)$$

where  $W_d$  is the width of each window. The search of energy threshold is done exploiting the iterative steps of Alg. 1. At each step, an increase in 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  $RSNR = 20 \log_{10} \left( \frac{\|\mathbf{x}\|_2}{\|\mathbf{x} - \hat{\mathbf{x}}\|_2} \right)$ , with  $\mathbf{x}$ , the original signal, and  $\hat{\mathbf{x}}$ , the reconstructed one. Based on [49] and considering a compression factor of  $15\times$  as in [49], we set this lower bound average RSNR to 16 dB. Fig. 4 shows acceleration data and highlights the portion of the signals selected by the tuned energy threshold with green background. Fig.4-B and Fig.4-C show a zoom of peak ② in the time and frequency domain. As detailed in Sec. V, applying this energy filtering improves the accuracy of all our analyses.

2) *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.

---

### Algorithm 1 Energy Filtering

---

```

1: Input:  $\mathbf{X}_{\text{train}}, \mathbf{X}_{\text{val}}$ 
2:  $th = 10^{-10}$ 
3: do
4:    $th+ = 2^{-8}$ 
5:    $\mathbf{X}_{\text{train}}, \mathbf{X}_{\text{val}} \leftarrow \text{filter}(\mathbf{X}_{\text{train}}, \mathbf{X}_{\text{val}}, th)$ 
6:    $\mathbf{W} \leftarrow \text{pca}(\mathbf{X}_{\text{train}})$ 
7:    $\mathbf{X}_r \leftarrow \mathbf{X}_{\text{val}} \mathbf{W} \mathbf{W}^\top$ 
8:    $S \leftarrow \text{RSNR}(\mathbf{X}, \mathbf{X}_r)$ 
9: while  $S < 16$  dB
10: Output:  $th$ 

```

---

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  $\mathbf{A} 1 \times 500$ ,  $\mathbf{B} 500 \times 16$  and  $\mathbf{C} 16 \times 500$ ) 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 is composed of 8 hidden layers followed by ReLU activations. Adam optimizer, along with 80 epochs, is used to train this model.

3) *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:

$$MSE = \|x_i - \bar{x}_i\|_{L2} = \frac{1}{n} \sum_{i=0}^{n-1} (x_i - \bar{x}_i)^2 \quad (6)$$

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 the 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, where PCA is used to compute MSE over normal and abnormal data.

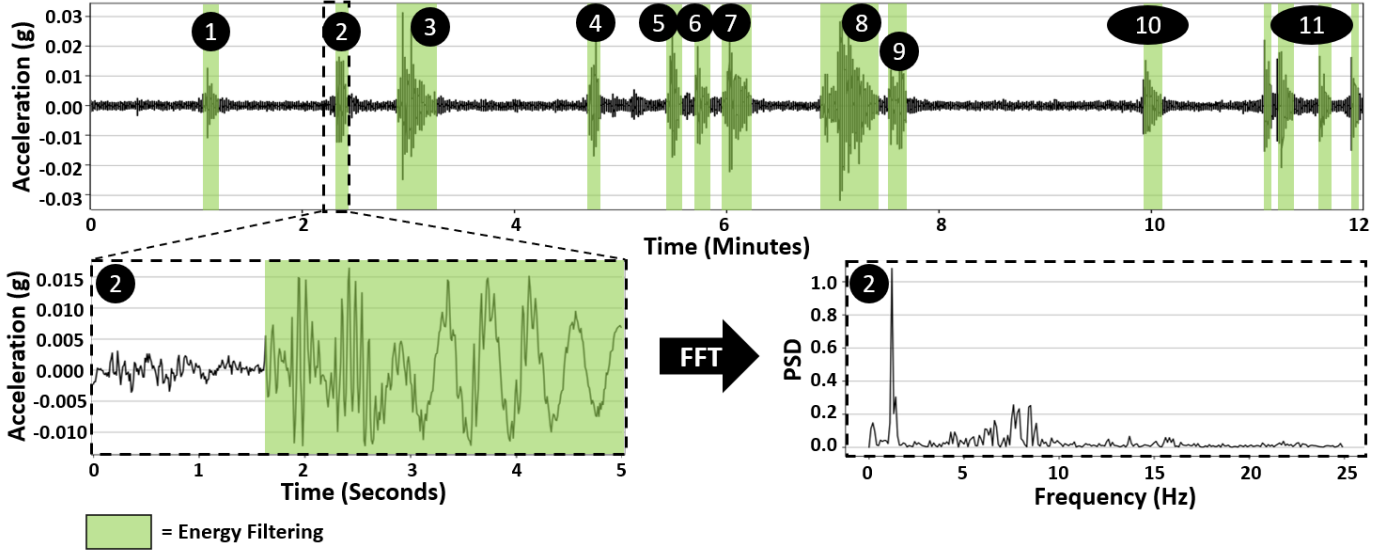


Fig. 4. Top panel. Twelve minutes of mean-centered raw acceleration data of the  $z$ -axis of the middle sensor installed on a pier of the bridge. Peaks are associated with vehicle passages. Left panel. Zoom of a 5-second window containing the oscillation associated with the passage of a vehicle. Right panel. Frequency response of the window of signal highlighted with the dashed rectangle.

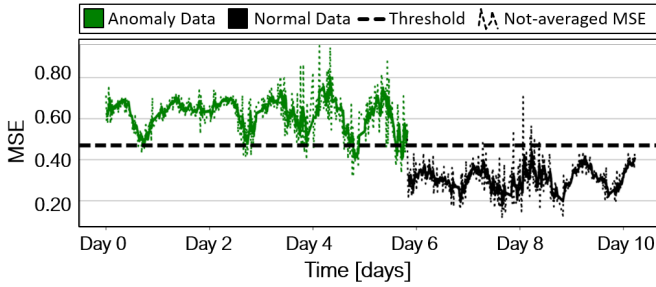


Fig. 5. PCA output mean square error (MSE) on the test dataset. Input window dimension is set to 5 seconds. Solid line is obtained by applying the post-processing with window dimension = 1 hour.

To further reduce the false alarms, we propose an average over time of the soft-predictions (MSE values). We explore windows between 15 minutes to 4 hours, showing that a larger window is positively correlated with better accuracy, increasing the gap between reconstructed normal data and reconstructed anomalies but causes larger delays in prediction.

### B. Algorithm Phases: Train, Detect, Re-Train

Our pipeline is characterized by three main phases (Fig. 3, 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 the definition of 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 the update of 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.

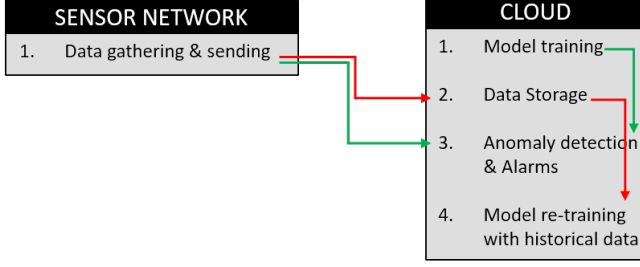
### C. Deployment: Sensor Vs. Cloud

Deploying our proposed anomaly detection pipeline (Fig. 3) 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, we discuss the trade-offs of performing the three different phases (i.e., training, anomaly detection, and re-training) of our algorithm either on the cloud or on the nodes, or as a mix of them. Noteworthy, these reasonings also hold for much bigger SHM installation of hundreds of nodes, where the scalability issues and the data storage can be the real bottleneck of the system.

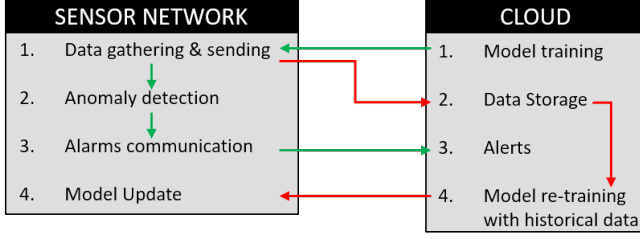
1) *Cloud Computing*: As shown in Fig. 6-1, transmitting all the data to the cloud while having no processing in the sensor network causes i) a high data communication cost, ii) the necessity of cloud data storage, and iii) a daily cloud computation of anomalies, alarms, and, less frequently, the iv) re-training of the model.

The transmission of data to the cloud is the first issue in this scenario. Although several cost minimization techniques

### 1. CLOUD COMPUTATION



### 2. SENSOR INFERENCE + CLOUD TRAIN



### 3. SENSOR COMPUTATION

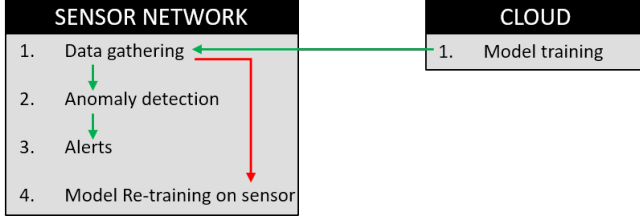


Fig. 6. Three deployment scenarios of our anomaly detection pipeline. Green arrows highlights the inference steps, while red ones the re-training and updating of the model over time.

such as new communication paradigms [52], [53] or edge data-reduction [49] 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) [54], 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 [55] 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.

2) *Sensor Interface with Cloud*: Involving sensors in the computation reduces the anomaly detection pipeline's costs. The anomaly detection model is exported to the sensor to predict the viaduct behaviour, while the model re-training is still performed in the cloud. Fig 6-2 shows the overall functionality of this approach. In green, we highlight the anomaly detection pipeline, while in red, the update of the model 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.

3) *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 one time 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, the scalability is not more a problem since the cloud is only used to monitor the sensors' status and initialize them. Fig. 6-3 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 constraint 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 [49], which aims at finding a compression matrix sequentially to avoid i) storing lots of data at edge and ii) compute the entire covariance matrix [48]. Compared to [49], instead of employing the PCA only for data compression, we use it also to perform anomaly detection at the edge of the sensor network.

## V. EXPERIMENTAL RESULTS: ALGORITHM EXPLORATION

In this section, we mainly focus on analyzing the proposed framework in Fig. 3. Using grid search over different hyperparameters and framework elements, we explore the performance of our pipeline while changing both 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. Finally, we compare the proposed anomaly detectors with state-of-the-art ones. To be fair, we reproduced the state-of-the-art algorithms and we applied them to our data using the same input window dimension and post-processing.

### A. Notations & Benchmark

First, we introduce the notations and metrics that we use to evaluate the different methods and hyperparameters in this work. 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

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



TABLE II  
PERFORMANCE OF OUR PIPELINE CHANGING ANOMALY DETECTION  
ALGORITHM WITH DISCRETE WAVELET TRANSFORM, FREQUENCY AND  
TIME DATA AS INPUT SPACE DOMAIN.

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 %

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 autoencoders data. Finally, we define input dimension, the length of each non-overlapping window in the data processing step, and output dimension, the total time considered after averaging multiple windows before final classification.

Our test dataset comprises 25 days of continuous monitoring of the viaduct with 5 sensors described in Sec. III-A. 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 days before the maintenance intervention, labeled as anomalies, and 20 days after, labeled as normal data. We select as the test set all the 5 days of anomalies and 5 days of normal data to have a balanced test dataset. We divided the remaining 15 days of normal data into a validation set (5 days) and the training set (10 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.

### B. Model selection & Data domain

Modal analysis is the gold standard used to analyze the dynamic characteristics of large-scale buildings [57]. On the other hand, previous studies have demonstrated the feasibility of using raw time series for anomaly detection [49]. Hence, both time and frequency domains are promising directions to analyze. Therefore, we test three anomaly detectors fed

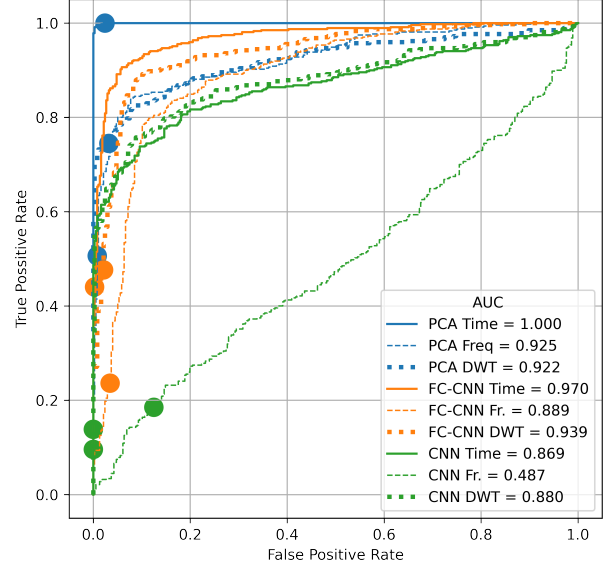


Fig. 7. ROC curve for different input signal domains, i.e., time, frequency, and time-frequency.

with both 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 [58]. We fix the input window dimension to 5 seconds of accelerometer output samples and the output dimension to 60 minutes for this comparison. The compression factor is fixed to 16; therefore, we select the most significant 16 principal components for PCA, while ensuring the innermost latent dimension of both fully connected and convolutional autoencoders to have a dimension of 16.

Table II and Fig. 7 report the evaluation results on the three models using the 10 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 are threshold independents. 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 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 ( $\simeq 30\%$  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 0.970. 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, also a small modification in the threshold can impair the accuracy,

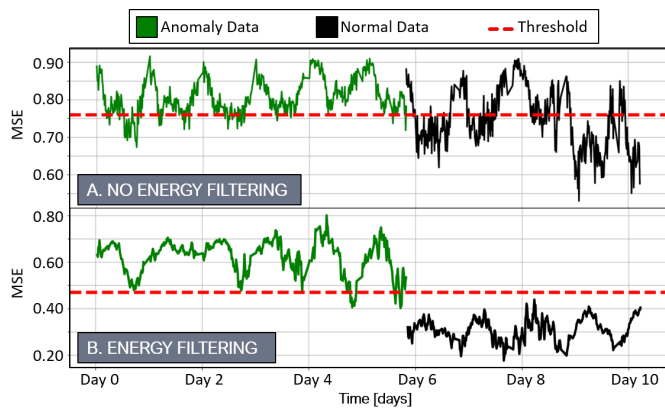


Fig. 8. The energy filtering step impacts on the PCA output MSE. In the top panel, we show the MSE when the energy filtering is not applied. In the bottom panel, we show the improved result with its application.

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  $37.1\%$ . 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 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 Hz. Although with the FFT pre-processing, we can reach high AUCs of 0.925 and 0.889 for our best models, 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-stationary or non-linearity [59]. 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 0.922 and 0.939 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 II and Fig. 7 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 best competitors, and we, therefore, use them in subsequent analysis and in deployment on edge nodes.

### C. Hyperparameters exploration

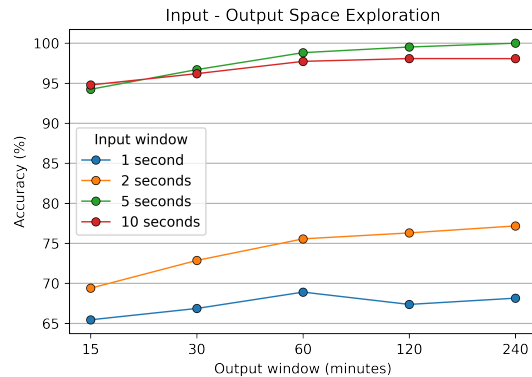


Fig. 9. Effect of input-output dimensions sweep on the performance of the best detector (PCA).

1) *Energy Filtering*: In Sec IV, 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 8 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  $\approx 41\%$ ). 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 improving the detector performance strongly. 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.

2) *Input & Output Dimension Exploration*: Fig. 9 shows the tuning of input and output dimensions, with twenty different combinations of four input dimensions with five output dimensions. Input dimension variation is not positively/negatively correlated with algorithm performance. We can notice that using 5 seconds (grid search between 1,2,5 and 10 seconds) outperforms the other input dimensions values from Fig. 9. 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, since viaduct structure modifications are slow, having very low delays is not required. Therefore, we decided to use 60 minutes of output dimension, which almost saturates performance for 5 seconds windows while having a reasonable delay. Impres-

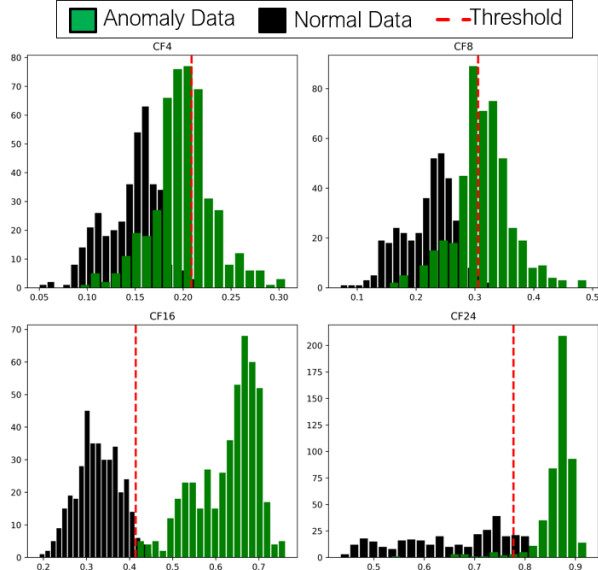


Fig. 10. MSE distribution while changing CF parameter

sively, increasing the output dimension to 120 and 240 minutes provides further 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 minutes as a dimension leads to the perfect distinction of anomalies and safe time slots (100% accuracy) but causes a delay of 4 hours in the notification of the damage alarm.

3) *Compression factor*: We also explore different compression factors for PCA to analyze its effect on framework 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. 10 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 MSE with CF = 4), while using a higher CF (CF = 24) causes a higher reconstruction error (0.4-0.9 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 distribution. 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 MSE) and a poor reconstruction of anomalies (0.4-0.8 MSE). As can be 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 input data dimension, this parameter affects both the computation and memory footprint of the algorithm and will be further explored in the following sections.

#### D. Synthetic Experiments

Lastly, we artificially generated degradations to monitor the robustness of the best-trained detector, i.e., PCA. To inject

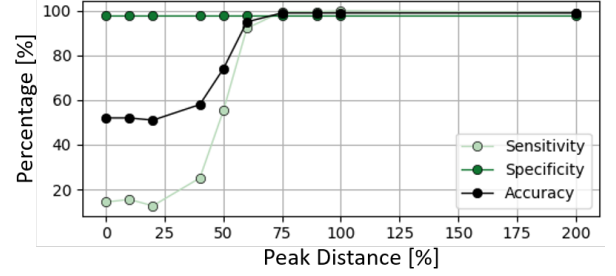


Fig. 11. Performance of the PCA classifier while sweeping over several severities of anomalies w.r.t real case scenario of the bridge.

TABLE III  
COMPARISON OF OUR PROPOSED SOLUTION WITH STATE-OF-THE-ART METHODS APPLIED TO OUR DATASET. THE 60 MINUTES POST-PROCESSING IS IDENTICALLY APPLIED TO ALL METHODS.

Method	Acc.	Spec.	Sens.
<b>State-of-the-art algorithms</b>			
FFT + peaks detection [21]	67.79 %	99.2%	43.09 %
MGD [23]	59.48 %	95.66%	10.25 %
AR features + MSD [60]	58.43 %	85.90%	36.82 %
AR features + L1	81.11 %	88.49%	71.80 %
<b>Our Work</b>			
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%

different sets of anomalies, we modified the distance between the two peaks of normal and anomaly in spectrum density and transformed them back to the time domain. We take each 15 minutes 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 back the data to the time domain prior and use these new data to test the algorithm. Reducing the gap between the two peaks allows producing data more similar to normal data, implying a harder task for the detector. Fig 11 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.

#### E. Model Comparison

In this section, we compare our anomaly detectors with methods presented in Sec II. To do this, we reproduce the pipeline shown in Fig.3, substituting the anomaly detection algorithm with the state-of-the-art ones but keeping unchanged the pre-processing and post-processing steps. We compare the PCA with four other statistical-based approaches based on frequency peak detection [21], Multivariate Gaussian Distribution [23], and AutoRegressive models [60]. We do not add to the comparison any supervised deep learning methods since they require labels for normal and anomaly cases which are not available at training time in normal SHM use-cases. Table III 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 [60], [61]. Entezami et al. [14] 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 anomalous data. Table III shows that L1 achieves an overall better accuracy (+22.78%) than Mahalanobis Square Distance (MSD). Santos et al. [21] 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 not suitable to distinguish safe and anomalous time windows on our dataset. Finally, we investigated a recent study that targets edge computing [23], exploiting seven 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, also this method fails in our use-case, with a drop in accuracy to  $\approx 60\%$ .

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

## VI. EXPERIMENTAL RESULTS: PIPELINE DEPLOYMENT

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. III-B, the STM32L476VGTx. All pipeline steps, including data processing, signal reconstruction, and anomaly detection, have been deployed using optimized C code and 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 each case's 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 IV.

### A. CF tuning Vs. Metric Figures

Starting from the accuracy results shown in Sec. V, 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 minutes. We show the results of our exploration in Fig.12. 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 seconds, respectively. Despite the higher accuracy of CF = 16, increasing the CF allows reducing both the memory footprint and energy consumption. For instance, from the first graph of Fig.12, we can notice that using CF = 24 with a window of 5 seconds still allows us to reach an

TABLE IV  
DEPLOYMENT METRICS OF PCA ALGORITHM WITH CF = 16, OUTPUT DIMENSION = 60 MINUTES, AND VARIABLE INPUT DIMENSION. TIME AND ENERGY ARE ONLY FOR ONE INFERENCE.

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	<b>91.04</b>	<b>77.55</b>	<b>6.428</b>	<b>73.96</b>
10	276.54	Overflow	-	-

acceptable accuracy of 92.97%. Contrary, using a lower CF causes i) higher energy, ii) higher memory consumption, and iii) lower accuracy, totally 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 seconds input dimension since its compression matrix does not fit the small 96 kB RAM (dotted line in the second graph of Fig.12). 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 seconds. 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 seconds, achieving 98.82% accuracy with a 73.96uJ energy consumption per inference.

Table IV 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 kB for 5 seconds). Although the compression matrix is not a problem (roughly 10% of total FLASH for 5 seconds), 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 seconds, reducing the input dimension to 1 second allows us to maintain accuracy of 67.34%, with a reduction of latency 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 Hz), the latter causes a shorter node lifetime, creating a trade-off between accuracy vs. energy consumption.

### B. 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 [62].

NB-IoT consumes more energy per payload packet than other similar technologies. However, 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 [62]. 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, the streaming of data need not to 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

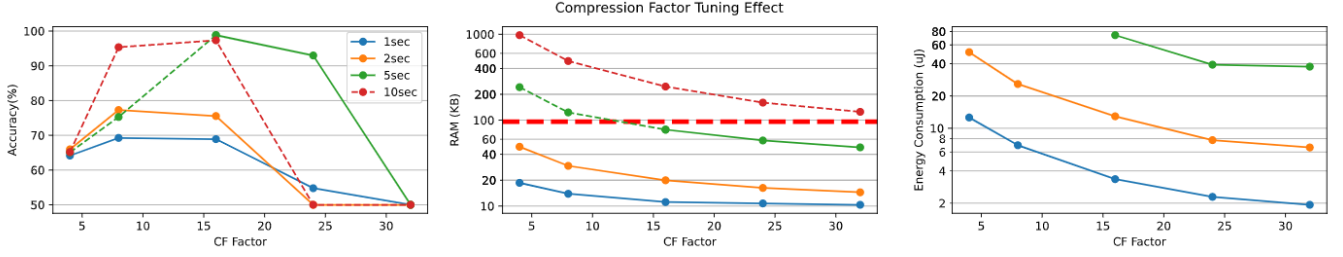


Fig. 12. CF tuning versus accuracy, memory, and energy. Horizontal red line points to the limit of MCU memory. Dotted lines represent not deployable solutions.

TABLE V  
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

Scenario	Network Traffic (B/h)	NB-IoT E. [J] (1h)	Node Comp. E. [J] (1h)	Gathering E. [J] (1h)
<b>Inference</b>				
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	<b>3B</b>	<b><math>0.7130 + E_{sleep}</math></b>	<b>0.005</b>	<b>62.4</b>
<b>Train</b>				
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	<b>0 B</b>	$E_{sleep}$	<b>0.00162</b>	<b>62.4</b>

to gauge the benefits of the different scenarios introduced in [52].

Fig. 6 shows all the options for training and inference. The cloud-based method continuously streamlines the data to the cloud for both the training and detection phases. In contrast, sensor computation only reports the structure's status to the cloud on an hourly basis. We want to quantify each of these scenarios regarding energy consumption and transition costs to develop a scalable solution for SHM applications.

Tab. IV reports the deployment results of model inference at the node. The best solution in terms of accuracy, i.e., the PCA with 5 seconds input window dimension, consumes 73.96 uJ. Exploiting a smaller input window dimension, i.e., 1 second, only consumes 3.35 uJ. 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 V, 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 seconds input dimensions to preserve the performance.

The node installed on the viaduct works with an output sampling rate of 100 Hz; thus, it generates 100 16-bits samples per second. Therefore, the node generates 200 Bytes per second, leading to 720KB per hour. To estimate this node's energy consumption with the NB-IoT protocol, we use the estimations provided in [52], where diversity in the payload for each packet affects the power consumption of the node. We decided to use a payload of 1300 B for this experiment. This selection of payload can send 650 (1300/2) samples per packet. Hence, we need 554 packets to transmit 720Kb 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 ( $\approx 1J/h$ ) to an off-chip memory (e.g., a micro SD card). Table V summarizes the energy consumption regarding different sections of both the training and inference part. It shows that exploiting the full deployment of the cloud computing approach consumes 312.848 J/h where it reduces to 63.508 J/h for the localized sensor deployment of our pipeline. An approximate  $5\times$  drop of 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 structure's status to the cloud, we can reduce the traffic of the system to only 3B (i.e., "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 Bytes (smallest possible number) for each packet. Then, we only transmit one packet to the cloud leading to less than 1 J energy consumption. Although our solution reduces transmission cost to the cloud, allowing scalable solutions for large-scale structures, it consumes a high energy rate at the node. Table V also reports different sides of transmission vs. node energy consumption trade-off for both training and inference phases. The high traffic rate during inference ( $\sim 780$  kB/hour) for a complete cloud-based approach prohibits its utilization for large-scale SHM scenarios. Contrary, our solution reduces the traffic of only 10B/h to the cloud 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).



## VII. CONCLUSION

This work proposes 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. First, we propose a new damage detection pipeline, comprising a pre-processing step, an anomaly detection algorithm, and a postprocessing step. Comparing PCA, and two different autoencoders, we show that PCA outperforms the other two methods by approximately 30% and 48%, on our SHM dataset collected on a real-standing Italian bridge. We show that by tuning hyperparameters of our pipeline, we further improve the accuracy in the detection of anomalies by 20%.

Additionally, we demonstrate the embedding of 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 KBytes/hour to 10 Bytes/hour, compared to a cloud-based anomaly detection solution. Further, we reduce the power computation for the node by  $5 \times$ .

## REFERENCES

- [1] 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*, 2020.
- [2] C. R. Farrar and K. Worden, *Structural health monitoring: a machine learning perspective*. John Wiley & Sons, 2012.
- [3] 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)*. IEEE, 2020, pp. 1–6.
- [4] R. Xi, Q. He, and X. Meng, "Bridge monitoring using multi-gnss observations with high cutoff elevations: A case study," *Measurement*, vol. 168, p. 108303, 2021.
- [5] M. Morgese, F. Ansari, M. Domaneschi, and G. P. Cimellaro, "Post-collapse analysis of moravito's polcevera viaduct in genoa italy," *Journal of Civil Structural Health Monitoring*, vol. 10, no. 1, pp. 69–85, 2020.
- [6] E. Figueiredo and E. Cross, "Linear approaches to modeling nonlinearities in long-term monitoring of bridges," *Journal of Civil Structural Health Monitoring*, vol. 3, no. 3, pp. 187–194, 2013.
- [7] F. Lamonaca, P. F. Sciammarella, C. Scuro, D. L. Carni, and R. S. Olivito, "Internet of things for structural health monitoring," in *2018 Workshop on Metrology for Industry 4.0 and IoT*, 2018, pp. 95–100.
- [8] 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.
- [9] S. S. Arslan, R. Jurdak, J. Jelitto, and B. Krishnamachari, "Advancements in distributed ledger technology for internet of things," 2020.
- [10] C. Arcadius Tokognon, B. Gao, G. Y. Tian, and Y. Yan, "Structural health monitoring framework based on internet of things: A survey," *IEEE Internet of Things Journal*, vol. 4, no. 3, pp. 619–635, 2017.
- [11] H. Cai, B. Xu, L. Jiang, and A. V. Vasilakos, "IoT-based big data storage systems in cloud computing: Perspectives and challenges," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 75–87, 2017.
- [12] 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*, vol. 9, no. 2, p. 917923, 2013. [Online]. Available: <https://doi.org/10.1155/2013/917923>
- [13] 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*, vol. 19, no. 6, pp. 1685–1710, 2020.
- [14] A. Entezami, H. Sarmadi, B. Behkamal, and S. Mariani, "Big data analytics and structural health monitoring: A statistical pattern recognition-based approach," *Sensors*, vol. 20, no. 8, 2020. [Online]. Available: <https://www.mdpi.com/1424-8220/20/8/2328>
- [15] 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*, vol. 8, no. 5, pp. 3108–3119, March 2021.
- [16] G. Pang, C. Shen, and A. van den Hengel, "Deep anomaly detection with deviation networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 353–362. [Online]. Available: <https://doi.org/10.1145/3292500.3330871>
- [17] P. Thaprasop, K. Zhou, J. Steinheimer, and C. Herold, "Unsupervised outlier detection in heavy-ion collisions," *arXiv preprint arXiv:2007.15830*, 2020.
- [18] L. Feltrin, G. Tsoukaneri, M. Condoluci, C. Buratti, T. Mahmoodi, M. Dohler, and R. Verdore, "Narrowband IoT: A survey on downlink and uplink perspectives," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 78–86, 2019.
- [19] Q. Ling, Z. Tian, Y. Yin, and Y. Li, "Localized structural health monitoring using energy-efficient wireless sensor networks," *IEEE Sensors Journal*, vol. 9, no. 11, pp. 1596–1604, 2009.
- [20] P. E. E. R. Center. (1998) Home page. [Online]. Available: <https://opensource.berkeley.edu/index.php>
- [21] 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*, vol. 15, pp. 114–129, 2014.
- [22] I. Crossbow Technology. MICAz datasheet. [Online]. Available: [http://courses.ece.ubc.ca/494/files/MICAz\\_Datasheet.pdf](http://courses.ece.ubc.ca/494/files/MICAz_Datasheet.pdf)
- [23] R. K. Verma, K. Pattanaik, P. Dissanayake, A. Dammika, H. Buddika, and M. R. Kaloo, "Damage detection in bridge structures: An edge computing approach," *arXiv preprint arXiv:2008.06724*, 2020.
- [24] 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)*, 2018, pp. 4600–4607.
- [25] 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.
- [26] Intel® core™ i7-4910mq processor specification page. [Online]. Available: <https://ark.intel.com/content/www/us/en/ark/products/78939/intel-core-i7-4910mq-processor-8m-cache-up-to-3-90-ghz.html>
- [27] 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*, vol. 136, p. 106454, 2020.
- [28] 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*, vol. 27, no. 6, p. e2535, 2020.
- [29] Y. Liu, T. Voigt, N. Wiström, and J. Höglund, "Ecovibe: On-demand sensing for railway bridge structural health monitoring," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1068–1078, 2019.
- [30] M. Gul and F. N. Catbas, "Damage assessment with ambient vibration data using a novel time series analysis methodology," *Journal of Structural Engineering*, vol. 137, no. 12, pp. 1518–1526, 2011.
- [31] L. Yu and J.-H. Zhu, "Nonlinear damage detection using higher statistical moments of structural responses," *Struct. Eng. Mech.*, vol. 54, no. 2, pp. 221–237, 2015.
- [32] 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*, vol. 7, no. 2, pp. 153–162, 2017.
- [33] 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*, 2021, pp. 1–5.
- [34] 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*, vol. 85529. American Society of Mechanical Engineers, 2021, p. V001T07A004.
- [35] L. Zhu, Y. Fu, R. Chow, B. F. Spencer, J. W. Park, and K. Mechitov, "Development of a high-sensitivity wireless accelerometer for structural health monitoring," *Sensors*, vol. 18, no. 1, p. 262, 2018.
- [36] 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*, vol. 388, pp. 154–170, 2017.

- [37] 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*, vol. 34, no. 9, pp. 774–789, 2019.
- [38] 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," *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 42, no. 1, p. 34, 2019.
- [39] 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*, vol. 23, no. 2, p. 025014, 2013.
- [40] 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*, pp. 1–17, 2020.
- [41] HiveMQ. (2019) HiveMQ documentation v4.5. [Online]. Available: <https://www.hivemq.com/docs/hivemq/4.5/user-guide/introduction.html>
- [42] K. D. Team. (2020) Developers guide documentation. [Online]. Available: <https://keras.io/#why-this-name-keras>
- [43] scikit-learn developers. (2017) scikit-learn user guide. [Online]. Available: [https://scikit-learn.org/0.18/\\_downloads/scikit-learn-docs.pdf](https://scikit-learn.org/0.18/_downloads/scikit-learn-docs.pdf)
- [44] STMicroelectronics. (2008) LIS344ALH datasheet. [Online]. Available: <https://www.st.com/resource/en/datasheet/lis344alh.pdf>
- [45] —. (2016) HTS221 datasheet. [Online]. Available: <https://www.st.com/resource/en/datasheet/hts221.pdf>
- [46] Z. Cui, F. Li, and W. Zhang, "Bat algorithm with principal component analysis," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 3, pp. 603–622, 2019.
- [47] X. Zhao, W. Lin, and Q. Zhang, "Enhanced particle swarm optimization based on principal component analysis and line search," *Applied Mathematics and Computation*, vol. 229, pp. 440–456, 2014.
- [48] P. Yang, C.-J. Hsieh, and J.-L. Wang, "History pca: A new algorithm for streaming pca," *arXiv preprint arXiv:1802.05447*, 2018.
- [49] 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*, 2020.
- [50] C. S. N. Pathirage, L. Li, W. Liu, and M. Zhang, "Stacked face denoising auto encoders for expression-robust face recognition," in *2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 2015, pp. 1–8.
- [51] 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*, vol. 172, pp. 13–28, 2018.
- [52] F. Di Nuzzo, D. Brunelli, T. Polonelli, and L. Benini, "Structural health monitoring system with narrowband iot and mems sensors," *IEEE Sensors Journal*, vol. 21, no. 14, pp. 16 371–16 380, 2021.
- [53] 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)*. IEEE, 2015, pp. 158–162.
- [54] 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*, pp. 1–15, 2020.
- [55] H. Wang, A. Jasim, and X. Chen, "Energy harvesting technologies in roadway and bridge for different applications—a comprehensive review," *Applied energy*, vol. 212, pp. 1083–1094, 2018.
- [56] Z. Juhasz, "Quantitative cost comparison of on-premise and cloud infrastructure based eeg data processing," *Cluster Computing*, pp. 1–17, 2020.
- [57] J. Su, Y. Xia, and S. Weng, "Review on field monitoring of high-rise structures," *Structural Control and Health Monitoring*, vol. 27, no. 12, p. e2629, 2020.
- [58] 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*, p. 1475921720924601, 2020.
- [59] 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*, vol. 13, no. 20, p. 5338, 2020.
- [60] E. P. Carden and J. M. Brownjohn, "Arma modelled time-series classification for structural health monitoring of civil infrastructure," *Mechanical systems and signal processing*, vol. 22, no. 2, pp. 295–314, 2008.
- [61] M. Gul and F. N. Catbas, "Damage assessment with ambient vibration data using a novel time series analysis methodology," *Journal of Structural Engineering*, vol. 137, no. 12, pp. 1518–1526, 2011.
- [62] 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*, vol. 16, no. 12, pp. 7802–7811, 2020.

## AUTHORS



toring systems.

**Amirhossein Moallemi** received his B.Sc degree in Electrical Engineering - Electronics at Zanjan University, Iran, in 2017 and M.Sc degree (cum laude) in Electronic Engineering at the University of Bologna, Italy, in 2020. He is currently working on his Ph.D. degree at the Department of Electrical, Electronic and Information Technologies Engineering (DEI) of the University of Bologna, Italy. His research interests include IoT, low-power hardware and firmware design for embedded systems, machine and deep learning models, Structural Health Monitoring systems.



**Alessio Burrello** received his B.Sc and M.Sc degree cum laude in Electronic Engineering and Embedded Systems at the Politecnico di Turin, Italy, in 2016 and 2018. He is currently working toward his Ph.D. degree at the Department of Electrical, Electronic and Information Technologies Engineering (DEI) of the University of Bologna, Bologna, Italy. His research interests include parallel programming models for embedded systems, machine and deep learning models, hardware oriented deep learning, and code optimization for multicore systems.



**Davide Brunelli** received his M.S. (cum laude) and Ph.D. degrees in electrical engineering from the University of Bologna, Italy, in 2002 and 2007, respectively. He is currently an associate professor at the University of Trento, Italy. His research interests include IoT and distributed lightweight unmanned aerial vehicles UAV, the development of new techniques of energy scavenging for low-power embedded systems and energy-neutral wearable devices. He was leading industrial cooperation activities with Telecom Italia, ENI, and STMicroelectronics. He has published more than 200 papers in international journals or proceedings of international conferences. He is an ACM member and a senior IEEE member.



**Luca Benini** holds the chair of digital Circuits and systems at ETHZ and is Full Professor at the Università di Bologna. He received a PhD from Stanford University. He served as chief architect in STMicroelectronics France. Dr. Benini's research interests are in energy-efficient parallel computing systems, smart sensing micro-systems and machine learning hardware. He has published more than 1000 peer-reviewed papers and five books. He is a fellow of the ACM and a member of the Academia Europaea.