Compression–Accuracy Co-Optimization Through Hardware-Aware Neural Architecture Search for Vibration Damage Detection

Edoardo Ragusa, *Member, IEEE*, Federica Zonzini, *Member, IEEE*, Luca De Marchi, *Senior Member, IEEE*, and Rodolfo Zunino

***Abstract*—Internet of Things (IoT) is a key enabler for the transition to the automatic structural health monitoring (ASHM) of technical facilities, thanks to the seamless flow of data from a multitude of always connected devices. Current IoT-ASHM installations, however, face the double challenge to ensure high accuracy while meeting the requirement of minimal energy consumption. This article tackles these issues from a deep-learning perspective and describes an IoT-enabled monitoring approach based on a distributed end-to-end deep neural network (DNN). The architecture supports both data compression and damage detection. A low-end microcontroller hosts a specific local DNN; a hardware-aware neural architecture search strategy rules network optimization, in order to satisfy the resource constraints set by low-end computing devices. The features extracted from data feed an aggregating unit, which includes a stacked global classification layer for full-scale damage detection. After proper quantization, the designed models are eventually deployed on a wireless accelerometer sensor. Finally, a cost-benefit analysis evaluates the system's impact on the sensor energy autonomy. Experiments on a well-known data set proved that the proposed solution could achieve state-of-the-art classification scores (all metrics above 98.4%) with a minimal transmission cost (less than 53 B on average); as compared with conventional approaches, the described strategy yielded a reduction of three orders of magnitude in energy consumption.**

***Index Terms*—Automatic structural health monitoring (ASHM), compression–accuracy co-optimization, energy profiling, hardware-aware neural architecture search (NAS), smart sensor systems.**

Manuscript received 21 March 2024; revised 20 June 2024; accepted 23 June 2024. Date of publication 26 June 2024; date of current version 25 September 2024. This work was supported in part by PNRR—M4C2—Inv. 1.3 through "FAIR" Project—Spoke 8 "Pervasive AI" funded by the EU through the NextGeneration EU Programme under Grant PE00000013, and in part by the "ALMArie CURIE 2022—Linea SUpER finanziata a valere sulle risorse del D.M. 737/2021 through the NextGeneration EU Programme under Grant CUP: J45F21002000001. *(Edoardo Ragusa and Federica Zonzini contributed equally to this work.) (Corresponding author: Federica Zonzini.)*

Edoardo Ragusa and Rodolfo Zunino are with the Department of Electrical, Electronic, Telecommunications Engineering and Naval Architecture, University of Genoa, 16126 Genoa, Italy (e-mail: edoardo.ragusa@unige.it; rodolfo.zunino@unige.it).

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

Digital Object Identifier 10.1109/JIOT.2024.3419251

I. Introduction

AUTOMATED inspection procedures, also known as automatic structural health monitoring (ASHM), allow to detect deterioration processes in structures with minimal or null human intervention. This ultimately results in cost-effective and timely repair actions. ASHM requires the hardware–software co-design of power-efficient resources for signal acquisition, conditioning, and digitalization. Operational features affect the associate infrastructure for data management, data analytics, and structural assessment. The Internet of Things (IoT) paradigm enables the vertical coalescence of these functionalities. In particular, it enables the realization of complex systems, featuring the permanent deployment of several sensors on the target assets and the periodic streaming of data to remote servers in a sensor-to-cloud continuum [1].

In the area of ASHM, vibration diagnostics entails the assessment of structures (ranging from civil infrastructures to industrial components) in dynamic regime, and largely benefited from those advancements in the last decade [2]. Two major issues hamper the full transition to IoT-enabled vibration ASHM, namely, the need for high accuracy and the necessity of minimal power consumption. The former aspect is critical because an inaccurate monitoring solution might bring about damage misdetection, and lead to unnecessary downtime and/or dangerous consequences. As to the energy-related issue, one should consider that, for easy deployment, IoT-ASHM systems typically rely on self-powered battery-operated wireless sensor networks. At the same time, wireless communications usually cover most of the power budget and hinder long-term sensor deployments. The energy-supply issue becomes critical when considering that many sections of large structures are not electrified or do not permit easy wiring, hence they cannot provide stable power sources. For this reason, suitable strategies that minimize data transfer across the network while preserving diagnostic accuracy are of paramount importance.

This article presents a decentralized approach based on a distributed end-to-end deep neural network (DNN), which integrates data compression and damage detection. The novelty of the proposed solution lies in using the compressed features themselves as the source of damage-sensitive information. This approach allows to bypass the reconstruction of structural

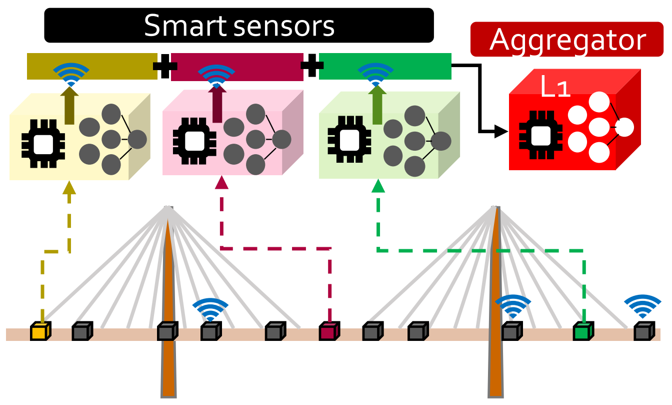


Fig. 1. Scheme of the proposed approach. The figure shows three sensing nodes placed in three different regions of a bridge. Each node features an MCU hosting a deep network that elaborates the raw data and extracts a compressed version of them. Processed and compressed features are transmitted to an aggregation unit that performs the final prediction about the health status of the structure.

parameters, and thus outperforms conventional methods for vibration monitoring, which are typically affected by high latency and energy consumption.

Fig. 1 outlines the overall scheme of the system in the case of a small example network with three sensors (represented by as many colors). The architecture features two levels. Resource-constrained, intelligent sensing nodes equipped with microcontroller units (MCUs) adhere to the structure; they both acquire data (from the local sensor) and process them onboard (by means of a local neural network), thus supporting data compression. In the subsequent level, the IoT backbone forwards the features extracted from each local node to an aggregation unit (in red). To complete the eventual health assessment, the aggregator module relies on a classifier network. The distributed scheme stems from the optimization of a full-scale DNN, yielded by a neural architecture search (NAS) strategy followed by an end-to-end training. The overall design process aims to maximize the accuracy in classification while limiting the overall transmission bandwidth. The network component hosted in the aggregation unit is trained by including an L1 penalty term. The resulting sparse solution can automatically pick a subset of the sensing nodes, by neglecting the elements that appear redundant or less informative. More importantly, the end-to-end training leads to a local representation within each network that implicitly takes into account the interactions between the various sensors.

The main contributions of this article can be summarized as follows.

1) The novel approach for vibration ASHM based on a distributed DNN can jointly support damage detection and data compression in a co-optimized fashion.
2) The nonlinear compression strategy reduces the transmission payload, while maintaining high accuracy in damage detection; the workflow attained state-of-the-art (SOTA) performances (classification metrics higher than 98%) on a well-known, real-world bridge data set.
3) The end-to-end training approach, based on L1 regularization, induces sparse solutions, thus allowing to disregard the less informative parts of the network (e.g., the less informative sensors); this proved a key feature to shrink the transmission payload to merely 53 B without affecting classification accuracy.
4) The optimal neural network architectures for simultaneous compression and damage detection derive from a hardware-aware (HW)-NAS strategy, which takes into account the constraints set by the inference-supporting hardware.
5) The overall workflow attained near SOTA accuracy (classification metrics above 97%) on a well-known real-world data set even when one single sensor is used, thus avoiding data transmission altogether.
6) The designed tiny architectures were deployed, validated, and benchmarked on a low-end computing platform; different quantization strategies massively reduced both execution time and memory requirements (up to 3× lighter) as compared with desktop implementations.
7) A full-scale energy profiling of the quantized inference models (on a real IoT wireless sensor node) show that the NN-supported compression scheme can reduce the energy consumption per hour up to three orders of magnitude, as compared with compression-free monitoring solutions.
8) The analysis of noise effects indicated a negligible reduction of the network performances for signal-to-noise ratio (SNR) levels as low as 20 dB.

This article is organized into five technical sections. Section II presents a thorough review of SOTA solutions for vibration data compression and structural damage detection in the compressed domain; the analysis covers both deep learning (DL) and conventional statistical methods. The detailed description of the novel approach, including the design of individual DNNs, the end-to-end training, and L1 regularization, is presented in Section III. Section IV reviews the adopted real-world data set, while Section V reports on the experimental results. The analysis in Section VI deals with model quantization and the deployment on a resource-constrained MCU, hosted by a low-end wireless accelerometer. The treatment also covers a cost-benefit analysis in terms of energy consumption and the evaluation of noisy data. The final section of the manuscript makes some concluding remarks and envisions future perspectives.

II. Related Works

Damage detection from compressed data is attaining increasing interest in the DL community, as it can open unprecedented opportunities for increasing system efficiency, from both an electrical and a communication point of view. DL-driven approaches can somehow relax the tradeoff between compression and detection, thanks to their remarkable learning capabilities. The definition of the most appropriate architecture for both the compressing and the detecting component is crucial. In principle, one might consider a plethora of combinations, depending on the characteristics of the processed signals and the subtleties inherent in the anomaly patterns. The

TABLE I
ANALYSIS OF SOTA APPROACHES FOR VIBRATION DIAGNOSTICS AND REDUCTION IN THE COMPRESSED DOMAIN (ACCURACY MEASURED IN %)

Ref.	**Compression method**	**Inference model**	**Application**	**Performance**		**Edge**
				Acc	CR	
[4]	PCA, AE	PCA Fully-connected AE Convolutional AE	Viaduct	98.8 70.0 56.3	16	✓
[5]	AE	CNN+AE	Bridge	99.2	10	✗
[6]	Deep convolutional AE	Reconstruction loss	Motor bearing (CWRU)	55.0	32	✗
[8]	CS	Stacked sparse AE	Motor bearing (CWRU)	85.0	3	✗
[9]	CS	Improved Multiscale Network	Motor bearing (CWRU)	95.2	4	✓
[11]	CS + Statistical features	RF	Bridge (Z24)	94.0	256	✓
[12]	CS	1D CNN	Bridge (Z24)	96.7	8	✓
[10]	CS	AE, One class classifier	Bridge (Z24)	96.0	6	✓
[13]	Canonical correlation analysis	Hierarchical MLP	Motor bearing	95.0	600	✗
[14]	Histogram of time series data	CNN	Frame (IASC-ASCE)	93.0	10	✗
[15]	Autoregressive modeling	CNN	Bridge (Z24)	91.0	121	✓

wide taxonomy of available architectures includes, among the many others, convolutional neural networks (CNNs), standard multilayer perceptron (MLP), and autoencoders (AEs).

An instance of an AE can learn an efficient representation of a data set, such that the predicted output matches the supplied input. AEs achieve this task in a completely unsupervised manner; one can exploit this property for both compression and diagnostics purposes, by observing how the residual error between inputs and reconstructed outputs changes over time. The applicability of that approach to damage detection in vibration-based SHM has been shown in some representative research works, such as [3] and [4], in which the authors monitored the health condition of a highway bridge viaduct in Italy. In the former paper, an AE supported data reduction in conjunction with an MLP for flaw detection, and attained a good level of performance. The latter work featured a fully connected AE; its convolutional variant is explored as both encoder and decoder, yielding a maximum accuracy of 98% and a compression ratio (CR) of 16×. The AE-based framework was also investigated by Ni et al. [5] for the monitoring of a long-span bridge in China; the approach attained high accuracy (99%) with a considerable CR (nearly 10×). In the application of the same strategy to industrial settings [6], a physics-informed deep convolutional AE could identify cracks in the bearing of an induction motor; the solution achieved notable compression gain but lower diagnostic capabilities (below 55%).

Hybrid architectures typically rely on a mixture of statistical and DL blocks. The overall approach typically implies edge-oriented compression methods; compressed sensing (CS) [7] allows a straightforward hardware implementation involving multiply-and-accumulate operations. Most works in literature adopt CS for encoding, matched with various anomaly detection strategies. For example, in [8] and [9], CS supported the condition monitoring of a rotating motor; it operated in combination with either a stacked sparse AE or an improved multiscale network, i.e., a model with multiple branches for extracting and merging parallel characteristics at different levels. These experiments proved effective only at a limited compression depth (less than one fourth), and scored a defect-prediction capability above 90%. The comparative analysis in [10] for bridge monitoring took into account both CS and a one-class classifier network. Significant improvements (data compression 256×, accuracy 94%) were attained in the inference step when applying a random-forest classifier on a reduced pool of statistical features extracted from CS-processed data [11]. Recently, integrating CS with a DL framework has led to an increase in accuracy with respect to existing approaches [12].

Alternative approaches yielded interesting scores on vibration data sets, including purely data-driven compression methods, such as principal component analysis (PCA) [4], canonical correlation analysis [13], histograms of data distributions [14], and autoregressive parameters [15]. Table I provides a summary of the performances of these works, by specifying the compression method, the diagnostic model, the performances (i.e., percentage accuracy and achieved CR), and the suitability for edge deployment.

Independently of the implementation principle and the obtained results, some issues limit the full-scale deployment of the above solutions. First, computing requirements often exceed the available memory and algorithm resources, thus preventing the methods' deployment on low-end devices; in other words, the deployment of artificial intelligence models on tightly constrained scenarios calls for ad-hoc solutions covering specialized algorithms and dedicated hardware [16], [17], [18]. Second, most approaches cannot provide both a local (sensor-related) and a global (aggregated) structural insight, which could actually prove very useful not only for detection but also for localization purposes. Finally, those methods do not support an integrated optimization of compression and damage detection, thus obtaining suboptimal neural architecture representations.

HW-NAS [19] might provide a viable solution to those issues by the design of deep networks that comply with hardware limitations [20], although its practical adoption is not straightforward. For example, the correct quantification of hardware requirements is a function of the software layer, thus making a-priori accurate quantification difficult [21], [22], [23]. The search time required by the automatic procedure might be another shortcoming: the optimum-search procedure might require thousands of hours

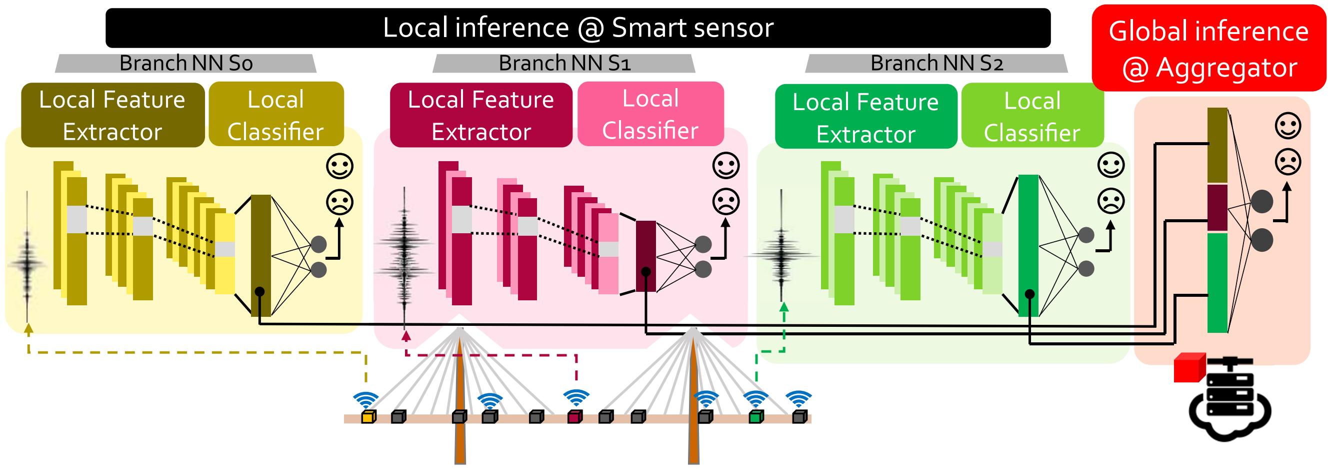


Fig. 2. Architectural description of the proposed local-to-global monitoring approach.

of GPU computing. Accelerated approaches, such as super networks [24], [25], aim to mitigate that problem.

The application of HW-NAS yet proved effective in handling a very large set of computing constraints [26], leading to SOTA methodologies for specific target hardware devices [27], [28]. HW-NAS represents a promising research direction but requires a careful evaluation of all the peculiarities of the specific application domain.

III. Local-to-Global Monitoring With End-to-End Training and Regularization

End-to-end deep networks can process raw data, optimize intermediate representations, and extract the relevant features of interest automatically. DNNs are usually represented in terms of computational graphs. The deployment requires to allocate graph nodes to the respective computing resources, while training completes independently of the target infrastructure. In the application context of this research, the problem requires to maximize the overall accuracy while minimizing wireless traffic.

Fig. 2 outlines the complete structure of the end-to-end neural network, and further details the representation anticipated in Fig. 1. Each peripheral sensor performs both compression and detection by hosting a DNN that integrates two main blocks: 1) a local feature extractor, corresponding to the core computing block, holds the set of layers to process and compress raw data and 2) a subsequent classification stage works out the local prediction of the health status as a result of the sensor information itself. As opposed to standalone implementations, the output of each local-level representation is duplicated in a subsequent, logically equivalent path, that is outsourced to an external aggregation network and used in a further step for full-scale diagnostics.

The proposed approach takes advantage of a defining feature of DNNs, namely, the ability to aggregate and encode input information to the output classification layer, while implementing feature extraction and data compression via nonlinear DL operators concurrently. Classification networks aim to minimize accuracy while optimizing the intermediate representations for the purpose at hand; this implies that the training process tends to delete any information that is irrelevant to the classification task. As a result, the uppermost feature maps (that are closer to the classification layers) most likely tend to include the useful information for damage detection.

This behavior can be induced by imposing specific architectural constraints; for instance, by letting tensors of progressively fewer elements propagate through the network. In the method described here, a feature set, stemming from a global average pooling layer, forces compression at the output of each local feature extractor. This allows to perform a notable compression of the input tensor by dropping any residual geometrical information.

The local-to-global aggregating structure in the proposed end-to-end scheme has points in common (and is actually compatible) with federated learning (FL), with some distinctive features. FL splits the computational cost of *training* among several devices that cooperate in tuning the network's parameters, while the aggregator assembles their contributions into one inference model. Conversely, the method presented in this article follows a train-then-deploy approach, in which sensors carry out the actual *compression* process, whereas the aggregating unit merges information from the peripheral ones to predict the full-scale health status.

A. NN Design

This section deals with the design of the single-branch NNs, that is, the sensor architectures supporting feature extraction and pattern classification at the local level. This is a preliminary, crucial step to the overall optimization problem and the associate training procedure.

1) Branch Architecture: An HW-NAS optimizes the architecture of each branch NN. The algorithm drives the architecture search by limiting both the number of parameters (i.e., model size) and the dimensions of the propagated tensors, such that they can satisfy the resource constraints of the supporting computational units. Toward that purpose, HW-NAS solves an optimization problem that pursues two goals: 1) to define the architecture that best fits the health-status prediction goal, depending on the data provided by the sensor under analysis and 2) to reduce the volume of data by forcing small-size representations. HW-NAS optimizes a general model, featuring a single branch structure made of an initial convolutional block ("Local Feature Extractor"), followed by a classification block ("Local Classifier"). The latter module consists of a global average pooling layer and an output classification layer, with two neurons and softmax activation. The output vector prompted by each branch NN can vary in shape, depending on the amount of geometrical features that survive after pooling.

Equation (1) formalizes the target optimization problem. The validation error $\mathcal{L}_{val}(w^*(a), a)$ scored by architecture a with the set of trained weights w^* drives the optimization process. The associate architecture search is subject to two computational constraints: first, a limitation on the number of parameters $|a|$ in the architecture a; second, an upper bound to the number of elements $|T|$ in the largest tensor that propagates through the architecture. The first constraint ultimately sets a requirement on flash memory, whereas the second one relates to the used RAM. A pair of threshold values, F_{Th} and R_{Th}, rules this mechanism and represents the largest numbers of elements that can be stored in flash memory and RAM, respectively. The search space contains networks composed of standard cells, which match those adopted in [12] and rely on convolutional operations

$$\begin{aligned} \underset{a\in\mathcal{A}}{\text{minimize}} \quad & \mathcal{L}_{val}\big(w^*(a), a\big) \\ \text{subject to} \quad & w^*(a) = \arg\min_{w} \mathcal{L}_{\text{train}}(w, a) \\ & |a| < F_{\text{Th}} \\ & |T| < R_{\text{Th}}. \end{aligned} \tag{1}$$

2) Full Architecture Optimization: The global monitor network that supports the aggregator module relies on a straightforward, fully connected classification layer holding one neuron per class, and needs to be optimized, as well. The choice of this architecture has a twofold rationale. First, the features feeding the aggregator are connected to the classification layer in the local feature extractor blocks; thus, the aggregating network actually weights the contributions from different nodes. In addition, by connecting the branch networks to the global classification layer one can deploy an effective sparsification strategy and implicitly prune noninformative sensing nodes. Toward that end, using L1 regularization by penalty fosters sparse solutions that may cancel the contributions of specific features: each neuron in the aggregator's classification layer only connects (by one weight) to one of the sensor-generated residual features. If the weights of all the classification neurons connected to a given feature nullify, that feature does not need be transmitted, thereby reducing transmission costs remarkably.

The classical penalty term as per (2) rules L1 regularization

$$\mathcal{R} = |W_{\text{aggr}}| \tag{2}$$

where $|W_{\text{aggr}}|$ is the ℓ_1 norm of the weights of the aggregator layer.

The entire DNN (branch NNs and global aggregator) is then trained in an end-to-end fashion, jointly adjusting all the branches. The major advantage of retraining the whole architecture is that the weights of each branch NN are optimized by considering the information from all the sensing nodes. This leads to an optimized representation that accounts for the information extracted by the entire pool of sensors.

In this setup, finding a suitable expression for the network loss function is of paramount importance. In principle, one could train the network so that it only matches the output prediction prompted by the global inference model after aggregation. In contrast, taking into account the classification outcomes at the sensor level offers two additional benefits: it allows to get a diagnostic status directly at the sensor node level, and it can be used to adjust a local representation to maximize the classification performance of each sensing node. A straightforward loss function weights all the outputs of the aggregator ($\mathcal{L}_{\text{Aggr}}$) and local network ($\mathcal{L}_{\text{node}_i}$) as formalized in

$$\mathcal{L} = \mathcal{L}_{\text{Aggr}} + \sum_{i}^{N_s} \lambda_i \mathcal{L}_{\text{node}_i} + \lambda_R \mathcal{R} \tag{3}$$

where the term λ_R weights the relative contribution of the regularization term, whereas the quantity λ_i weights the loss associated with the ith branch NN ($i \in \{1, \ldots, N_s\}$, N_s being the number of sensors).

IV. Experiments

A. Data Set Description: Z24 Bridge

The Z24 bridge[1] is considered as one of the most representative data set for the validation of vibration-based diagnostic approaches. It is related to a highway viaduct connecting Bern and Zürich that has been subject to a one-year monitoring campaign (September 1998–August 1999). During experiments, accelerations were collected from eight force-balance-type FBA-11 accelerometer sensors by Kinemetrics (sampling frequency of 100 Hz, acquisition windows of 32 768 samples every hour) under different operational and environmental conditions. Before its demolition, progressive damages were provoked in a controlled manner to replicate medium-to-severe deterioration processes. The data set contains a total amount of 5651 measurements, the first 4922 tests related to normal conditions, the remaining 729 coming from defective configurations. Each time series has been split into windows of 512 elements to balance between memory and computational requirements of low-end devices; thus, a single sensor instance is reshaped as a matrix of 512×64 elements.

B. Training and Evaluation Setup

The code design to verify the effectiveness of the proposed approach has been implemented in Python using Keras and Tensorflow libraries. All the tested architectures shared the same training setup: a maximum of 100 epochs with an initial learning rate of 10^{-3}, learning rate reduction on the plateau, and early stopping using the validation loss as the metric. In particular, the categorical cross entropy has been selected as loss function for all the single networks contributing to (3).

[1]https://bwk.kuleuven.be/bwm/z24

All the training procedures have been repeated five times using a multistart approach selecting the best one based on the validation set. A standard fivefold cross-validation procedure has been imposed. The validation set, for each fold, has been extracted from the training set selecting a random subset of 20% of the training data. The classification performance has been measured for all the models on the testing folds that have never been involved in any parameter or hyperparameter tuning.

The NAS procedure considered the computing resources of a standard MCU: a maximum flash size of 512 kB and a RAM size of 256 kB have been imposed, respectively. These threshold values are sufficiently large to entail the majority of low-power MCUs available in the market. Increasing such quantities is not of relevant interest since it would entail the family of mainstream or high-performance processors not suited for extreme-edge deployment. Noteworthy, while the size of the model weights impinge on the flash size, the RAM is typically set by external constraints, first among all the dimension of the time series to be acquired. The number of generations has been set to 500. Random mutation function considered the insertion, deletion, or modification of one block of the network.

Accuracy, Precision, Recall, and F1 have been selected as metrics. These scores are 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, accuracy alone proves unreliable in the presence of unbalanced data sets [29].

C. Neural Network Definition

me analyses on network selection for the Z24 data set have been carried out in the literature. Previous works pointed out that 1-D convolution in the time domain is a preferable solution when compared with other convolution-based approaches [12]. In the same work, a profitable search space was presented leading to SOTA generalization performance while maintaining modest compute requirements. Coherently, the present analysis uses instances of the same search space.

V. Results Outline

The following objectives have been pursued within the experimental validation:

1) investigate the role of end-to-end training proving that the newly proposed approach leads to SOTA performance on the Z24 data set;
2) analyze the role of the branch architecture for compression and classification at a sensor level;
3) demonstrate the superiority of the proposed approach in terms of transmission payload;
4) deploy, profile, and benchmark the designed models on a low-end MCU by making use of different quantization techniques;
5) evaluate the impact of the devised processing flow on the energy consumption of a wireless accelerometer sensor for IoT-driven vibration monitoring;
6) analyze how the performances can scale when practical issues due to operative and intrinsic noise sources are entailed.

TABLE II
Comparative Study of the Performance of the Proposed End-to-End Architecture in Terms of Classification Scores and Transmission Requirements, Compared With SOTA Alternatives Working With the Same Data Set. Better Results for Each Metric Are Magnified by Bold Fonts

Ref	Architecture	Acc [%]	Prec [%]	Rec [%]	F1 [%]	Out Size [B]
[12]	CNN*-Z24	96.7	97.6	**98.6**	98.1	16,384
[11]	RF	94.4	96.4	97.1	96.8	412
[30]	CNN1	95.7	97.4	97.7	97.5	16,384
[30]	CNN2	96.7	98.3	97.9	98.1	16,384
[30]	OCCNN	93.0	94.0	95.0	91.0	21,845
[10]	ANN	95.0	99.0	94.0	97.0	21,845
[31]	GMM	95.0	98.0	93.0	95.0	131,072
[31]	PCA	62.0	95.0	29.0	45.0	131,072
[31]	KPCA	95.0	99.0	92.0	95.0	131,072
This work	CNN^*_{EE}	**97.1**	**98.1**	98.5	**98.3**	**372**

A. Effect of End-to-End Training

In the first experiment, the importance of end-to-end training has been evaluated by using, for all the branch NNs, a preselected architecture with two 1-D convolutional layers (Conv 1D) followed by average pooling with window size 2 between the two convolutions. The kernel size was set to 3 for both layers, while the number of filters was fixed to 5 and 3 for the first and second layer, respectively. The output of the second convolution is directly connected to the aggregation network composed of a concatenation layer and two additional convolutional layers with 30 and 10 filters, respectively. This architecture has been selected following design choices presented in previous works [12].

Table II compares the performance of the end-to-end architecture (referred to as CNN^*_{EE}, last row) with nine existing solutions trained and tested on the Z24 bridge data set using the same evaluation metrics. Experimental results clarify the beneficial effects of the end-to-end training when applied to raw data. The proposed model obtained the best accuracy, precision, and F1 among all the investigated solutions. Only the model proposed in [12] slightly improved in terms of Recall, but was less accurate in the other indicators. Importantly, the newly proposed CNN^*_{EE} architecture can attain such performances with a drastic reduction in the transmission payload requiring only 372 B[2] to be communicated, a quantity which is 44× lower than that of the competing CNN*-Z24 model. These outcomes highlight the capability of DNNs to extract nonlinear relationships that cannot be handled by linear compression matrices as it is imposed in standard CS.

B. Role of the Branch NN

1) Search Space Analysis: To prove the validity of the adopted convolution-based network, additional experiments

[2]It is assumed that each feature is represented as a `float` data quantity whose storage requires 4 B.

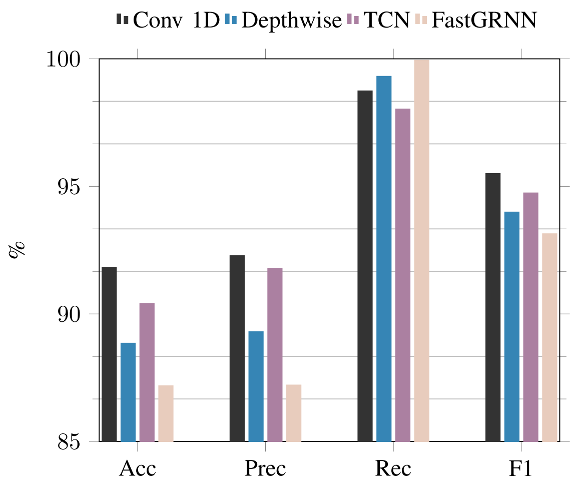


Fig. 3. Average classification scores comparing different architectures for the design of the branch NN.

have been performed in which tiny architectures specialized for edge inference have been explored for the realization of the branch NN. The list of competitors includes Depthwise convolution [32] and temporal convolution network (TCN) [33], which are two variants of standard convolution, as well as FastGRNN [34]. For the first two models, analyses have been conducted by replacing the Conv 1D layers in the CNN_{EE} with either the Depthwise or TCN block. More specifically, the TCN has been designed considering blocks with groups of dilatated layers having a dilatation value of 1, 2, 4, 8, 16, and 32, respectively. In the case of Depthwise, depth_multiplier has been set to the same number of filters of standard convolution. Regarding FastGRNN, the comparison has been performed using a similar rationale. We have included two recurrent layers with 5 and 3 neurons, respectively. In practice, a neuron for each filter used in the convolutional architectures.

Average classification scores among all the eight sensors are depicted in the bar chart of Fig. 3. As can be seen, Conv 1D is superior in all the metrics, apart for the isolated case of Recall which is, however, due to the strong bias affecting both Depthwise and FastGRNN. This is proven by the fact that all other scores are remarkably lower for such networks.

2) HW-NAS: The second experiment aims to clarify the role of HW-NAS in extracting information from the single sensors. The main findings are summarized in Fig. 4, which is organized in eight subplots, one per sensor, reporting on the performances of three different neural architectures.

1) *CNN_{NAS} (Gray Bars):* NN architecture returned by the HW-NAS for the corresponding sensor node.
2) *CNN_{SA} (White Bars):* NN model explored in Section V-A with training performed in a standalone manner, i.e., the architecture of each branch NN is trained independently.
3) *CNN_{EE} (Black Bars):* Sensor-related classification scores returned by the suboptimal model in Section V-A trained in an end-to-end fashion.

A point-wise comparison with respect to the classification scores obtained by the global prediction network CNN^*_{EE} (star markers) explored before is also included, to elucidate about the effects of the different strategies.

Results confirm the major role played by the network architecture. In all cases, in fact, the model obtained from the HW-NAS procedure performs better for all the considered metrics (apart for a few exceptions). Interestingly, the three networks converge to the same solution for S3, indicating that data from this sensor are subject to a strong bias, as also shown by the poor accuracy and precision. The importance of optimized design strategies is proven by the fact that, for the majority of the branch NNs (see Table III), the algorithm selected a solution with many levels, confirming the necessity of nontrivial architectural configurations which cannot be fine-tuned empirically. The same Table reports on the depth (header "Level"), number of parameters ("Params"), amount of algorithmic operations ("FLOPS"), and dimension of the tensor returned by the global average pooling ("OutSensorSize") for all the eight branch architectures. The same characteristics are also specified for the suboptimal model introduced in Section VI (last row), which indeed exhibits a simpler architecture when compared to the automatically generated ones. Eventually, one can observe a large variance in the retrieved network architecture highlighting that each sensor introduces a peculiar behavior.

When comparing the results of the two DNN architectures (end-to-end versus standalone) it is evident that the co-optimized version trained in the end-to-end manner obtains slightly lower generalization performance. This outcome confirms that end-to-end training can bias the extracted representations reducing the accuracy on the node, while preserving the global accuracy. This is proved by the fact that star markers, i.e., those retrieved at the output of the aggregation unit with end-to-end training, are almost always superior to the scores retained for the individual sensing units.

A very important outcome is that the HW-NAS architecture trained and tested on S1 overcomes alone the best results in the literature. In fact, comparing the four performance metrics in Fig. 4 for S1 with CNN^*_{EE} (stars versus gray bars), one can see that, except for a small decrease in Recall, the other tree metrics are better than the previous benchmark. This result becomes even more important because these solutions can be hosted directly on the sensing node without transmission.

Nevertheless, it should be noted that defining the policy for the selection of this sensor is far away from being trivial. NAS solves a nonconvex problem that can lead to very disparate solutions based on different initial conditions, that can hinder the search problem. Accordingly, one can argue that the CNN_{NAS1} architecture is simply a better architecture in general. This observation could be further strengthened by the fact that vibrations are, in principle, global proprieties of a structure. Further experiments were, then, performed to investigate these aspects.

In Fig. 4, the performance of CNN_{NAS1} trained and tested on data from other sensors is also depicted with red rounded makers. In all the cases, except for S4, the results of architecture CNN_{NAS1} are suboptimal with respect to the ones scored by NAS individually on each sensor data set, confirming that different architectures are necessary to best fit different

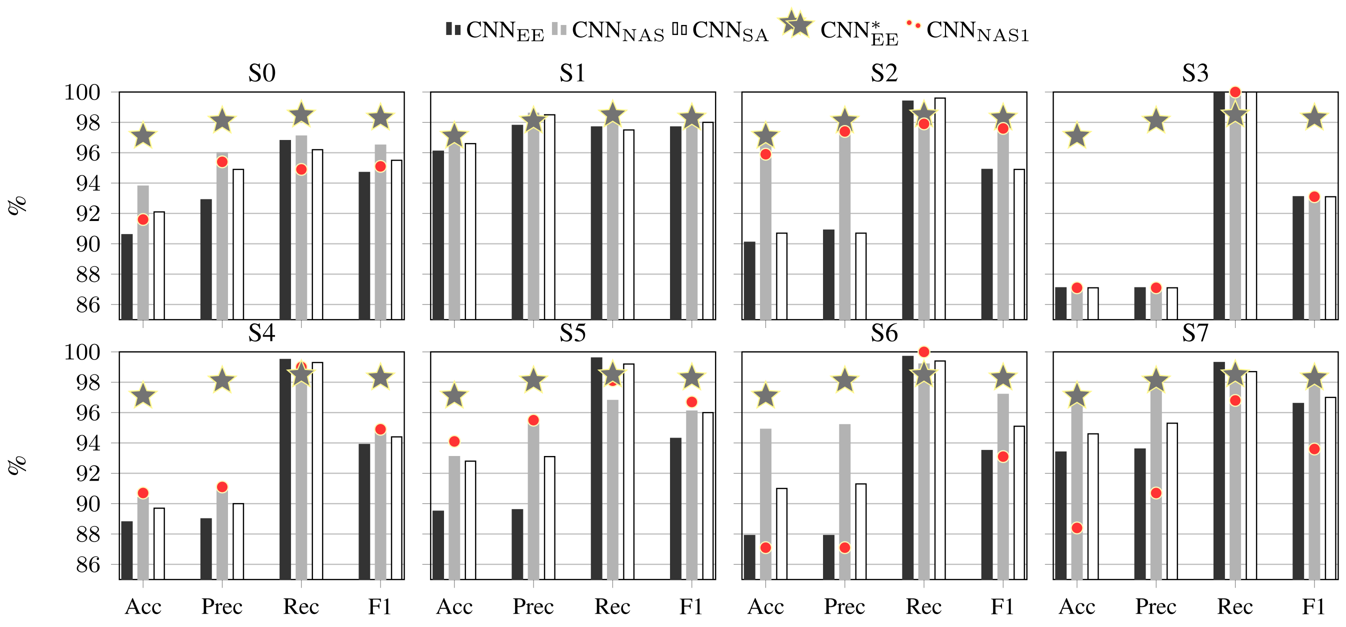


Fig. 4. Classification performances at the sensor level when comparing SA and EE monitoring approaches.

TABLE III
COMPUTING PERFORMANCE AND ARCHITECTURE DETAILS FOR THE BRANCH ARCHITECTURES RETRIEVED BY HW-NAS FOR EACH SENSOR

Model	Levels	OutSensorSize	Params	FLOPS
CNN$_{NAS0}$	8	3×10	34,968	3,822,288
CNN$_{NAS1}$	8	4×11	33,230	2,738,912
CNN$_{NAS2}$	8	1×1	28,937	3,063,962
CNN$_{NAS3}$	2	30×3	12,861	1,539,504
CNN$_{NAS4}$	7	1×29	48,738	5,105,813
CNN$_{NAS5}$	6	7×19	26,857	2,670,847
CNN$_{NAS6}$	3	14×28	48,281	5,562,904
CNN$_{NAS7}$	8	3×12	50,836	5,226,918
CNN$_{EE}$	2	31×3	7,741	955,940

sensing locations. This outcome can be justified by the fact that, even if natural modes are global parameters for the structure, the sensitivity of the sensors in terms of amplitude and richness of the captured vibration response is strongly dependent on its position; for example, some sensors could be less informative since they are more proximal to static mechanical components in correspondence of which vibration is minimal. In the case of S4, architecture S1 performs slightly better on the test set. This minor advantage could be due to the nonconvex nature of the optimization problem approached by the NAS. Anyway, the difference is very small, confirming that the architecture retrieved by the automatic procedure is still a good solution. In addition, the test involving network CNN$_{NAS1}$ confirms that the performance gap between CNN$_{NAS}$, CNN$_{EE}$, and CNN$_{SA}$ is not due to the number of parameters, but strongly conditioned by the network topology.

C. Hyperparameters Analysis

Once the optimal branches are obtained through HW-NAS, the proper selection of the regularization parameter and the relative weights λ_i associated with different terms of the loss function in (2) deserves primary importance.

Four weighting strategies have been investigated.

1) *"Unitary Weights" (1_W):* $\lambda_i = 1$ is assumed for all the branch NNs, such that the weight associated with each local network equates the weight associated with the distributed network.
2) *"Uniform Weights" (Un_W):* $\lambda_i = 1/N_s$ is assigned to each branch term; consequently, the total contribution of all the subnetworks' losses equates the contribution of the distributed network.
3) *"Accuracy Weight" (Acc_W):* The validation error of each network is considered independently by setting $\lambda_i = \text{acc}_i$, i.e., the value of the validation accuracy is set as the weight for the ith branch. Indeed, since all λ_i will have a value between 0 and 1, in this new setup, the loss of the network with better accuracy has larger weights. Thus, in the extreme case of a branch network with an accuracy of 1, the associated term will weigh 1, corresponding to the value of the loss of the distributed network.
4) *"Uniform Accuracy" (Un_Acc):* Combining schemata (b) and (c), $\lambda_i = \text{acc}_i / \sum_i \text{acc}_i$ is assumed such that the score for each branch is given proportional to the validation accuracy while ensuring that the summation of all weights is unitary.

Moreover, to investigate the role of the regularization term, each of the above described schemes has been tested with four values of λ_R, namely, 0.001, 0.01, 0.1, and 1, ranging from small (0.001) to strong (1) values of regularization.

The results are depicted in the radar charts of Fig. 5 and confirm that the regularization parameter plays a major role. Indeed, when λ_R is in the range (0.001 or 0.01), all the weighting schemes lead to new SOTA results for all the

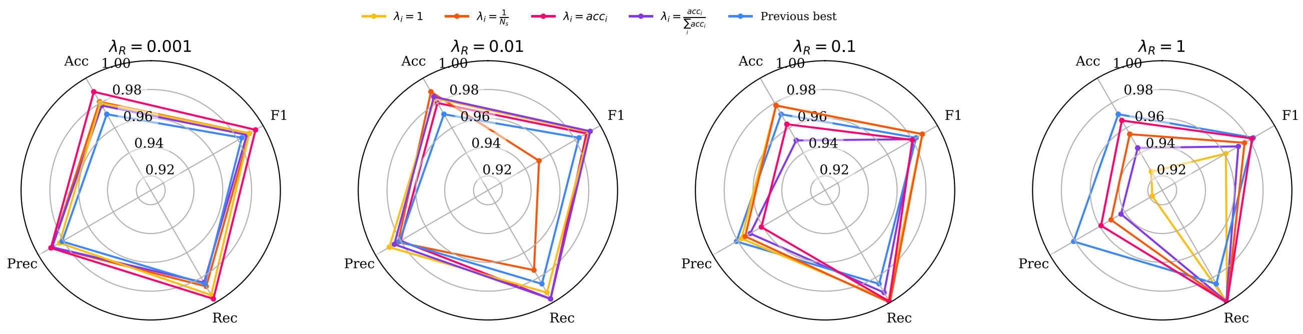


Fig. 5. Radar plots for the four considered regularization mechanisms and weighting strategies.

TABLE IV
PERFORMANCE OF THE L1-REGULARIZED ARCHITECTURE COMPARED TO THE BEST RESULTS (THOSE IN BOLD) IN TABLE II AND OTHER REGULARIZATION ALTERNATIVES

Architecture	Acc [%]	Prec [%]	Rec [%]	F1 [%]	Out Size [B]
Previous best	97.1	98.1	98.6	98.3	372
1_W-L1-CNN$^*_{\text{NAS,EE}}$ (0.01)	**98.4**	**98.9**	99.2	**99.1**	52.8
1_W-L1-CNN$^*_{\text{NAS,EE}}$ (0.001)	98.0	98.3	**99.4**	98.9	452
1_W-L2-CNN$^*_{\text{NAS,EE}}$ (0.01)	98.0	98.6	99.1	98.9	452
Fully	98.1	98.6	99.2	98.9	452

considered metrics (apart from accuracy and F1 of strategy Un_W) if compared with the previous best solution. This confirms both the suitability of the proposed approach and the robustness against variation of the hyperparameters. When the regularization value, instead, becomes large (0.1 or 1), the sparsity term becomes prevalent leading to a significant deterioration of the overall performance. Eventually, the best solution is the one among the sparse networks with the smallest values of λ_R, indicating that this value balances the contribution of the regularizer and the contribution of the empirical error. Comparing the four weighting schemes per given λ_R, plots show that it is not possible to identify a best solution winning in all cases.

For the sake of conciseness, in the following, only results for the unitary weighting scheme are presented, which has deliberately been selected based on two observations. First, from a transmission point of view, the case with $\lambda_R = 0.001$ is not advantageous since it imposes 100% of the features to be outsourced. The second reason is that, instead, the case of $\lambda_R = 0.01$ is almost 8× sparser (see Section V-D), with the 1_W and Un_Acc being the best weighting schemes. Among these, the unitary weighting scheme has been selected because of its simplicity of implementation.

Two competitive aggregating techniques have been considered for the sake of thorough comparison. The first is a model identical to CNN$^*_{\text{NAS,EE}}$($\lambda_R = 0.01$, 1_W weighting) but with L2 regularization, while the latter (Fully) is a solution in which the classification layer is substituted by a fully connected neural network with one single hidden layer of 50 neurons. Table IV highlights that the HW-NAS plus L1 regularization overcomes the previous solutions for all the metrics, proving the effectiveness of the greedy selection of the architecture for the single branches, the effectiveness of the architecture constraints introduced, and the effectiveness of the proposed loss function. Eventually, the comparison between the two values of λ_R remarks that the smallest value of regularization led to nonsparse solutions, making the value of 0.01 a preferable choice.

TABLE V
AMOUNT OF TRANSMITTED FEATURES OVER DIFFERENT FOLDS

Fold	**S0**	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**Total**
fold 0	0	3	0	0	3	0	0	3	9
fold 1	0	5	0	0	11	1	0	5	22
fold 2	0	3	0	0	4	0	0	3	10
fold 3	0	3	0	0	5	2	0	3	13
fold 4	0	2	1	0	3	3	0	3	12

D. Transmission Cost Analysis

A thorough analysis of the transmission cost implied by the designed architectures is mandatory to judge their benefit in the context of IoT-oriented deployments. For this reason, the number of payload digits (measured as data B) to be transmitted to run a single global inference has been included in the last column of Table II.

Interestingly, the proposed approach is unique in that it transmits a different amount of features for individual sensing nodes, based on the optimal representation retrieved from HW-NAS. Performing compression using an end-to-end approach drastically diminishes the transmission payload when compared with purely CS methods, with a payload below 53 B (only 13.2 features on average), which is 24.8× more efficient than the CNN$^*_{\text{EE}}$ identified before. For the sake of clarity, the indicated output quantities are not integer because of the five-folded training procedure which implies a different set of features for each fold; coherently, the presented value is the average computed over the total folds. Eventually, this is more efficient than the L2-regularized and the fully connected alternative, having a payload of 452 B (113 features).

An in-detail analysis of the contribution of each sensing node when $\lambda = 0.01$ is shown in Table V. The last column reports the sum over all the different folds. Each element of the table reports the number of active features, computed as the sum of the absolute values of the weights (one for each class) connected to the specific feature. The first result is that, in none of the folders, sensors S0, S3, and S6 have a role in the final prediction. Similarly, S2 participates only in fold 4. The selected sensors are not merely the ones that score the best accuracy alone (see Fig. 4). For example, sensor S4 that takes a large role on the global prediction scores modest results alone, confirming the capability of the network to mix, during the end-to-end process, information from the different sensing nodes.

VI. Prototyping on Wireless Accelerometer Sensor: Deployment and Cost-Benefit Analysis

The deployment of ASHM systems in real settings is influenced by hardware and software constraints. On one side, the sensing technology has to be appropriately selected to support the realization of affordable, low-power, and small-size sensing platforms. Micro-electromechanical systems (MEMSs) sensors are particularly suited to meet these requirements thanks to their high level of integration, reduced price, and market availability [35]. Concerning the computing architecture, key performance features, such as clock speed, power management, memory space, and instruction per cycles, have to be matched with the computational complexity of the sought inference models. Thus, dedicated quantization techniques that can reduce model latency and data depth with limited degradation on accuracy must be applied, while enhancing the overall system efficiency. Finally, the transmission module and communication protocol deserve primary attention for two main reasons: first, they represent the most energy-hungry entry in the power budget; second, they should allow for sufficient and reliable coverage, preventing data loss while ensuring constant point-to-point link.

A. Model Quantization and Deployment

Four different model quantization approaches have been investigated.

1) *Float (`FullFloat`):* All network parameters and input/output tensors are represented as 32-bit `float` quantities, adopting the deepest depth available for MCU devices.
2) *Full Integer (`FullInt8`):* All the mathematical operations are implemented via full integer quantized operators, forcing the weights and activations to have 8-bit precision, including input and output data tensors.
3) *Float Fallback (`FloatFall`):* Analogous to `FullInt8`, but it creates a hybrid solution that replaces some of the integer operators with a `float` equivalent function; `float` precision is used to represent input/output data as well.
4) *Quantization-Aware (`Qaware`):* Different from the post-training approaches above, in which quantization is applied to the already trained model, this technique emulates the possible loss due to 8-bit precision at training time using precision reduction in weights and activation representation. Its pretraining nature can prevent performance degradation after deployment at the cost of a higher convergence time. By virtue of this, it is expected to increase the classification performance if compared with post-training quantization.

The STM32L496 MCU [36] has been used for prototyping, since it is an ultralow-power microcontroller based on the high-performance Arm Cortex-M4 32-bit core operating at a frequency up to 80 MHz. It integrates 320 kB and 1 MB of volatile and static memory, respectively, which are sufficient to accommodate both model parameters and input data instances according with the prospective application scenarios.

Deployment focused on the HW-NAS models and has been implemented via the TensorFlow Lite for Microcontrollers runtime, resulting in the memory (flash and RAM) occupation and inference time outlined in Fig. 6. For the sake of clarity, it is worth recalling that, while `Qaware` and `FloatFall` have identical computational requirements (and this is the reason why we have aggregated them in Fig. 6), they can lead to remarkably different classification performances. As can be seen, the model storage (flash) required by the full float models is more than three times larger than the one demanded by the other versions, independently from the considered sensing unit. Similarly, an average drop higher than $2\times$ is witnessed for RAM when moving to the `Int8` solution; contrariwise, the temporary memory occupied by `FloatFall` and `Qaware` slightly increases as a consequence of the hybrid quantization applied by the corresponding techniques.

Alongside, the average execution time computed over 16 runs confirms that strongly quantized models exhibit significantly lower latency (more than $10\times$ speed-up for full-integer versus full-float) at the price of a little loss of performances (less than 0.3%). This is testified in Table VI, in which the average degradation over the eight sensors with respect to the PC implementation has been listed for all the four classification metrics. The same does not apply to `FullInt8` and `FloatFall`, which are slightly less performative even if being more convenient from a memory viewpoint. A negligible drop is experienced for the `FullFloat` model and might be attributed to potential rounding mechanisms when running on the low-end MCU. These outcomes align with the benefits foreseen by the TensorFlow Lite developers[3] and enforces the importance of selecting the appropriate conversion technique depending on the best compromise between the key performance indicators.

B. Sensor Energy Consumption

Starting from the time analysis above, the effects of running the deployed model on a real wireless accelerometer sensor have been assessed. More specifically, the smart sensor node in [37] has been considered for prototyping, since it offers embedded signal processing functionalities enabled by the same MCU already tested in Section VI. The sensor presents a dual accelerometer-based triggering principle necessary to maximize power efficiency, while the transmission is realized in a wireless manner via the Digi X-Bee 3 module implementing the 802.15.4 communication protocol. The latter is one of the most common choices for IoT-enabled SHM given its favorable compromise between

[3]https://www.tensorflow.org/lite/performance/model_optimization?hl=en

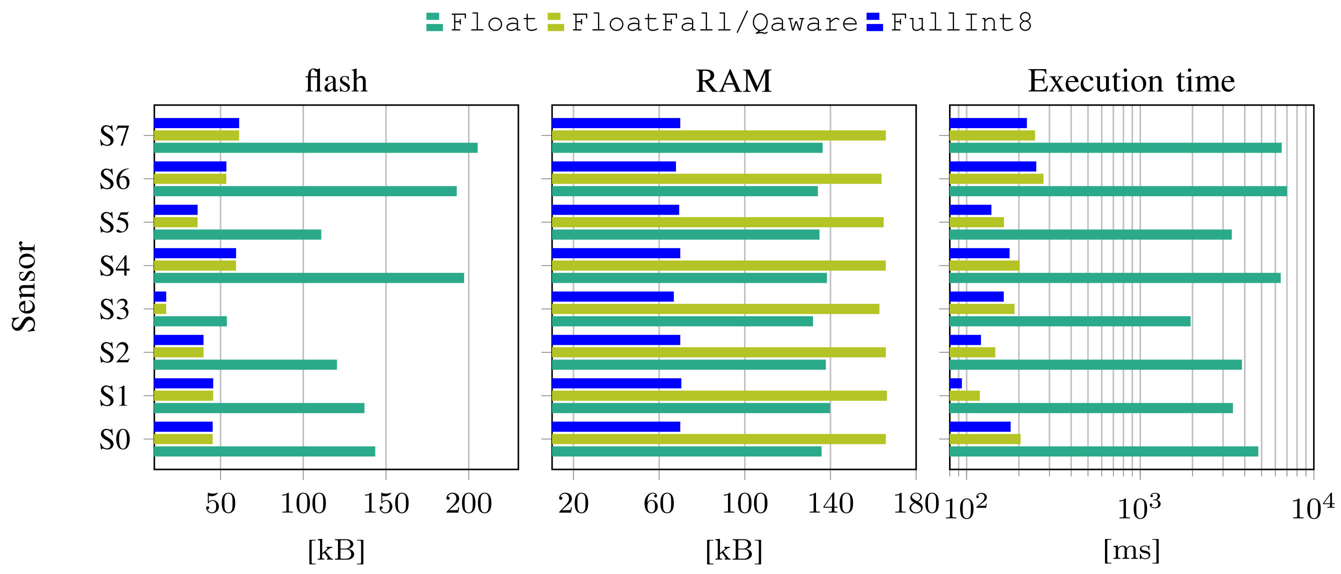


Fig. 6. HW-NAS model deployment profiling in terms of Flash, RAM, and inference time.

TABLE VI
AVERAGE LOSS IN CLASSIFICATION METRICS FOR THE QUANTIZED HW-NAS MODELS WITH RESPECT TO THE OFF-MCU IMPLEMENTATION

Model	Acc [%]	Prec [%]	Rec [%]	F1 [%]
`FullFloat`	0.01	0.05	0.02	0.05
`FullInt8`	3.51	1.90	4.79	3.79
`FloatFall`	2.55	2.17	3.36	2.99
`Qaware`	0.20	0.20	0.30	0.10

TABLE VII
ENERGY PROFILING OF THE BEST (CNN_{NAS1}) AND WORST (CNN_{NAS6}) BRANCH HW-NAS NETWORKS WHEN DEPLOYED ON THE SMART SENSOR IN [37] COMPARED WITH THE BEST CS-BASED STRATEGY (CNN*-Z24) AND A RAW IMPLEMENTATION (RAW) DISCARDING ONBOARD IMPLEMENTATION

	Current [mA]		**Time** [s]		**Energy/h** [mJ/h]		
Model	**Proc.**	**Tx**	**Proc.**	**Tx**	**Proc.**	**Tx**	**Tot**
CNN_{NAS1}	26.21	34.40	0.093	0.070	0.002	0.024	0.026
CNN_{NAS6}			6.930	0.070	0.182	0.024	0.206
CNN*-Z24			0.052	2.417	1.363	83.140	84.503
Raw			0	77.000	0	2,649.000	2,649.000

transmission bandwidth, power consumption, and range. The node is powered by a 2600-mAh battery with nominal voltage of 4.15 V and safety discharge equal to 20. The electrical characteristics in terms of average current, time, and energy required by the entire node for processing and transmission are listed in Table VII, in which the best (CNN_{NAS1}) and worst (CNN_{NAS6}) models are compared with a naive scenario without on-sensor data processing (i.e., transmission of the entire time series-label Raw) and the best CS-based strategy (CNN*-Z24).

Thereby, an analysis of the sensor energy autonomy has been performed, under the following additional assumptions: 1) measurements consist of 32 768 samples collected at 100 Hz with a duty-cycle of 1 h and 2) the MCU runs at the maximum speed of 80 MHz. While the time spent in data acquisition is constant and common to all sensors, the remaining intervals are a function of the single HW-NAS models, their computational complexity, and forced level of compression imposing the dimension of the output payload. Even if data outsourcing is the most energy-hungry operation since it demands for the largest current supply, the introduction of near-sensor data compression techniques renders the newly proposed scenario extremely efficient thanks to the very short dimension of the output tensor, reducing transmission to a negligible part of the overall energy expenditure. In this new configuration, the total energy per hour spent by the proposed approach for processing and transmitting data varies from 0.206 to 0.026 mJ/h for the most cumbersome (CNN_{NAS6} with `Float` quantization) and faster (CNN_{NAS1} with `FullInt8` quantization) model, respectively. Noteworthy, even the worst performing model is 410× more efficient than the best CS-based architecture (CNN*-Z24 [12]) in Table II, a gain which further increases to 12 859× in comparison with a raw monitoring scenarios in which no onboard data processing is embedded and the raw time series are transmitted to the aggregator: in these alternative cases, the energy consumption arises to nearly 84 and 2649 mJ/h, correspondingly. Notably, in the analyses above, the energy absorbed in the idle, sleep, and acquisition states have been neglected since they are common to all the methods; in this way, it has been possible to perform a fair comparison of the sole contribution of the data management operations.

C. Robustness Against Noisy Data and Practical Issues

Field acquisitions can be corrupted by multiple noise sources, which may originate from external factors (such as operational and environmental interference) and/or are inherent to the sensing components, such as the intrinsic sensor noise density of inertial accelerometers. Thus, it is important to evaluate the impact of such corruption on the robustness of the proposed monitoring. To this end, we have investigated how the classification performance is affected by noisy data,

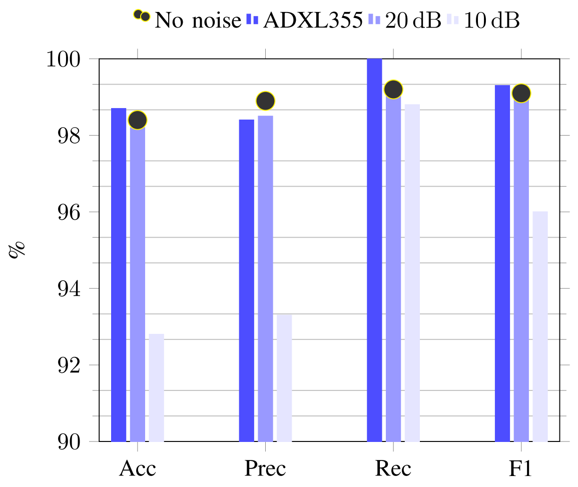


Fig. 7. Classification scores of architecture 1_W-L1-CNN$^*_{\mathrm{NAS,EE}}$ (0.01) in the presence of noisy data.

which have been obtained by adding random Gaussian noise to the original time series. Such noise properly captures the nonideal behavior of electronic components and is typically the dominant one in vibration acquisitions [38].

In particular, the noise density of the ADXL355 sensor ($20\ \mu g/\sqrt{\mathrm{Hz}}$) implies an average SNR of 45 dB (considering the data set under analysis). Under these circumstances, no appreciable degradation in the classification score of the 1_W-L1-CNN$^*_{\mathrm{NAS,EE}}$ (0.01) architecture has been observed, as reported in the bar charts of Fig. 7 (dark blue bars versus yellow-countered black markers). Then, two alternative cases have been investigated, in which the SNR has been decreased to 20 dB (violet bars) and 10 dB (light-violet bars), modeling modest and very critical condition of disturbance, respectively. Coherently, as the chart shows, the performance remains remarkably high also for the operative setting of 20 dB, while a more prominent decrement in accuracy and precision (near to 5%) verifies in the extreme case of 10 dB.

VII. Conclusion

This work presented a solution that maximizes the damage detection capability of an automatic inspection system composed of a network of vibration sensors equipped with a DNN. Exploiting an HW-NAS-based procedure, the architecture of each DNN is designed to maximize accuracy while allowing deployment on commercial MCU and reduced transmission payload. The outputs of the individual DNNs are combined for the retrieval of the global health status: the entire architecture is trained in an end-to-end fashion driven by L1 regularization, essential to further sparsify the data representation without affecting the damage detection capability. Classification scores significantly above 98.4% and a data payload as low as tens of B for the Z24 bridge data set ascertain the suitability of the proposed approach. Furthermore, the energy efficiency after deploying the designed architectures on a custom wireless accelerometer sensor is demonstrated, revealing that the quantized models can reduce the energy consumption per hour by more than three orders of magnitudes.

Possible future works will be devoted to the investigation of solutions supported by even more constrained target embedded devices and the exploration of the potential benefits of NAS when applied on the whole architectures, as well as the feasibility for its integration into an FL scenario to also optimize the training stage.

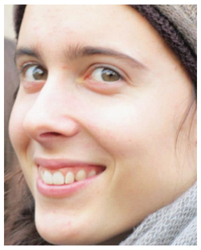
References

[1] C. A. Tokognon, B. Gao, G. Y. Tian, and Y. Yan, “Structural health monitoring framework based on Internet of Things: A survey,” *IEEE Internet Things J.*, vol. 4, no. 3, pp. 619–635, Jun. 2017.
[2] C. Scuro, P. F. Sciammarella, F. Lamonaca, R. S. Olivito, and D. L. Carni, “IoT for structural health monitoring,” *IEEE Instrum. Meas. Mag.*, vol. 21, no. 6, pp. 4–14, Dec. 2018.
[3] 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.
[4] A. Moallemi, A. Burrello, D. Brunelli, and L. Benini, “Exploring scalable, distributed real-time anomaly detection for bridge health monitoring,” *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17660–17674, Sep. 2022.
[5] F. Ni, J. Zhang, and M. N. Noori, “Deep learning for data anomaly detection and data compression of a long-span suspension bridge,” *Comput.-Aided Civil Infrastruct. Eng.*, vol. 35, no. 7, pp. 685–700, 2020.
[6] M. Russell and P. Wang, “Physics-informed deep learning for signal compression and reconstruction of big data in industrial condition monitoring,” *Mech. Syst. Signal Process.*, vol. 168, Apr. 2022, Art. no. 108709.
[7] 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, Apr. 2023.
[8] J. Sun, C. Yan, and J. Wen, “Intelligent bearing fault diagnosis method combining compressed data acquisition and deep learning,” *IEEE Trans. Instrum. Meas.*, vol. 67, no. 1, pp. 185–195, Jan. 2018.
[9] Z.-X. Hu, Y. Wang, M.-F. Ge, and J. Liu, “Data-driven fault diagnosis method based on compressed sensing and improved multiscale network,” *IEEE Trans. Ind. Electron.*, vol. 67, no. 4, pp. 3216–3225, Apr. 2020.
[10] 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, p. 2229, 2022.
[11] E. Ragusa, F. Zonzini, L. De Marchi, and P. Gastaldo, “Vibration monitoring in the compressed domain with energy-efficient sensor networks,” *IEEE Sensors Lett.*, vol. 7, no. 8, pp. 1–4, Aug. 2023.
[12] E. Ragusa, F. Zonzini, P. Gastaldo, and L. De Marchi, “Combining compressed sensing and neural architecture search for sensor-near vibration diagnostics,” *IEEE Trans. Ind. Informat.*, early access, May 13, 2024, doi: 10.1109/TII.2024.3395648.
[13] M. D. Prieto, G. Cirrincione, A. G. Espinosa, J. A. Ortega, and H. Henao, “Bearing fault detection by a novel condition-monitoring scheme based on statistical-time features and neural networks,” *IEEE Trans. Ind. Electron.*, vol. 60, no. 8, pp. 3398–3407, Aug. 2013.
[14] M. Azimi and G. Pekcan, “Structural health monitoring using extremely compressed data through deep learning,” *Comput.-Aided Civil Infrastruct. Eng.*, vol. 35, no. 6, pp. 597–614, 2020.
[15] F. Zonzini, L. Burioli, A. Gashi, N. F. Mancini, and L. De Marchi, “A tiny convolutional neural network driven by system identification for vibration anomaly detection at the extreme edge,” in *Proc. IEEE Int. Conf. Omni-layer Intell. Syst. (COINS)*, 2023, pp. 1–6.
[16] C. Gupta et al., “ProtoNN: Compressed and accurate kNN for resource-scarce devices,” in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1331–1340.
[17] A. Kumar, S. Goyal, and M. Varma, “Resource-efficient machine learning in 2 KB RAM for the Internet of Things,” in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1935–1944.
[18] A. Biglari and W. Tang, “A review of embedded machine learning based on hardware, application, and sensing scheme,” *Sensors*, vol. 23, no. 4, p. 2131, 2023.
[19] 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*.
[20] 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.

[21] 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.
[22] M. Li et al., "The deep learning compiler: A comprehensive survey," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 3, pp. 708–727, Mar. 2021.
[23] C. Li et al., "HW-NAS-bench: Hardware-aware neural architecture search benchmark," 2021, *arXiv:2103.10584*.
[24] A. Burrello, M. Risso, B. A. Motetti, E. Macii, L. Benini, and D. J. Pagliari, "Enhancing neural architecture search with multiple hardware constraints for deep learning model deployment on tiny IoT devices," *IEEE Trans. Emerg. Topics Comput.*, early access, Oct. 10, 2023, doi: 10.1109/TETC.2023.3322033.
[25] Z. Guo et al., "Single path one-shot neural architecture search with uniform sampling," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 544–560.
[26] L. Capogrosso, F. Cunico, D. S. Cheng, F. Fummi, and M. Cristani, "A machine learning-oriented survey on tiny machine learning," 2023, *arXiv:2309.11932*.
[27] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, C. Gan, and S. Han, "On-device training under 256KB memory," in *Proc. Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2022, pp. 1–14.
[28] C. Banbury et al., "MicroNets: Neural network architectures for deploying tinyml applications on commodity microcontrollers," in *Proc. Mach. Learn. Syst.*, vol. 3, 2021, pp. 517–532.
[29] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning*, vol. 4. New York, NY, USA: Springer, 2006.
[30] F. Zonzini, E. Ragusa, L. De Marchi, and P. Gastaldo, "Evaluating the effect of intrinsic sensor noise for vibration diagnostic in the compressed domain using convolutional neural networks," in *Proc. Int. Conf. Appl. Electron. Pervading Ind., Environ. Soc.*, 2023, pp. 103–108.
[31] E. Favarelli and A. Giorgetti, "Machine learning for automatic processing of modal analysis in damage detection of bridges," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–13, 2020.
[32] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1251–1258.
[33] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 156–165.
[34] A. Kusupati, M. Singh, K. Bhatia, A. Kumar, P. Jain, and M. Varma, "FastGRNN: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–23.
[35] A. Sabato, C. Niezrecki, and G. Fortino, "Wireless MEMS-based accelerometer sensor boards for structural vibration monitoring: A review," *IEEE Sensors J.*, vol. 17, no. 2, pp. 226–235, Jan. 2017.
[36] "Ultra-Low-Power Arm Cortex-M4 32-bit MCU+FPU, Rev.17," Data Sheet STM32L4R5AI, ST Microelectron., Geneva, Switzerland, Jan. 2020. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32l496ae.pdf
[37] M. Zauli et al., "A novel smart sensor node with embedded signal processing functionalities addressing vibration–based monitoring," in *Proc. Eur. Workshop Struct. Health Monit.*, 2022, pp. 1000–1008.
[38] D. Goyal and B. Pabla, "The vibration monitoring methods and signal processing techniques for structural health monitoring: A review," *Arch. Comput. Methods Eng.*, vol. 23, pp. 585–594, Dec. 2016.

Edoardo Ragusa (Member, IEEE) received the master's degree (cum laude) 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 DITEN, University of Genoa, where he teaches digital systems electronics and machine learning. He co-authored 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 a Junior Research Assistant of 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.

Luca De Marchi (Senior 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 of 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 SHM applications.

Rodolfo Zunino received the "Laurea" degree (cum laude) in electronic engineering from the University of Genoa, Genoa, Italy, in 1985.

From 1986 to 1995, he was a Research Consultant with the Department of Biophysical and Electronic Engineering, University of Genoa, where he is currently a Full Professor with DITEN. He co-authored more than 260 refereed papers in international journals and conferences. He participated in the scientific committees of several international conferences on neural networks and their applications. He has chaired the Master Course (II lev) in Cyber Security with the University of Genoa. His main scientific interests include efficient models for data representation and learning, intelligent electronic systems for neural networks, intelligent systems for security, and advanced methods for multimedia data processing.

Open Access funding provided by 'Alma Mater Studiorum - Università di Bologna' within the CRUI CARE Agreement