Combining Compressed Sensing and Neural Architecture Search for Sensor-Near Vibration Diagnostics

Edoardo Ragusa , *Member, IEEE*, Federica Zonzini , *Member, IEEE*, Paolo Gastaldo , and Luca De Marchi , *Member, IEEE*

***Abstract*—Compressed sensing (CS) for sensor-near vibration diagnostics represents a suitable approach for the design of network-efficient structural health monitoring systems. This article presents a solution for vibration analysis based on deep neural networks (DNNs) trained on compressed data. The envisioned maintenance system consists of a network of sensing nodes orchestrated by a very constrained centralizing unit. The latter is equipped with a microcontroller unit (MCU) that predicts the health state using the aggregated information. As a major contribution, the DNN architectures are generated automatically from the data through a procedure inspired by hardware-aware (HW) neural architecture search (NAS), called as HW-NAS-CS, which is uniquely refined with additional constraints that consider both the peculiarities of CS parameters and the limitation of embedded devices. The proposed approach has been validated using two real-world SHM datasets for vibration damage identification and eventually deployed on a low-end computing platform (the STM32L5 MCU). Results demonstrate that DNNs combined with adapted CS schemes can attain classification scores always above 90% even in case of very huge compression levels (higher than 64x): these performances significantly improve the ones attained by state-of-the-art approaches in the field, with the utmost advantage of being portable on embedded devices.**

***Index Terms*—Compressed sensing (CS), neural architectural search (NAS), tiny convolutional neural networks (CNNs), vibration-based diagnostics.**

Manuscript received 1 June 2023; revised 30 October 2023 and 6 March 2024; accepted 17 April 2024. This work was supported in part by PNRR—M4C2—Inv. 1.3, PE00000013 - “FAIR” project—Spoke 8 “Pervasive AI,” funded by the EU under the NextGeneration EU programme. Paper no. TII-23-1977. *(Edoardo Ragusa and Federica Zonzini contributed equally to this work.) (Corresponding author: Edoardo Ragusa.)*

Edoardo Ragusa and Paolo Gastaldo are with the Department of Electrical, Electronic, Telecommunications Engineering, and Naval Architecture (DITEN), University of Genoa, 16126 Genova, Italy (e-mail: edoardo.ragusa@unige.it; paolo.gastaldo@unige.it).

Federica Zonzini and Luca De Marchi are with the Department of Electrical, Electronics, and Information Engineering (DEI) and Advanced Research Center on Electronic Systems (ARCES), University of Bologna, 40126 Bologna, Italy (e-mail: federica.zonzini@unibo.it; l.demarchi@unibo.it).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2024.3395648.

Digital Object Identifier 10.1109/TII.2024.3395648

I. Introduction

THE integrity condition of technical assets is constantly jeopardized by aging, environmental, and/or man-made hazards. As such, it is not surprising that building “resilient infrastructure, promoting inclusive and sustainable industrialization and fostering innovation” has been stated as one of the sustainable development goals to be attained by 2030.[1]

Permanently installed structural health monitoring (SHM) systems can provide a viable means in the pursuit of this objective, paving the way to the development of *smart structures* [1], [2], i.e., facilities equipped with intelligent sensors capable to perform advanced data analytics in a self-contained manner. Intelligent-embedded systems can indeed revolutionize the field thanks to their low-power and sensor-near functionalities, with the promise of overcoming the limitations of current cloud-based solutions in terms of costs and long-term performances. The latter can be reached via ad hoc data management/processing policies running on edge/extreme edge nodes, such as those implementing data compression that is mandatory to handle the potential data deluge problem [3].

To address the aforementioned challenge, this article presents a solution based on smart sensing nodes placed on different areas of the structure under analysis. These extreme edge sensors collect, compress, and transmit structural measurements, which are then forwarded to a low-end centralizing node featuring a resource-constrained microcontroller unit (MCU). The latter is in charge of data aggregation and damage identification directly from compressed data. Accordingly, the goal is to maximize the accuracy of the health assessment process while minimizing the amount of information transmitted over the monitoring network. The transmission of data toward the aggregation unit impacts, in fact, the network's energy consumption [4]. Placing the aggregation unit far from the sensors implies power-hungry long-range communications, encouraging configurations where the centralizing node is as close as possible to the sensing devices. As a consequence, computing constraints similar to the ones proper of sensing nodes hold for the aggregator that should be hosted on energy-efficient and battery-supplied low-end computing platforms. Eventually, this positively reduces the overall cost of the system too.

[1][Online]. Available: https://sdgs.un.org/goals

Despite being highly desirable, there are still some limitations hampering the full exploitation of data reduction algorithms in SHM systems: these are mostly linked to the fact that all the classification methods available to date strongly depend on structural parameters, which can only be estimated once the original, uncompressed time series are reconstructed. This is the case of vibration-based monitoring, that is the typical inspection methodology for structures in dynamic regime (e.g., mechanical rotors, bridges, residential buildings, wind turbines, etc.), where the structural status is assessed by extracting the so-called modal parameters (e.g., natural frequencies) [5]. Nevertheless, this modal-dependent approach inherently implies several drawbacks, primarily linked to the following facts:

1) it introduces much longer computation time due to the need of decompressing the acquired time-series, a procedure that usually involves solving complex optimization problems not apt for extreme edge deployment [6];
2) the compression rate should be kept low to ensure sufficient quality in the reconstructed structural features.

Consequently, different alternatives capable of retrieving health indicators directly in the compressed domain must be developed to bypass these issues.

Deep neural networks (DNNs) could offer an optimal solution for data mining from compressed coefficients, thanks to their generalization and learning capabilities circumventing the limitations of standard statistical approaches. The use of DNNs has the potential to discriminate damaged from healthy configurations [7] by learning the pattern hidden in the residual coefficients even if the original time information is not fully preserved; in turn, higher compression rates could be attained. Nevertheless, selecting the best-performing DNN model is a function of the problem under analysis and requires advanced skills [8]. Tight constraints imposed by embedded devices further complicate the design process. Unfortunately, it is impossible to set *a priori* the best architecture. Currently, neural architecture search (NAS) represents the most reliable solution to address this issue. NAS automatizes the procedure of designing a DNN for a given task, thus avoiding time-consuming and error-prone processes.

This article explores the interaction between compressed sensing (CS) and DNNs, providing a novel CS-informed strategy for DNN-driven vibration monitoring at the edge. The approach is strengthened by a suitable NAS design procedure that, when needed, automates the network definition using precious domain-specific information. In particular, the proposed approach selects simultaneously the CS settings and the network architecture, jointly optimizing the two most important sets of parameters for the envisioned application.

Thereby, the main contribution of this work can be summarized as follows.

1) We propose a novel approach to sensor-near vibration-based SHM, demonstrating that vibration inspection can be performed directly in the compressed domain. Hence, it is possible to overcome the need to reconstruct time-domain signal, as required by state-of-the-art CS-driven methodologies.
2) To pursue the goal in 1), we resort to a DNN framework built on convolutional neural networks (CNNs), which requires CS-processed vibrations as input and returns a binary classification (i.e., safe versus damaged structure) as output.
3) We introduce a novel hardware-aware (HW)-NAS strategy exploiting the properties of CS, called as HW-NAS-CS, to select the most effective CNN solution for vibration data processing at the edge. Note that, to the best of the authors' knowledge, this is the first time in which NAS is used in conjunction with CS for SHM applications.
4) We thoroughly validate the performances of the devised workflow on two well-known benchmarks for vibration monitoring, reaching state-of-the-art classification scores (i.e., compression level as high as 64x and classification metrics always above 90%).

The rest of this article is organized as follows. The theoretical background behind CS and NAS is outlined in Section II, while Section III depicts an overview of the state-of-the-art. In Section IV, the core of the novel HW-NAS-CS framework is described, while its validation on two relevant SHM bridge use cases can be found in Section V. Results are extensively discussed in Section VI; and Finally, Section VII concludes this article.

II. Preliminaries

In this Section, basic concepts instrumental to the algorithmic solutions adopted in this work are provided.

A. Compressed Sensing (CS)

CS is a compression strategy that relies on the assumption that the information to be processed is sparse in a given representation domain [9]. This condition is typically valid for vibration data, since the spectral profile is dominated by a reduced batch of components located at specific frequencies [10]. Let us suppose that most of the overall vibration energy is captured, once the signal is projected in the Fourier domain, by the topmost k spectral values: in this case, an N-long signal x is defined to be k-sparse. Then, the CS theory states that the informative content can be collapsed in a vector $\hat{x}$ of dimensions $M \ll N$. The latter can be obtained by selecting of a proper compression matrix $\Phi \in \mathbb{R}^{M\times N}$, via a simple matrix-vector multiplication: $\hat{x} = \Phi x$.

Despite its simple algebraic formulation, a critical step in CS implementation is the selection of the optimal sensing matrix. This operator can have a crucial impact on the quality of the retrieved temporal information after completing the recovery procedure. The reason is related to the lossy nature of CS, which ensures accurate, but still not perfect, reconstruction upon the satisfaction of tight constraints on the analytical properties of the compression scheme. To this end, multiple strategies have been proposed, which can be categorized into two main groups: nonadaptive and adaptive methods [11]. The former aim at designing Φ from a normal Gaussian distribution, and hence, are also known as random compression (RND); the latter approach the problem from a physics-informed perspective, meaning that *a priori* information about the sparsity pattern of the signals is exploited for better adaptation to the target scenario.

B. Neural Architecture Search (NAS)

NAS is an optimization problem where an algorithm sets the architecture of a DNN automatically using labeled data [12], [13]. Given a search space $\mathcal{A}$ that sets the admissible candidates and a dataset $\mathcal{D}$, the goal is to find an architecture $a \in \mathcal{A}$ that scores the lowest validation loss $\mathcal{L}_{\text{val}}$ when trained using dataset $\mathcal{D}$.

Beside $\mathcal{A}$, dedicated search algorithms $\mathcal{S}$ are used to drive the selection process, i.e., the approaches to the optimization problem, since a complete exploration of $\mathcal{A}$ is computationally impossible. To this end, the formulation of the optimization problem, and the specific metrics used to evaluate the best architecture, also called evaluation criteria $\mathcal{EC}$, are used to compare different instances; eventually, they lead to the optimal model among all the possible realizations of $\mathcal{S}$.

For example, HW-NAS involves specifically designed optimization problems [12] that reflect the limitations imposed by the target devices. HW-NAS can use specialized loss functions $\mathcal{L}$ as evaluation criteria, or include additional constraints $\psi(a, \text{HW})$ that model the requirements of a when deployed on the hardware HW as follows:

$$\begin{aligned} \min_{a \in \mathcal{A}} \quad & \mathcal{L}_{\text{val}}(w^*(a), a) \\ \text{s.t.} \quad & w^*(a) = \text{argmin}_w \mathcal{L}_{\text{train}}(w, a) \\ & \psi(a, \text{HW}) < \text{Thr} \end{aligned} \tag{1}$$

where w is the set of weights optimized during the training procedure and Thr are the thresholds that highlight the limitation of the device and $\mathcal{L}_{\text{train}}$ is the training loss.

III. Related Works

A. Pattern Recognition From Compressed Data

Systematic methodologies capable of mining information directly in the compressed domain are still an open field of research. The literature accounts just a few attempts, which are primarily built on conventional Eigen-based approaches combined with residual thresholding mechanisms, either exploiting high-order statistical metrics or low-level energy features. Successful evidence of such methods is provided, for example, in [14], in which authors performed anomaly identification by tracking outliers in the magnitude of the total residual signal energy yielded by projecting CS-processed data onto a lower dimensional subspace. The motivation supporting this is that, in a common subspace-based framework, the residual should capture noise-related features, whereas the effect of anomalies usually translates in a different distribution of the spectral properties of the signal, but keeping unaltered its overall power. Therefore, a small variation in the principal subspace should cause a corresponding change in the residual energy. This strong assumption, however, should be verified case by case [15]. In addition, residual thresholding mechanisms exhibit poor robustness against noise and rounding effects.

A comprehensive selection of methods for pattern recognition from compressed measurements is offered in [16] focusing on both unsupervised and supervised solutions, and exploring the potential of artificial intelligence (AI) strategies built on fully connected neural networks. Despite reaching good prediction capability, the results obtained in this study cannot be generalized because only one type of abnormalities was considered; moreover, performance of the detectors was tested on numerically simulated data that cannot reproduce completely the variability induced by real-field operational conditions. Alternatively, the effectiveness of wavelet packet transform applied on compressed signals was shown in [17], but on a different application scenario related to through-wall human detection.

In SHM-oriented scenarios, there is a lack of practical evidence about the applicability of the mentioned methodologies. A CNN was applied in the context of a vision-based monitoring dataset for the binary classification of defects from vibration data, compressed in the form of discrete histograms [18]. The main problem of this approach is due to the nature of the processed time series, which require costly and high precision vision sensors, incompatible with low cost installations. A recent work [19] addressed damage identification using statistical features of the compression coefficients for vibration data to reduce communication bandwidth between sensing nodes and the aggregation unit. This method led to smaller payloads at the expense of a slight deterioration of the overall accuracy of the system and the impossibility to reconstruct the original signals.

B. Tiny Deep Networks

The deployment of deep learning algorithms at the extreme edge requires ad hoc optimization strategies [20], [21]. Currently, the state-of-the-art is based on software-hardware cooptimization [22]. This brings the design strategies to a new level, where resource-constrained nodes locally mine sophisticated information [23]. The latest research trend uses NAS to automatically select the DNNs architecture [24]. The literature, indeed, offers specialized search spaces [25] based on the target application. The major limitation in the application of NAS is the computational cost of the search procedure. Supernetworks that contain all the possible architectures envisioned by a search space [26], [27] represent a promising method. However, it can prove suboptimal when compared with iterative strategies [12].

After the selection of the DNNs architecture, the deployment on the embedded devices requires additional precautions [21]. Different devices do not necessarily achieve the same latencies when implementing networks with the same number of Flops and parameters [28] because they support different software inference engines. ARM microprocessors, for example, benefit from optimized libraries for inference [29], biasing the performance of optimization techniques [30]. When hardware resources fail to support optimization techniques, they can even worsen the performance. For example, quantization can slow down models run on MCUs if the instruction set fails to support this representation [31]. To address such challenges, the MCUnet uses a unique optimization procedure for the selection of the architecture and the setup of the computing layer, obtaining excellent scores [32], [33], [34]. Unfortunately, the custom software layer makes it difficult to use and customize the model [35].

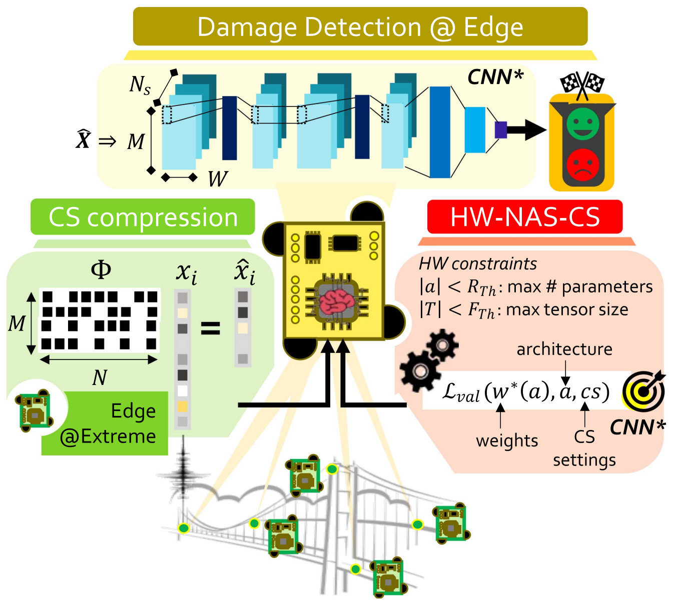

Fig. 1. Proposed framework for vibration-based damage identification from compressed data.

The proposed approach differs from state-of-the-art solutions as it includes CS parameters in the optimization problem; in turn, this leads to an adaptive search space. Eventually, one can significantly improve the tradeoff between the performance of the generated networks and their computational cost by exploiting the peculiarities of CS.

IV. Proposal: Joint Design of DNN and Signal Compression

A. Monitoring Framework

This article targets the hierarchical monitoring system depicted in Fig. 1, in which several extreme edge sensors (i.e., sensing units) are divided into a few clusters, each of them mastered by an edge unit featuring a low-end computing platform built on an MCU. Even if, in principle, more powerful computing capabilities could be allocated to the aggregator, a resource-constrained platform has been selected to account for worst-case inspection scenarios in which one of the extreme edge nodes can act as a centralizing unit. This solution offers the following two additional advantages:

1) low-cost, hence promoting large scale installations with a minimal economic impact on the structural assessment process;
2) ultralow-power consumption, thus reducing the power budget and favoring prolonged sensor life cycle, which is mandatory for long-term analysis.

Noteworthy, this network topology is preferable over purely centralized and cloud-based schemes, since the latter can barely deal with data flooding while suffering from poor scalability, modularity, and security issues [36].

Let us suppose that every cluster includes N_s devices. In this case, each centralizing unit receives W consecutive acquisitions of CS-processed measurements $\hat{x}_{i(i=1,\ldots,N_s)} \in \mathbb{R}^{M\times W}$, which are globally aggregated in a tensor $\hat{X} = [\hat{x_1}, \ldots, \hat{x_{N_s}}] \in \mathbb{R}^{M\times W\times N_s}$ before being passed to the DNN implementing structural health assessment. Accordingly, W consecutive messages, each of them consisting of M compressed coefficients, are transmitted from N_S extreme edge sensors to the centralizing unit. This corresponds to a data bandwidth, which is N/M times lower with respect to compression-free scenarios.

While the clustering scheme is of paramount importance in view of defining the best sensor arrangement which can meet the quality of service in terms of electrical connectivity and mechanical constraints, this aspect is expected to have limited impact on the inference stage. Indeed, vibrations reflect the macroscopic behavior of the target structure (or of its major building blocks), whose dynamic response is uniquely determined by the geometrical and material properties, which are common to all the measurement points. Coherently, in the most simplistic scenario, the same DNN can be deployed on different clusters. Even if doable, this solution might-be suboptimal in those circumstances in which sensors are installed on poorly sensitive locations: in these cases, a better choice would be to design a preliminary DNN model, and then, fine tune it on a cluster basis taking into account the peculiarities of the associated sensors.

B. HW-NAS-CS Design

In this setup, two major aspects bias the selection of the best architecture a, henceforth referred to as CNN*. First, the flash and RAM memory available on the computing device limits the set of solutions because low-end MCUs feature memory sizes in the order of a few hundred of Kilobytes. The inference time could be more relaxed due to the slow duty-cycles typical of vibration monitoring, where inspection is performed on an hourly basis. Second, CNN* is a function of the compression parameter cs (i.e., the size of the sensing matrix), since higher levels of compression could require more demanding configurations, making the solution nontrivial when HW constraints are involved. To model these aspects, the optimization problem in (1) has been updated as follows:

$$\begin{aligned} \min_{a\in\mathcal{A}, cs\in\mathcal{CS}} \quad & \mathcal{L}_{\text{val}}(w^*(a), a, cs) \\ \text{s.t.} \quad & w^*(a, cs) = \text{argmin}_w \mathcal{L}_{\text{train}}(w, a, cs) \\ & |a| < F_{\text{Th}} \\ & |T| < R_{\text{Th}} \end{aligned} \tag{2}$$

where the two constraints model memory requirements, while $\mathcal{CS}$ remarks the role of compression settings. $|a|$ is defined as the number of parameters of the architecture a, while $|T|$ is the number of elements of the largest tensor propagated through the architecture. Network parameters are constant and saved on the flash memory, whose constraint is indicated as F_{Th}. These values require the largest portion of data memory to store a DNN. Conversely, tensor values T are computed at runtime as a function of the specific input, therefore, they have to be stored in the RAM memory of maximum size R_{Th}. Given the sequential nature of DNNs, memory reusage strategies allow sizing RAM to be proportional to the amount of memory required to store the largest tensor.

The definition of $\mathcal{A}$ heavily influences the global performances. On the one hand, too broad search spaces lead to long search times and may produce suboptimal results. On the other hand, very tight search spaces induce bias that excludes valuable solutions. For the target application, geometrical information about the input data can be exploited to reduce the search space without compromising the generality of $\mathcal{A}$. For this reason, it is advisable to focus on CNNs, which can merge and aggregate local and global information. In principle, multidimensional convolutions could be employed. However, for the specific application context considered in this work, it is expected that anomalies are strongly associated with changes in the relative amplitude and distribution of the compressed coefficients over time: for this reason, 1-D convolutional kernels applied along the time dimensions would be most suited for implementation of the CNNs in $\mathcal{A}$, being the most compatible with the physical nature of the observed structural dynamics.

The compression setting cs plays a major role in the task addressed by the CNN because it sets the amount of processed information: reducing the compression level dramatically increases RAM requirements, since the dimension of the input features increases; in fact, the size of the input tensor in single branch tiny DNNs is typically the bottleneck for RAM consumption leading to the violation of constraint R_{Th} for small values of compression. This constraint, in practice, has a positive effect on the overall amount of energy spent in communication, penalizing implicitly the transmission of large size arrays. In addition, a smaller input size promotes a reduced number of parameters, for example, in architectures where fully connected layers are included; in turn, this means increasing the number of architectures fulfilling constraint F_{Th}. Conversely, increasing cs could lead to larger networks to compensate for the loss of information, and in some cases, it could lead to lower accuracy. For this reason, in the proposed solution, cs parameters are encapsulated into the network definition treating them as an architectural parameter.

These observations led to a block-wise search space composed of building blocks with a 1-D convolution layer possibly followed by a max pooling layer and a dropout regularization. Accordingly, each building block admits four parameters, namely the number of filters N_f, the kernel size K_s, pooling operations $Pool$, and dropout operations Dr. The number of blocks N_b, their configuration, and cs quantities set the complete architecture. A standard evolutionary algorithm as the one implemented in Algorithm 1 allows a wide exploration of the search space. It explores the possible combination of the pairs (a, c_s) starting from a preset initial configuration (a_0, c_0). Then, the procedure iteratively generates a set of new pairs (a_c, c_c) by applying random mutations to the parent architecture. Finally, for all pairs, a training procedure generates a classifier: eventually, the network that scores the best result on the evaluation procedure becomes the new parent architecture. The procedure ends when the preset number of iterations is reached.

Algorithm 1: Evolutionary NAS With CS Settings Selection.

```
Dataset Training set $\mathcal{X}_{\mathcal{T}} = \{X_i, y_i\}_{i=1\ldots n}$ with $X_i$ raw
  data and $\mathbf{y_i}$ the corresponding health state.
Validation set $\mathcal{X}_{\mathcal{V}} = \{X_i, y_i\}_{i=1\ldots m}$.
Hypothesis Architectures search space $\mathcal{A}$, Compression
  settings $\mathcal{C}_f$, Parent configuration $(a_p, c_p)$, Child
  generator function $(a_c, c_c) = R_m((a_p, c_p))$, $\mathbf{N_c}$ number
  of Child architectures, $\mathbf{N_g}$ number of generations,
  Evaluation function $\mathbf{E}(\mathbf{a}, \mathcal{X}_{\mathcal{V}})$, Compression function
  $C_f(X, c_s)$
Pseudocode
  0. Init: $(a_p, c_p) = (a_0, c_0)$
  for $g$ in $N_g$ do
      for $c$ in $N_c$ do
          1. Mutation: $(a_c, c_c) = R_m((a_p, c_p))$
          2. Training: Train $a_c$ on $C_f(\mathcal{X}_{\mathcal{T}})$
      end for
      3. Selection: $(a_p, c_p) =$
              $argmin(E((a_c, c_c))_{j=1\ldots N_c}, C_f(\mathcal{X}_{\mathcal{V}}))$
  end for
```

V. Experimental Validation

Experimental verification of the proposed methodology was performed on two well-known benchmarks for vibration-based inspections.

A. Dataset Description

1) Z24 Bridge: Built in Switzerland in the early 60s, the Z24 bridge was at the center of an important highway viaduct between Bern and Zürich. The serviceability of the structure was interrupted in 1999 for modernization. Before its demolition, it has been artificially damaged via a rigorous experimental protocol, moving from slight flaws to very severe disruption actions [37]. A large monitoring network consisting of acceleration and environmental sensors was installed and data were gathered over one year: a total amount of 5651 time series was collected, the first 4922 related to healthy conditions, while the remaining ones pertain to the structure under progressive damage tests. Each instance contains data acquired by eight force-balance-type FBA-11 accelerometer sensors by Kinemetrics working at a sampling frequency of 100 Hz (acquisition time of nearly 5 min every hour).

2) KW51 Bridge: Currently in operation, the KW51 is a relatively new steel bowstring railway bridge in Leuven, Belgium. A dataset has recently been released [38], which contains heterogeneous measurements obtained over a monitoring period of 15 months (from October 2018 to January 2020). Even if not threatened by any evident damage, the facility has been retrofitted during the inspection period to resolve a construction error. Coherently, 1787 time series have in total been measured, the first 1239 of which being related to the preretrofitting status. A total of 12 uniaxial accelerometers of type PCB 393B04 was installed, gathering 5 min data at a sampling frequency of 1.6 kHz per hour.

B. Compression Matrix Design

The model-assisted rakeness-based (MRak-CS) strategy [10] has been chosen among the adaptive CS mechanisms for the following two main reasons:

1) it demonstrated better robustness against the perturbations induced by environmental and operational parameters, hence being able to discriminate between real and spurious anomalies;
2) it attained superior performances, i.e., a deeper compression level with negligible loss in the accuracy of the reconstructed damage sensitive features.

More in detail, MRak-CS forces an analytical bandpass-like correlation profile to the sensing matrix, with spectral bands centered around the expected dominating modes of the structure [39].

For the two considered bridges, the spectral characteristics discussed in [37] and [38] were exploited to design Φ according with the MRak-CS procedure. A total of four frequency components were identified for the Z24 bridge in the $[0, 20]$ Hz band (i.e., $k_{\text{Z24}} = 4$), while previous works suggest that $k_{\text{KW51}} = 12$ is reasonable for the KW51. Finally, $N = 512$ has been selected for the window size. Note that, this quantity cannot be arbitrarily increased since it strongly impinges on the dimension of the compression matrix ($\Phi = O(N^2)$), hence rapidly saturating all the memory slots available in the sensing units. In this work, we have considered also a nonadaptive RND alternative. In this case, the entries of the sensing matrix were randomly sampled from a multivariate Gaussian distribution.

C. NAS-CS Training

The code was implemented in Python using Keras and Tensorflow libraries. A standard fivefold cross validation was considered during the experiments. For each fold, a validation set was extracted from the training set selecting a random subset of 20% of the training data.

All the architectures were trained for a maximum of 100 epochs with an initial learning rate of 10^{-3}, the learning rate reduction on the plateau, and early stopping using the validation loss as metric. In addition, all the networks were trained five times using a multistart approach. The best architecture was selected based on the validation set.

The NAS executed 500 generations, using a search space with tight constraints, which are necessary to derive the most effective CNN architecture in compliance with typical MCU storage capabilities: a maximum flash size of 512 kB and RAM size of 256 kB have been imposed, respectively. Due to the small size of the tested networks, local minima can affect generalization performance, and therefore, standard proxy training strategies fail to provide reliable descriptions of the network's behavior. Each generation was composed of ten children obtained from a mutation of the parent network and a change in the compression settings. Random mutation function considered the insertion, deletion, or modification of one block of the network. The modification was performed through the change of the value of one of the block parameters. Compression settings were changed acting on the compression rate $\text{CR} = N/M$, keeping constant the window size.

The resulting CNNs* were finally deployed on the target board, i.e., the STM32L522ZE-Nucleo board, which hosts an STM32L5 MCU based on the ARM Cortex-M33 core. This device was selected for prototyping purposes since it offers valuable features in terms of power management, security, and digital processing functionalities, making it a class-leading microprocessor for embedded applications. Deployment was achieved via the following pipeline. First, the network was converted into a MCU-compliant format by means of the API made available by the TensorFlow Lite library;[2] then, the STM32 X-Cube-AI suite was exploited to optimize the model. Data representation was set to 32-bit because STM32 X-Cube-AI supports 8-bit representation only for fully connected layers. Therefore, one can consider that the measured performance corresponds to the worst-case analysis considering that quantization can additionally reduce memory requirements. All the measurements necessary for the DNN assessment were performed by using the STM32 design suite utility for testing.

D. Evaluation Metrics

The classification performance has been measured for all the models on the testing fold that have never been involved in any parameter or hyperparameter tuning. In all the experiments, accuracy, precision, recall, and F1 have been adopted as metrics. The joint usage of the four scores is a well-established procedure to evaluate classifiers without the risk of pathological measures that could happen when basing the analysis only on one metric. For example, when using an unbalanced dataset, a classifier could exhibit very high accuracy paired with unsatisfactory precision or recall [40].

VI. Results

Results are reported in this section, pursuing the following four objectives.

1) Prove that 1-D convolution through time is the most appropriate form of convolution for the CNN architecture (see Section VI-A).
2) Assess the damage identification capability of the CS-based CNN detector as a function of increasing CR and of the adopted compression scheme (MRak versus RND) (see Section VI-B).
3) Demonstrate the superiority of the identified HW-NAS-CS configuration (CNN*) in terms of classification scores and model complexity when compared to existing methods (see Section VI-C).
4) Test and verify the performances of CNN* when deployed on a general-purpose MCU (see Section VI-D).

A. Search Space Analysis

The optimal convolution type has been investigated first to restrict the loci of viable architectures to be considered during

[2][Online]. Available: https://www.tensorflow.org/lite?hl=it

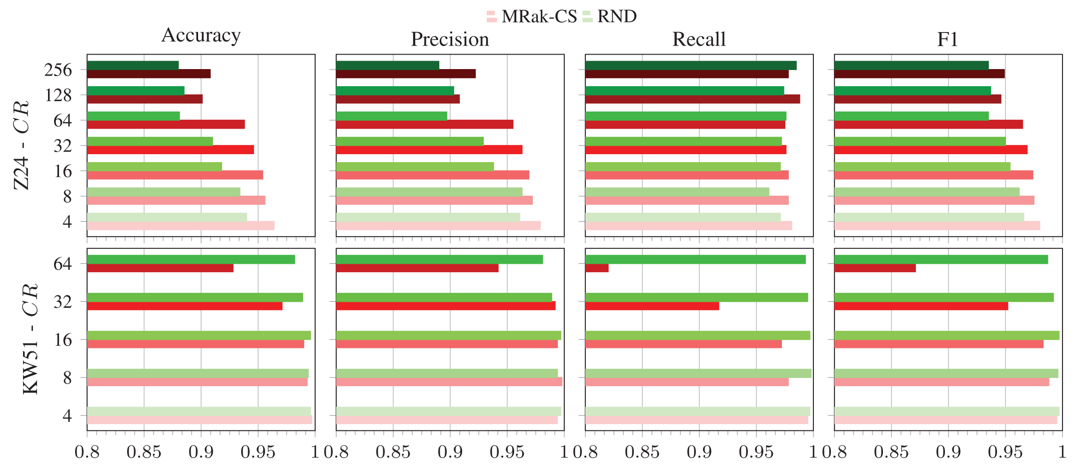

Fig. 2. Classification scores for various CR and different sensing matrix (MRak-CS, red-shaded bars; RND, green-shaded histograms) when processing the Z24 (first row) and the KW51 (second row) data.

TABLE I
PERFORMANCE OF CNNS WITH DIFFERENT VERSIONS OF CONVOLUTION TYPE APPLIED ON THE Z24 AND KW51 DATASET COMPRESSED VIA THE MRAK-CS MECHANISM WITH CR $= 4$

Dataset	**Conv. type**	**Accuracy**	**Precision**	**Recall**	**F1**
Z24	2D	0.869	0.873	0.994	0.930
	Coeff	0.868	0.873	0.992	0.930
	Time	0.964	0.979	0.981	0.980
KW51	2D	0.845	0.826	0.989	0.898
	Coeff	0.800	0.782	0.994	0.873
	Time	0.998	0.997	0.999	0.998

the search space analysis. To this end, data from the Z24 and KW51 structure were processed, keeping constant both the CR (equal to 4) and all the remaining hyperparameters of the CNN, except from the kernel size and direction. To maximize the role of the convolution, a baseline architecture has been selected, hereinafter BASE_CNN, which consists of the following blocks: a first convolutional layer with five filters and max pooling, receiving aggregated compressed data $\hat{X}$, a second convolutional layer with three filters followed by global average pooling, extracting a second set of features, and a final fully connected layer predicting the (binary) class of the input datum.

Table I summarizes the results given as standard classification scores (accuracy, precision, recall, and F1) when using MRak-CS compression at the sensing stage. The upper part of the table summarizes results for Z24 data, while the lower depicts the results for KW51. The outcome confirms that structural information is extracted by convolutions over the time dimension, i.e., by training the network to recognize specific trends and patterns in the values of the compressed coefficients. Proof is the fact that all measures for Z24 (apart from recall where a slight reduction can be noticed) largely exceeds 96%, which is at least 10 points percentage higher than those attained by 2-D convolution or 1-D convolution along the coefficient dimension. Similarly, results on KW51 prove the advantage of convolution through time. Accordingly, 1-D CNN with convolution over time will be used in all the remaining analyses.

B. Compression

The impact of the compression rate has then been evaluated to verify the capability of the network to discriminate healthy from damaged patterns. This goal has been fulfilled by measuring the classification metrics for different values of CR, while using the same BASE_CNN architecture described in the previous section. Results are summarized in Fig. 2 for the two distinct CS schemes: MRak-CS and RND with red- and green-shaded bars, respectively.

1) Z24: The reported scores (first block row) show that classification remains considerably above 90% for all the metrics even when CR increases significantly above the theoretical boundaries dictated by the CS theory, which prescribes $M = O(k\log(N))$ to accurately preserve spectral data: assuming $k_{Z24} = 4$ and $N = 512$ as stated before, $\text{CR}^{\max}_{Z24} < 21$ is implied. However, our outcome proves that the proposed DNN solution could yet return accurate predictions even when this limit is overcome. This is ensured by the superior learning abilities of CNN in understanding patterns from residual information, i.e., when not all the structural properties are retained after compression. More in detail, when CR increases considerably over this theoretical bound, the performance deteriorates by 3%, still remaining acceptable ($> 90\%$) even for the extreme case of CR $= 256$.

In addition, it is worth mentioning that the adopted sensing matrix can play a significant role, as demonstrated by the averagely higher performance indicators of the MRak-CS methods with respect to RND, which is a consequence of the inherent information-preserving nature of the adapted rakeness-based strategy.

2) KW51: The same setup was replicated for the KW51 bridge use case, properly adapting the MRak-CS to the spectral signature of the novel bridge, while doubling the CR at integer steps from 2 to 64. Results are depicted in the second block row of Fig. 2 and reveal that the CNN-based solution can obtain accurate classification scores for compression rates higher than the theoretical boundary, which imposes $\text{CR}^{\max}_{\text{KW51}} = 6$. This discrepancy between the theoretical compression depth for the two bridges is mostly related to the much denser spectral profile ($k_{\text{KW51}} = 12$ versus $k_{\text{Z24}} = 4$) of the KW51 bridge in the frequency band of interest, as a consequence of its geometrical (material, span length, etc.) properties. Both MRak-CS and RND attain very high accuracy at moderately low compression levels (CR < 16), without significant differences between the two compression approaches. Contrariwise, for the deepest CR, the performance of all networks deteriorates and is more pronounced for the MRak-CS setting. This outcome can be justified by the fact that, compared to the Z24 use case whose spectral signature is dominated by four well-spaced modes, the KW51 asset is characterized by the presence of tightly-coupled vibration components, which impose a flatter (i.e., less peak-like) spectral profile of the MRak-CS sensing matrix. In turn, this might lead to a lower adaptation level for this particular kind of spectral profiles, if compared to the totally agnostic RND mechanism.

3) Energy versus Compression: In addition, an analysis of the energy consumption vs compression has been performed from an extreme edge perspective to prove the benefit of on-sensor data compression in industrial Internet of Things (IoT) scenarios. More in detail, the total energy consumption spent by an extreme edge sensor for processing and transmitting data has been measured. To this end, the open source IoT analyzer[3] has been exploited since it allows for a realistic simulation of transmission costs associated with enabling wireless technologies. Specifically, experiments involving the BLE 5.0, LoRa, 802.15.4, and 802.11 ah protocol have been considered. On the computing side, the electrical characteristics of the STM32L5 MCU were selected, supposing monitoring is performed once per hour: 15 mA and 7 μA current consumption in normal operating mode and sleep mode, respectively, a clock frequency of 110 MHz, and a voltage supply equal to 3.3 V.

Results are reported in Fig. 3 (CR varying from 1 to 256) and clearly demonstrate that the energy demand decreases for increasing compression levels. This energy gain is particularly pronounced for the LoRa and the 802.11 ah protocols, which undergo a significant reduction (up to three orders of magnitudes) when moving from compression-free configurations (CR $= 1$, i.e., no compression applied) to very huge data compression (CR $= 256$). Beside, such gain is less pronounced for BLE 5.0, the usage of which does not introduce substantial energy saving for ratios higher than 16. This is due to the specific handshake transmission mechanism, and the power requirement of the individual modules.

C. HW-NAS-CS

1) Z24: Table II summarizes the result of the NAS-generated architecture (first row) with those pertaining to forefront competitors. The columns summarize the model and the possibility to deploy it on the target MCU (gray rows indicate models not portable on embedded devices), the compression rate, and the four classification descriptors. The second and third rows are, instead, referred to the best networks explored in the preceding CR analysis: BASE_CNN with CR $= 4$ is the one yielding

[3][Online]. Available: https://gitlab.imag.fr/morine/iot-analyzer

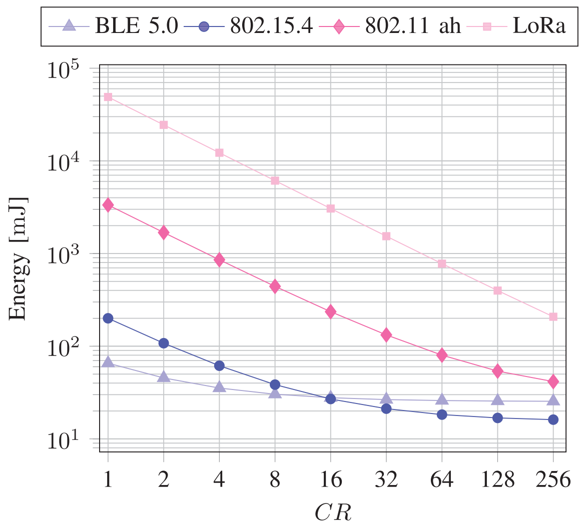

Fig. 3. Total energy consumption in 1-h monitoring under various CR configurations for different IoT technologies: BLE 5.0 (triangle-light blue), 802.15.4 (circle-blue), 802.11 ah (diamond-magenta), and LoRa (squared-pink).

TABLE II
COMPARISON BETWEEN NAS AND OTHER SOLUTIONS FOR DATASET Z24

Model	CR	Deployability	Acc	Prec.	Rec	F1
CNN*-Z24	8	✓	0.967	0.976	0.986	0.981
BASE_CNN	4	✗	0.964	0.979	0.981	0.980
BASE_CNN	8	✓	0.956	0.972	0.978	0.975
OCCNN [39]	6	✓	0.930	0.940	0.950	0.910
ANN [39]	6	✓	0.950	0.990	0.940	0.970
PCA [41]	1	✓	0.620	0.950	0.290	0.450
KPCA [41]	1	✗	0.950	0.990	0.920	0.950
GMM [41]	1	✗	0.950	0.980	0.930	0.950
RF [19]	8	✓	0.940	0.963	0.969	0.956

Models in gray are nondeployable on edge devices.

the best absolute scores but without MCU compatibility, while BASE_CNN with CR $= 8$ indicates the best accurate architecture that can also accommodate the target microcontroller constraints.

Beside, one class classifier neural network (OCCNN) and autoassociative neural network (ANN) are two supervised neural models proposed in [39] for Z24 vibration assessment. These networks were used for damage detection in a framework involving a bulky CS encoding–decoding procedure, according to which data were first compressed, and then, reconstructed in the time domain; once retrieved, damage sensitive features (i.e., frequencies) were extracted, and finally, passed as input to the neural detector. OCCNN is based on a one-class classifier neural network consisting of two fully connected dense layers, while ANN models an autoencoder. RF [19] refers to a recent solution based on the use of a random forest classifier that processes statistical features extracted by the compressed coefficients. As can be observed, the CNN*-Z24 architecture is superior in that it delivers at least 1.1% more accurate results while increasing the compression ratio ($\text{CR}_{\text{NAS-Z24}} = 8$ versus $\text{CR}_{\text{OCCNN/ANN}} = 6$). Compared with other approaches, RF obtains high transmission

TABLE III
SUMMARY OF THE NAS-GENERATED CNN FOR DATASET Z24

Block	K_s	N_f	*pool*	*Dr*	input_size
#1	2	19	✗	✗	64
#2	1	26	✓	✓	64
#3	1	5	✗	✗	32
#4	5	6	✗	✗	32
#5	1	19	✓	✓	32
#6	1	12	✓	✓	16

TABLE IV
COMPARISON BETWEEN NAS AND BASELINE FOR DATASET KW51

Model	CR	**Deployability**	**Acc**	**Prec.**	**Rec**	**F1**
CNN*-KW51	8	✓	0.998	0.998	0.997	0.997
BASE_CNN	4	✓	0.997	0.994	0.995	0.995

efficiency, at the expense of a drop in the generalization performance larger than 2% in terms of accuracy and a complete loss of any residual information about the spectral content of the original data for a compression rate identical to the one of CNN*-Z24. Finally, it is worth highlighting that the proposed solution behaves comparatively better than alternative strategies, such as the principal component analysis (PCA), kernel PCA (KPCA), and Gaussian mixture model (GMM) documented in [41], which can be treated as state-of-the-art unsupervised classifiers for the considered dataset while working with uncompressed data. Our analysis proves that the HW-NAS-CS-driven approach can increment the detection metrics up to 6%, with the uttermost benefit of working with a data payload 8x lower and full MCU-compatibility.

The NAS procedure led to the network with the best results. Noteworthy, even if prediction improvements over BASE_CNN with CR equal 4 are limited, it is paramount to emphasize that CNN*-Z24 can, moreover, be successfully deployed on the target class of low-end computing platforms. In addition, when comparing CNN*-Z24 with the other architectures that meet deployment requirements (OCCNN, ANN, and PCA), the overall gain is significant in almost all indicators.

Details about the DNN architecture obtained through the NAS procedure are given in Table III. Each row is a summary of a building block where we show the kernel size (K_s), the number of filters (N_f), and the presence/absence of max pooling (pool) and/or dropout layers (Dr). The algorithm selected a nontrivial configuration composed of six blocks. A peculiarity is the selection of kernels with very tight sizes, in many cases equal to 1. Eventually, the network deployment requires 2.55 MFlops for a single inference phase, with an estimated memory consumption of 82.740 kB and a RAM peak of 131.072 kB. In addition, CNN*-Z24 is characterized by $CR = 8$ as optimal compression level, hence confirming that the best solution does not necessarily coincide with the smallest CR.

2) KW51: Table IV summarizes the results for dataset KW51. It compares the performance of the CNN*-KW51 model returned by the NAS with the configuration of BASE_CNN reaching the best scores between those explored in the analysis of the compression level.

The NAS procedure converged to a validation score of 1 for all four metrics in only three generations. This result confirms that KW51 is a simpler learning problem than Z24 and the role of the architecture is less relevant to designing effective predictors, but still leads to a slight improvement of 0.2% in the classification capabilities. Notably, this result is a function of the selected search space (see Table I). For example, it is expected that, in case nonoptimal forms of convolution had been chosen (e.g., 2-D convolution), the role of the NAS procedure would have had larger relevance to the final result.

TABLE V
SUMMARY OF THE GENERATED ARCHITECTURE FOR DATASET KW51

Block	K_s	N_f	*pool*	*Dr*	input_size
#1	2	30	✓	✓	24
#2	3	30	✓	✗	12

Indeed, as reported in Table V, the NAS algorithm selected CR = 8 for CNN*-KW51, which is almost double the theoretical limit and that achievable by BASE_CNN, confirming once again that, in the compressed domain, high accuracy can be obtained even when compression introduces important losses in parts of the original signal information. The resulting network is similar to BASE_CNN, coherently with the fact that the algorithm converged after a small number of generations: the final CNN*-KW51 architecture uses 1.2 MFlops for a single inference, with a memory requirement of 103 kB and a RAM usage peak of 37 kB.

D. Deployment

STM-32 Xcube-AI measured a requirement of 81 kB for flash memory and 130 kB for RAM for the Z24 case, which are totally compliant with the memory constraints of the prototyping embedded system. When embedded on the STM32L5 MCU, the deployed model generated for KW51 (CNN*-KW51) was hosted on 101 kB of flash memory and the largest RAM requirement reached 43 kB. In both cases, the prediction performed by the deployed network matched the results obtained on the desktop computer confirming the suitability of the approach.

The verified portability of the algorithmic methodologies on low-end devices is essential in the transition toward the realization of smart structures working in the dynamic regime, which could be endowed with self-diagnostic functionalities and capable of promptly reacting to changes in the surrounding environment. This is the case, for example, of electrical motors that experience a well-defined pattern of rotation, similar to that defining wind blades and other rotating machines.

VII. CONCLUSION

This article presented a novel approach for SHM based on a DNN trained on CS-processed data. In the proposed configuration, a simple microcontroller can act as a centralizing unit and performs the inference phase using compressed information from various sensing nodes. The result was achieved using an NAS-inspired procedure, named after as HW-NAS-CS, that optimized the network architecture based on the peculiarities of CS and the memory limitations of embedded devices. The empirical results scored state-of-the-art performances on two real-world and well-known benchmarks for vibration inspection. The generated architectures were deployed on a STM32L522ZE-Nucleo board. In future works, more complex scenarios dominated by a denser sensor grid will be investigated to analyze the effect of the sensor clustering procedure on the damage detection performance. Alongside, in-depth analysis of the effect of noise on the inference performance of the model will be evaluated in view of real-field installations, which will be expanded to consider alternative industrial settings (e.g., motors and wind turbines).

References

[1] K. Haricha, A. Khiat, Y. Issaoui, A. Bahnasse, and H. Ouajji, "Recent technological progress to empower smart manufacturing: Review and potential guidelines," *IEEE Access*, vol. 11, pp. 77929–77951, 2023.

[2] M. Compare, P. Baraldi, and E. Zio, "Challenges to IOT-enabled predictive maintenance for industry 4.0," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4585–4597, May 2020.

[3] C. Caione, D. Brunelli, and L. Benini, "Distributed compressive sampling for lifetime optimization in dense wireless sensor networks," *IEEE Trans. Ind. Inform.*, vol. 8, no. 1, pp. 30–40, Jan. 2011.

[4] A. Elouali, H. Mora, and F. J. M. Gimeno, "Data transmission reduction model for cloud-based IOT systems," in *Proc. IEEE Int. Conf. Smart Internet Things*, 2021, pp. 252–256.

[5] F. Zonzini, A. Girolami, L. De Marchi, A. Marzani, and D. Brunelli, "Cluster-based vibration analysis of structures with GSP," *IEEE Trans. Ind. Electron.*, vol. 68, no. 4, pp. 3465–3474, Apr. 2020.

[6] Y. Zhang, P. Guo, X. Liu, and K. Zhang, "In-network processing or feature compressive sensing? Case study of structural health monitoring with wireless sensor networks," *IEEE Internet Things J.*, vol. 10, no. 8, pp. 7051–7061, Aug. 2022.

[7] B. Du, C. Lin, L. Sun, Y. Zhao, and L. Li, "Response prediction based on temporal and spatial deep learning model for intelligent structural health monitoring," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13364–13375, Aug. 2022.

[8] J. Long, S. Zhang, and C. Li, "Evolving deep echo state networks for intelligent fault diagnosis," *IEEE Trans. Ind. Inform.*, vol. 16, no. 7, pp. 4928–4937, Jul. 2019.

[9] M. F. Duarte and Y. C. Eldar, "Structured compressed sensing: From theory to applications," *IEEE Trans. Signal Process.*, vol. 59, no. 9, pp. 4053–4085, Sep. 2011.

[10] F. Zonzini, M. Zauli, M. Mangia, N. Testoni, and L. De Marchi, "Model-assisted compressed sensing for vibration-based structural health monitoring," *IEEE Trans. Ind. Inform.*, vol. 17, no. 11, pp. 7338–7347, Nov. 2021.

[11] M. Mangia, F. Pareschi, V. Cambareri, R. Rovatti, and G. Setti, *Adapted Compressed Sensing for Effective Hardware Implementations: A. Design Flow for Signal-Level Optimization of Compressed Sensing Stages*. Berlin, Germany: Springer, 2018.

[12] C. White et al., "Neural architecture search: Insights from 1000 papers," 2023, *arXiv:2301.08727*.

[13] C. Yan et al., "ZeroNAS: Differentiable generative adversarial networks search for zero-shot learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 9733–9740, Dec. 2021.

[14] S. Budhaditya, D.-S. Pham, M. Lazarescu, and S. Venkatesh, "Effective anomaly detection in sensor networks data streams," in *Proc. IEEE 9th Int. Conf. Data Mining*, 2009, pp. 722–727.

[15] A. Moallemi, A. Burrello, D. Brunelli, and L. Benini, "Model-based vs. data-driven approaches for anomaly detection in structural health monitoring: A case study," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf.*, 2021, pp. 1–6.

[16] A. Marchioni, A. Enttsel, M. Mangia, R. Rovatti, and G. Setti, "Anomaly detection based on compressed data: An information theoretic characterization," *IEEE Trans. Syst. Man, Cybern.: Syst.*, vol. 54, no. 1, pp. 23–38, Jan. 2024.

[17] W. Wang, D. Lu, X. Zhou, B. Zhang, and J. Mu, "Statistical wavelet-based anomaly detection in Big Data with compressive sensing," *EURASIP J. Wireless Commun. Netw.*, vol. 2013, no. 1, pp. 1–6, 2013.

[18] M. Azimi and G. Pekcan, "Structural health monitoring using extremely compressed data through deep learning," *Comput.-Aided Civil Infrastructure Eng.*, vol. 35, no. 6, pp. 597–614, 2020.

[19] E. Ragusa, F. Zonzini, L. De Marchi, and P. Gastaldo, "Vibration monitoring in the compressed domain with energy-efficient sensor networks," *IEEE Sens. Lett.*, vol. 7, no. 8, Aug. 2023, Art. no. 6004604.

[20] C. Li, G. Wang, B. Wang, X. Liang, Z. Li, and X. Chang, "DS-Net : Dynamic weight slicing for efficient inference in CNNs and vision transformers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 4, pp. 4430–4446, Aug. 2022.

[21] L. Capogrosso, F. Cunico, D. S. Cheng, F. Fummi, and M. Cristani, "A machine learning-oriented survey on tiny machine learning," *IEEE Access*, vol. 12, pp. 23406–23426, 2024.

[22] H. Shi, H. You, Z. Wang, and Y. Lin, "NASA +: Neural architecture search and acceleration for multiplication-reduced hybrid networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 6, pp. 2523–2536, Jun. 2023.

[23] S. S. Saha, S. S. Sandha, and M. Srivastava, "Machine learning for microcontroller-class hardware—A review," *IEEE Sensors J.*, vol. 22, no. 22, pp. 21362–21390, Nov. 2022.

[24] H. Benmeziane, K. E. Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, "A comprehensive survey on hardware-aware neural architecture search," 2021, *arXiv:2101.09336*.

[25] M. Tan et al., "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2820–2828.

[26] Z. Guo et al., "Single path one-shot neural architecture search with uniform sampling," in *Proc. Euro. Conf. Comput. Vis.*, 2020, pp. 544–560.

[27] M. Zhang et al., "One-shot neural architecture search: Maximising diversity to overcome catastrophic forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 9, pp. 2921–2935, Sep. 2020.

[28] L. L. Zhang, Y. Yang, Y. Jiang, W. Zhu, and Y. Liu, "Fast hardware-aware neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 692–693.

[29] L. Lai, N. Suda, and V. Chandra, "Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus," 2018, *arXiv:1801.06601*.

[30] M. Li et al., "The deep learning compiler: A comprehensive survey," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 3, pp. 708–727, Mar. 2020.

[31] C. Li et al., "HW-NAS-Bench: Hardware-aware neural architecture search benchmark," 2021, *arXiv:2103.10584*.

[32] J. Lin et al., "MCUNet: Tiny deep learning on IOT devices," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 11711–11722, 2020.

[33] J. Lin, W.-M. Chen, H. Cai, C. Gan, and S. Han, "MCUNetv2: Memory-efficient patch-based inference for tiny deep learning," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2021.

[34] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, C. Gan, and S. Han, "On-device training under 256 kb memory," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2022.

[35] C. Banbury et al., "Micronets: Neural network architectures for deploying tinyml applications on commodity microcontrollers," *Proc. Mach. Learn. Syst.*, vol. 3, pp. 517–532, 2021.

[36] Z. Zhang, A. Mehmood, L. Shu, Z. Huo, Y. Zhang, and M. Mukherjee, "A survey on fault diagnosis in wireless sensor networks," *IEEE Access*, vol. 6, pp. 11349–11364, 2018.

[37] B. Peeters and G. De Roeck, "One-year monitoring of the z24-bridge: Environmental effects versus damage events," *Earthq. Eng. Struct. Dyn.*, vol. 30, no. 2, pp. 149–171, 2001.

[38] K. Maes and G. Lombaert, "Monitoring railway bridge kw51 before, during, and after retrofitting," *J. Bridge Eng.*, vol. 26, no. 3, 2021, Art. no. 04721001.

[39] F. Zonzini, A. Carbone, F. Romano, M. Zauli, and L. De Marchi, "Machine learning meets compressed sensing in vibration-based monitoring," *Sensors*, vol. 22, no. 6, 2022, Art. no. 2229.

[40] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning*, vol. 4, no. 4. Berlin, Germany: Springer, 2006.

[41] E. Favarelli and A. Giorgetti, "Machine learning for automatic processing of modal analysis in damage detection of bridges," *IEEE Trans. Instrum. Meas.*, vol. 70, Nov. 2020, Art. no. 2504013.

Edoardo Ragusa (Member, IEEE) received the master's (cum laude) degree in electronic engineering and the Ph.D. degree in electronic engineering from the University of Genoa, Genoa, Italy, in 2015 and 2018, respectively.

He is currently a Researcher with Department of Electrical, Electronic, Telecommunications Engineering and Naval Architecture, University of Genoa, where he teaches digital systems electronics and machine learning. He has coauthored more than 50 refereed papers in international journals and conferences. His research interests include machine learning in resource-constrained devices, convolutional neural networks, and deep learning applications.

Federica Zonzini (Member, IEEE) received the B.S. and M.S. degrees in electronic engineering and the Ph.D. degree in structural and environmental health monitoring and management from the University of Bologna, Bologna, Italy, in 2016, 2018, and 2022, respectively.

She is Junior Research Assistant in electronics with the University of Bologna. Her research interests include the design of intelligent sensor systems and edge computing in the context of structural health monitoring, encompassing advanced signal processing, and tiny machine learning.

Paolo Gastaldo received the Laurea degree in electronic engineering and the Ph.D. degree in space sciences and engineering from the University of Genoa, Genoa, Italy, in 1998 and 2004, respectively.

He is currently an Associate Professor with Department of Electrical, Electronic, Telecommunications Engineering and Naval Architecture, University of Genoa, where he teaches computer architectures and sensors. His main research interests include embedded machine learning, computational intelligence, embedded systems for advanced signal processing in robotics and prosthetics, and cybersecurity.

Luca De Marchi (Member, IEEE) received the M.Sc. and Ph.D. degrees in electronics engineering from the University of Bologna, Bologna, Italy, in 2002 and 2006, respectively.

He is currently an Associate Professor in electronics with the University of Bologna. He has authored more than 200 articles in international journals or the proceedings of international conferences. He holds two patents. His research interests include multiresolution and adaptive signal processing, with a particular emphasis on structural health monitoring applications.