On-Device Fault Diagnosis with Augmented Acoustic Emission Data: A Case Study on Carbon Fiber Panels

Yuxuan Zhang, *Graduate Student Member, IEEE*, Rhys Pullin, Bengt Oelmann, and Sebastian Bader, *Senior Member, IEEE*

***Abstract*—Acoustic Emission (AE)-based fault diagnosis in Structural Health Monitoring (SHM) systems faces challenges of data scarcity and model overfitting due to the complexity of AE data acquisition and the high cost of labeling. To address these issues, this study systematically explores various data augmentation techniques for AE signal processing and evaluates their impact on model robustness and accuracy. Furthermore, given the complexity of traditional machine learning (ML) models and their deployment challenges on resource-constrained embedded devices, we investigate lightweight ML algorithms and propose a Tiny Machine Learning (TinyML)-based fault diagnosis approach. Experimental validation on a carbon fiber panel fault diagnosis case demonstrates that the proposed method significantly improves classification performance under data-scarce conditions while enabling real-time fault diagnosis on embedded systems. These findings underscore the potential of integrating data augmentation, lightweight ML algorithms, and TinyML to enhance both diagnostic accuracy and real-time performance in SHM applications.**

***Index Terms*—acoustic emission, fault diagnosis, TinyML, data augmentation, non-destructive testing, structural health monitoring, embedded devices, real-time measurement**

I. Introduction

STRUCTURAL health monitoring (SHM) is a critical technology for ensuring the safety and reliability of engineering structures, including wind turbine blades, civil infrastructure, and railway vehicles and track structures [1]. By monitoring structural conditions, SHM systems can promptly detect potential damages and predict performance degradation trends, thereby mitigating safety risks and reducing economic losses caused by sudden failures [2]. Typical SHM systems involve sensor data acquisition, signal processing, feature extraction, and state evaluation, and are widely applied in structural safety assessment, health monitoring, and damage detection [3].

Among the various tasks in SHM, fault diagnosis plays a crucial role [4], [5]. It involves detecting, identifying, and classifying structural damage using sensor data, such as acoustic, vibration, strain, and temperature signals [6]. Additionally, fault diagnosis assesses damage severity and predicts its progression, offering valuable support for maintenance planning and operational safety [7]. Traditional methods in fault diagnosis have relied heavily on signal processing techniques, such as Fourier and wavelet transforms, to manually extract features like amplitude, frequency, and energy [8], [9]. These approaches are often praised for their physical interpretability but face significant limitations when dealing with complex, multidimensional damage modes. As a result, their applicability in modern SHM, where materials and damage mechanisms are increasingly complex, has been constrained.

In recent years, the advent of data-driven approaches has revolutionized fault diagnosis in SHM [10]. Machine learning (ML) and deep learning (DL) models, with their ability to automatically extract features and identify patterns from large datasets [11], have demonstrated remarkable success in fault detection and classification tasks [12]. These methods, including algorithms such as Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), Random Forests (RFs), and Convolutional Neural Networks (CNNs), can uncover intricate relationships in high-dimensional data that traditional techniques often overlook [13]. For instance, CNN-based methods have achieved over 99% accuracy in identifying damage in structures such as bridges and concrete panels [14]. However, these approaches come with notable challenges. Their reliance on large amounts of labeled data, high computational complexity, and poor real-time performance often limit their practical deployment in SHM applications.

Acoustic Emission (AE) technology is a widely used, non-destructive fault diagnosis method that excels in identifying early-stage damage, such as micro-cracks, fiber breakage, or interfacial separation, in composite materials [15]. AE signals, generated by the release of high-frequency elastic waves during the growth of structural defects, are particularly effective for detecting subtle damage mechanisms that may not be visible through traditional inspection methods [16]. AE-based fault diagnosis methods, when integrated with ML algorithms, have shown great potential for improving SHM accuracy [17]. For instance, researchers have combined AE with ML models such as RF and CNNs to classify cracks in concrete and additive manufacturing processes, achieving high classification accuracy [18]. Despite these successes, AE data acquisition is often expensive, time-consuming, and noisy, while the scarcity of large-scale labeled datasets poses additional challenges for training robust ML models.

To address these limitations, data augmentation has emerged as a key technique to enhance the performance of data-

Yuxuan Zhang, Bengt Oelmann and Sebastian Bader are with the Department of Computer and Electrical Engineering, Mid Sweden University, Sundsvall, SE-85170, Sweden. Rhys Pullin is with the School of Engineering, Cardiff University, Cardiff, CF24 3AA, United Kingdom.

Corresponding author: Sebastian Bader (email: sebastian.bader@miun.se).

Manuscript received mm dd, yyyy.

driven models [19]. By applying transformations to original data while retaining its labels, data augmentation effectively increases the diversity of training samples, making ML models more robust and generalizable [20]. For example, noise injection, time-warping, and signal transformation techniques have been widely adopted in fields like image recognition and audio classification to expand training datasets and improve model performance [21]. In SHM, AE signals augmented with methods such as Generative Adversarial Networks (GANs) have been shown to boost damage classification accuracy by up to 10% [22]. However, the use of data augmentation for AE signals remains relatively unexplored, with limited studies systematically comparing the effectiveness of different techniques. Although the aforementioned study proposed a novel GAN-based model and compared it with other GAN variants, it did not include comparisons with traditional data augmentation approaches, leaving a gap in understanding their relative performance.

In addition to data challenges, the increasing demand for real-time fault diagnosis poses further difficulties for SHM systems. Traditional data-driven methods often rely on deep and computationally intensive models, which are not suitable for deployment on resource-constrained devices like microcontroller units (MCUs) or edge devices [23]. Cloud-based solutions, while offering the computational power needed for complex models, are constrained by bandwidth limitations and transmission latency, making them unsuitable for real-time applications [24]. To overcome these issues, researchers are turning to Tiny Machine Learning (TinyML), a paradigm that enables the deployment of lightweight ML models on low-power embedded devices [25]. TinyML not only eliminates data transmission delays but also allows for on-device data processing and immediate fault detection [26]. For example, TinyML has been successfully applied in industrial machine anomaly detection, enabling real-time diagnostics directly on MCUs [27]. Despite its promise, current research in this area often focuses on individual ML models and lacks comprehensive comparisons of lightweight algorithms for SHM.

To tackle the above-mentioned challenges, this work proposes an efficient on-device TinyML system-level solution that is trained on an augmented AE dataset, thereby jointly mitigating the problems of data scarcity and real-time fault diagnosis. Using fault classification in carbon fiber panels as a case study, the principal contributions are summarized as follows:

1) **Systematic investigation of AE data augmentation:** Multiple augmentation strategies, including noise injection and signal transformations, are systematically explored and compared. To the best of our knowledge, this is the first comprehensive analysis of their impact on small-scale AE datasets and model robustness, providing new insights into improving data quality for AE-based machine-learning models.
2) **In-depth evaluation of lightweight classifiers:** Resource-constrained scenarios are addressed by thoroughly assessing lightweight models (RFs, CNN and SVM etc.) customized for embedded, AE-based fault classification. The study elucidates the trade-offs among model complexity, classification accuracy, and real-time feasibility, offering practical guidance for model selection in AE applications.
3) **Real-time AE fault-diagnosis pipeline for embedded devices:** A complete real-time AE fault-classification pipeline is developed and optimized for low-power embedded hardware. Extensive experiments confirm its accuracy, inference latency, and resource efficiency, demonstrating deployability in industrial environments and paving the way toward practical, real-time AE monitoring systems.

The remainder of this paper is organized as follows: Section 2 introduces the AE dataset. Section 3 reviews ML algorithms used, data augmentation methods, evaluation methods and TinyML deployment toolchain. Section 4 presents experimental results and analysis. Finally, Section 5 concludes the paper and suggests future research directions.

II. Dataset

In this study, acoustic emission data were obtained from a previous study of carbon fiber panels in [3]. The subsequent sections will elaborate on the following aspects: the preparation of the panels, the configuration of the acoustic emission devices, the testing protocol, and the results derived from the collected dataset.

A. Panel Preparation

A 500 mm × 500 mm carbon fiber reinforced polymer (CFRP) panel was fabricated using unidirectional pre-preg T800S carbon fiber, constituting 56.6% by volume, embedded in an epoxy resin matrix (M21/35%/UD268/T800S, Hexcel Corporation). The complete panel is depicted in Fig. 1a. The final laminate structure comprised eight layers, arranged in a [0/90]2s configuration, resulting in a total thickness of 2.2 mm, which aligns closely with the manufacturer's specified thickness of 2.1 mm. To facilitate matrix cracking within the innermost 0° layers (3rd and 6th layers), a 25 mm crack was introduced by incising the fibers with a knife. This procedure ensured the alignment of the cracked layers, thereby enhancing the probability of matrix cracks occurring in this region, as illustrated in Fig. 1b.

The panel was cured in an autoclave following the manufacturer's specifications and subsequently subjected to C-scan inspection to confirm the absence of macroscopic defects or curing failures. To accommodate tensile loading, two holes were drilled into the panel and reinforced with aluminum square tabs. These tabs were bonded using Araldite glue prior to drilling, with a subsequent C-scan conducted to ensure that no damage occurred during the process. The final geometry of the panel is presented in Fig. 1c.

B. Acoustic Emission Setup

The AE setup utilized a Vallen AMSY-4 acquisition system coupled with Physical Acoustics Corporation WD sensors and Vallen AEP3 pre-amplifiers set to a gain of 34 dB. The entire front-end is pre-configured by the manufacturer and is factory

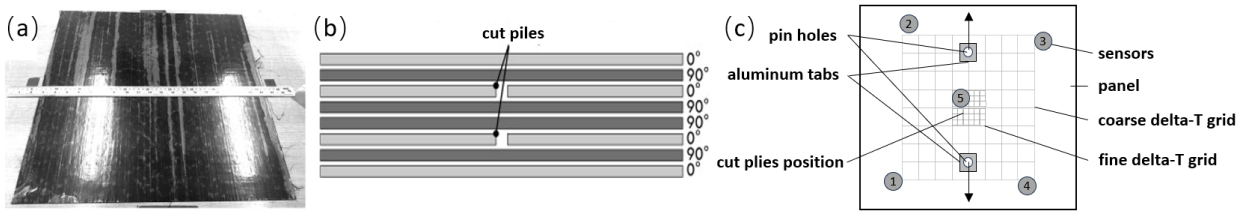


Fig. 1. Test subject fabrication and preparation: (a) Entire CFRP panel, (b) cut plies schematic, (c) artificial crack panel after manufacturing [3].

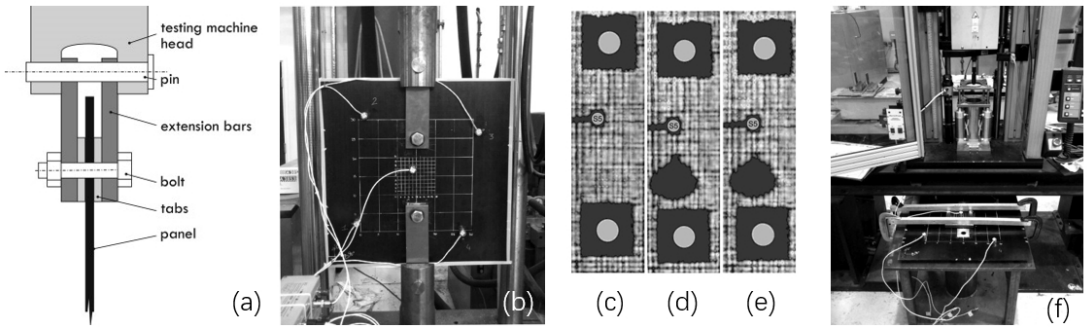


Fig. 2. Experimental setup for dataset generation: (a) Panel fitting in the tensile machine, (b) panel with sensors and fitted in the testing machine, (c) C-scan images of the panel before impact, (d) C-scan images of the panel after impact, (e) C-scan images of the panel end of test, (f) impact machine and panel fitting [3].

tuned for high sensitivity in the 95-1000 kHz ultrasound band, allowing reliable capture of high-fidelity AE signals without any additional signal conditioning. This means that a 95-1000 kHz hardware bandpass filter is built in to suppress low-frequency mechanical noise and prevent aliasing. The waveform is sampled at 5 MHz and has a recording length of 4096 points, which corresponds to 819.2 µs. According to AE theory, damaged materials release transient elastic waves in the 100–1000 kHz range, which are recorded as AE events [3], [28]. Low-frequency disturbances from testing machines, motors, and hydraulic servo controllers (typically below 50 kHz), as well as structural modal vibrations (20–80 kHz), are thus effectively excluded [29]. The acquisition threshold was set to 44.9 dB. Finally, a two-resolution square-lattice network was installed for Delta-T location calibration; however, because this study does not focus on damage localisation, the calibration procedure is not discussed in detail here.

C. Testing Plan

Following Delta-T calibration using the tensile machine (as depicted in Fig. 2a), the panel was mounted in the load testing machine (Fig. 2b) utilizing pins that passed through each extension bar to secure the panel. The panel was then bolted to the extension bars. The testing protocol consisted of fixed-amplitude batches of 5000 cycles at a frequency of 1 Hz, with C-scans conducted after each batch to monitor the progression of damage. Load increments were applied based on AE data, maintaining an R ratio (minimum load/maximum load) of 0.1 to prevent compressive loads and ensure sufficient preload. After acquiring adequate AE signals from the artificially introduced crack region, the panel was subjected to an impact using an Instron Dynatup 9250HV impact machine, delivering 14J of energy at a location away from the crack (Fig. 2f). This impact was designed to generate signals from both the pre-existing crack and the resulting delamination. The panel was subsequently retested, and C-scans were performed after each load batch. The scan images, shown in Fig. 2c–e, clearly indicate significant damage in the vicinity of sensor 5.

D. Dataset Content

For the purposes of classification reproducibility, only the signals recorded by sensor 5 were considered, as detailed in [3]. The collected AE data were categorized into two groups based on signal characteristics such as amplitude, duration, rise time, number of zero-crossings, energy, frequency center-of-gravity, and peak frequency, utilizing self-organized mapping, as well as energy and activity maps. The resulting labeled dataset comprises a total of 207 AE signals, including 103 signals associated with matrix cracking and 104 signals related to impact-induced in-plane delamination. Figure 3 provides examples of the two types of waveforms, with each sample consisting of 4096 data points.

III. Methods

This section outlines the proposed methodology for on-device carbon fiber panel fault classification, which includes four key components: ML algorithm selection, data augmentation techniques, evaluation methods for robustness and accuracy, and deployment via the TinyML toolchain. The workflow of the study is depicted in Fig. 4. The methodology begins with training multiple ML algorithms using both the raw AE dataset and augmented data. Data augmentation techniques, combining traditional signal transformations and deep learning-based generative approaches, are employed to expand the dataset and improve model robustness. The robustness and accuracy of the

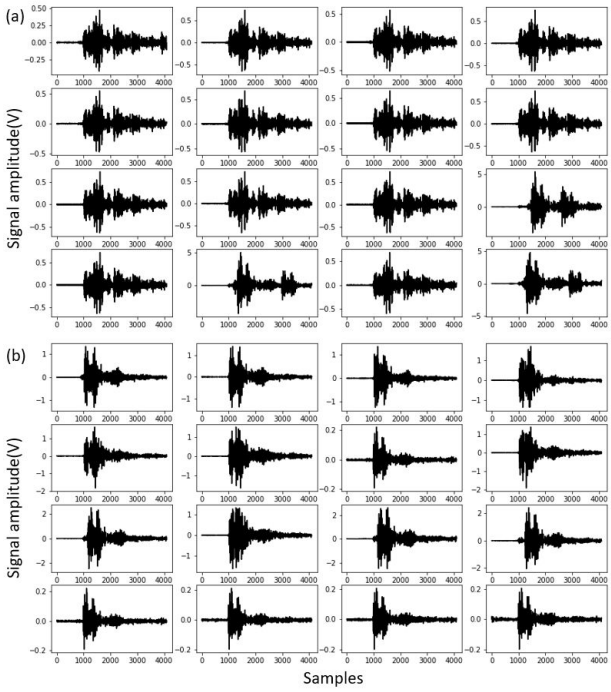


Fig. 3. Examples for AE signals waveforms for (a) delamination faults and (b) matrix cracking faults.

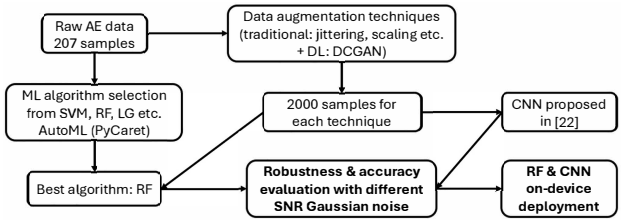


Fig. 4. Proposed carbon fiber panels damage identification approach on MCU.

ML models are evaluated under varying conditions of signal-to-noise ratio (SNR) to simulate real-world scenarios. Finally, the best-performing models are deployed on an MCU using the TinyML toolchain to enable real-time fault classification.

A. ML Algorithms Selection

In this study, we employed PyCaret[1], an open-source Python library for ML, to streamline the experimental workflow from hypothesis formulation to insight generation. PyCaret enables rapid and efficient execution of end-to-end ML experiments, making it particularly suitable for classification damage types in carbon fiber panels. A total of 12 ML algorithms were evaluated in this study, including Random Forest Classifier, Gradient Boosting Classifier (GBC), AdaBoost Classifier, Extra Trees Classifier, Decision Tree Classifier (DT), Linear Discriminant Analysis (LDA), K-Neighbors Classifier, Naive Bayes, Logistic Regression (LR), Ridge Classifier, Quadratic Discriminant Analysis (QDA) and Support Vector Machine. Cross-validation was conducted during training to assess the performance of each estimator in the library, ensuring reliable model evaluation. The hyperparameters of each algorithm were automatically optimized using the PyCaret tool. After training, we extracted the corresponding model's optimal hyperparameters and manually trained the model using Scikit-learn[2], saving it for further testing.

TABLE I
OVERALL ARCHITECTURE OF THE 1D CNN MODEL PROPOSED IN [24].

	Layer	Hyperparameters	Feature Maps	Weights
1	Input	1000x1	1000x1	0
2	Conv1D-1	7x1	32@994x1	256
3	MaxPool-1	8x1	32@124x1	0
4	Conv1D-2	5x1	48@120x1	7,728
5	MaxPool-2	4x1	48@30x1	0
6	MaxPool-3	4x1	48@7x1	0
7	Flatten	-	336x1	0
8	Dense-1	32	32x1	10,784
9	Dense-2	32	32x1	1,056
10	Output	2	2x1	66
	Total Parameters			19,890

For the convolutional neural network (CNN) approach, we utilized a lightweight 1-D CNN model adapted from the architecture proposed in [24] for AE signal-based damage classification in concrete materials. The model processes raw AE signals as input and outputs damage classifications. However, while the original CNN was designed to classify three damage types, our application focused on two damage types, necessitating adjustments to the output layer's hyperparameters to reflect this difference. In addition, the raw AE signal samples are downsampled to 1000x1 from 4096x1 to fit the input of CNN.

The CNN architecture consists of two 1-dimensional convolutional layers for feature extraction, three max-pooling layers for dimensionality reduction, and three fully connected dense layers for classification. Sparse categorical cross-entropy was used as the loss function, optimized using the Adam optimizer with a learning rate of 0.0001. All ML and CNN models were implemented and trained using TensorFlow[3] on a Windows platform, leveraging the CUDA framework for GPU acceleration with a GeForce GTX 3090 GPU. The detailed CNN architecture is outlined in Table I.

It is important to note that we deliberately refrain from designing new algorithms in order to evaluate the generalizability of the proposed data augmentation and quantization deployment pipeline. By keeping the model architecture unchanged, we isolate the experimental variables strictly at the data processing level, thereby avoiding performance fluctuations introduced by new models. This allows for a clearer quantification of the contribution of data augmentation itself and facilitates reproducibility and fair comparisons in future studies.

[1]https://github.com/pycaret/pycaret

[2]https://scikit-learn.org/stable/

[3]https://www.tensorflow.org

B. Data Augmentation Techniques

To address the challenges posed by the small size of the original AE dataset, data augmentation techniques were employed to generate additional samples, enhancing model robustness and generalization. In this study, half of the original dataset (103 samples) was set aside as the test dataset for performance evaluation, while the remaining half (104 samples) was used as the base for augmentation. The augmented dataset was then used to train and validate the best algorithm in selection process and CNN model proposed in authors' previous study.

This study explores seven traditional time-series signal augmentation methods and one deep learning-based approach, DCGAN. These techniques were applied directly to the raw AE data to preserve its original characteristics while expanding the dataset. Each augmentation technique was used to generate 2,000 samples (1,000 for each class), ensuring sufficient data for model training and evaluation. These augmented datasets were critical in avoiding overfitting, particularly for the CNN model, while improving classification performance and robustness. We intentionally rely only on established data augmentation techniques, rather than introducing new ones, to create a repeatable common baseline that enables like-for-like comparisons across studies. The impact of each augmentation method on model accuracy and robustness is discussed in Section 4. The data augmentation techniques that have been considered are:

Jittering: The jittering data augmentation technique can be mathematically represented in (1). Where $\mathbf{X}$ denotes the original data, $\mathbf{X_j'}$ represents the augmented data, and $\mathcal{N}(0, \sigma^2)$ is a Gaussian noise vector with mean 0 and variance σ^2. The parameter σ controls the standard deviation of the noise distribution. In this work, we set σ to 0.0001 to introduce a small but meaningful perturbation to the data.

$$\mathbf{X_j'} = \mathbf{X} + \mathcal{N}(0, \sigma^2) \tag{1}$$

Scaling: The scaling data augmentation technique involves multiplying each element of the input data $\mathbf{X}$ by a scaling factor drawn from a Gaussian distribution. Specifically, the scaling factor is a random variable following a normal distribution with a mean of 1 and a variance of σ^2. Mathematically, this process can be expressed in (2). Where $\mathbf{X_s'}$ represents the augmented data. By setting σ to 0.1, the scaling factors are kept close to 1, introducing slight variations to the data without significantly altering its original characteristics.

$$\mathbf{X_s'} = \mathbf{X} \cdot (1 + \mathcal{N}(0, \sigma^2)) \tag{2}$$

Magnitude Warping: Magnitude warping can be interpreted as adding smoothly varying noise to all data points across the AE sample to simulate natural variations. The process is divided into two parts - Generate Random Curves and Magnitude Warping. The random curves are generated by defining a set of knots and fitting a cubic spline to Gaussian-distributed values at these knots. The process can be described as in (3, 4, 5). By setting σ to 0.001, we ensure that the values drawn from the Gaussian distribution are centered around 1 with very small deviations, leading to minor and controlled perturbations of the original data. Where α are the knot positions, β are the Gaussian-distributed values at these knots, $f_{CubicSpline}$ is the cubic spline function and $\gamma_{\mathbf{x}}(\mathbf{t})$ is the cubic spline function fitted to the knot positions and values.

$$\alpha = \left[0, \frac{N}{k+1}, 2 \cdot \frac{N}{k+1}, \ldots, N\right]^\top \tag{3}$$

$$\beta = \mathcal{N}(1, \sigma^2, k+2) \tag{4}$$

$$\gamma_{\mathbf{x}}(\mathbf{t}) = f_{CubicSpline}(\alpha, \beta) \tag{5}$$

Then the magnitude warping is applied by element-wise multiplication of the input data with the generated random curve as in (6). Where $\mathbf{X}$ is the original data, $\mathbf{X'_m}$ is the warped data and $\gamma_{\mathbf{x}}(\mathbf{i})$ is the value of the random curve at position i.

$$\mathbf{X'_m}[i] = \mathbf{X}[i] \cdot \gamma_x(i) \quad \text{for} \quad i = 0, 1, \ldots, N-1 \tag{6}$$

Time Warping: The time warping distorts the time steps of the input data by generating random curves and interpolating the data accordingly. There are three steps to achieve it-Generate Random Curves, Distort Time Steps and Time Warping. The first step is the same as in magnitude warping. The second step uses the generated random curves as time intervals, cumulates these intervals to create distorted time steps, and scales them to match the original time range. The mathematical representation is shown in (7).

$$\tau_{\mathbf{cum}} = \mathbf{\Gamma} \cdot \frac{N-1}{\mathbf{\Gamma}[-1]} \tag{7}$$

Where

$$\mathbf{\Gamma} = f_{CumSum}\left(\gamma_x\big(f_{range}(N)\big)\right) \tag{8}$$

$\gamma_x(t)$ is the same function as in (5), $f_{range}(N)$ represents the range of indices from 0 to $N-1$, and f_{CumSum} is the cumulative sum function. The resulting $\tau_{\mathbf{cum}}$ represents the distorted time steps, which are then used to warp the original data shown in (9). $\mathbf{X'_t}$ is the augmentated data.

$$\mathbf{X'_t}[i] = \mathbf{X}(\tau_{\mathbf{cum}}[i]) \quad \text{for} \quad i = 0, 1, \ldots, N-1 \tag{9}$$

Random Sampling: Random sampling is used to augment time series data by selecting random timesteps for resampling. Let N denote the length of the original time series $X = \{X_t\}_{t=0}^{N-1}$, and M be the number of samples, where $M < N$. Random indices $\{t_i\}_{i=0}^{M-1}$ are generated in (10, 11).

$$t_0 = 0, \quad t_{M-1} = N-1, \tag{10}$$

$$t_i \sim \text{Uniform}(1, N-2), \quad \text{for } i = 1, 2, \ldots, M-2. \tag{11}$$

The selected indices $\{t_i\}$ are sorted to maintain order, and the resampled time series $\mathbf{X'_r}$ is obtained by interpolating the original signal $\mathbf{X}$ at the uniformly spaced time points $\tau_k = k$ for $k = 0, 1, \ldots, N-1$ shown in (12). This method generates

variations by resampling the signal at random intervals while preserving the overall structure of the data.

$$\mathbf{X}'_{\mathbf{r}} = \mathbf{X}(t_i) \quad \text{interpolated at } \tau_k. \tag{12}$$

Permutation: Permutation is applied to augment time series data by randomly reordering segments of the signal. Let N denote the length of the original time series $\mathbf{X} = \{\mathbf{X}_t\}_{t=0}^{N-1}$, and $n_{\text{perm}} = 4$ be the number of segments. The segment boundaries $\{s_i\}_{i=0}^{n_{\text{perm}}}$ are determined in (13, 14).

$$s_0 = 0, \quad s_{n_{\text{perm}}} = N, \tag{13}$$

$$s_i \sim \text{Uniform}(\text{minSegLength}, N - \text{minSegLength}) \tag{14}$$

subject to s_i being sorted and $s_{i+1} - s_i > \text{minSegLength}$ for all i.

Once the segment boundaries are defined, the segments are permuted according to a random permutation π shown in (15). Where, $\mathbf{X}_{[s_{\pi(i)}, s_{\pi(i)+1})}$ represents the i-th segment of the time series, selected based on the random permutation π applied to the indices $\{0, 1, \ldots, n_{\text{perm}} - 1\}$. This method introduces variability by rearranging non-overlapping segments of the time series, preserving local temporal patterns while altering the global structure. The parameters used in this work are $n_{\text{perm}} = 4$ and $\text{minSegLength} = 100$, ensuring meaningful segment sizes and sufficient variation in the augmented data.

$$\mathbf{X}'_{\mathbf{p}} = \bigcup_{i=0}^{n_{\text{perm}}-1} \mathbf{X}_{[s_{\pi(i)}, s_{\pi(i)+1})}. \tag{15}$$

Gaussian Noise: Gaussian noise is a very common technique for data augmentation. In this paper, Gaussian noise with a signal-to-noise ratio (SNR) of 95 dB is used. Firstly, the average signal power is calculated in dB, then the required noise power is calculated in Watts and finally the generated Gaussian noise is added to the original signal. The mathematical process can be represented as follows in (16, 17, 18).

$$dB_{SigAvg} = 10 \log_{10} \left(\frac{1}{N} \sum_{i=1}^{N} a_{\text{volts}}[i]^2 \right) \tag{16}$$

$$W_{NoiseAvg} = 10^{\left(\frac{dB_{SigAvg} - dB_{TargetSnr}}{10}\right)} \tag{17}$$

$$\mathbf{X}'_{\mathbf{g}} = \mathbf{X} + \mathcal{N}\left(0, \sqrt{W_{NoiseAvg}}\right) \tag{18}$$

DCGAN: The DCGAN integrates the principles of CNNs and GANs. Specifically, the generator employs a CNN architecture to map random vectors (typically noise) into image space, producing image samples that closely resemble real images. The discriminator, also a convolutional neural network, is tasked with distinguishing between the synthetic images generated by the generator and the actual real images. The primary objective of a GAN is to train the generator and discriminator in a minimax game, which is governed by (19).

$$\begin{aligned} \min_G \max_D V(D,G) = &\mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] \\ &+ \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]. \end{aligned} \tag{19}$$

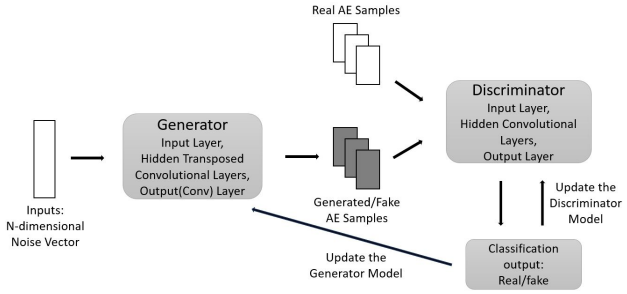


Fig. 5. Block diagram of the DCGAN model generation.

where:

- x is a real sample from the training data.
- z is a noise vector sampled from a prior distribution $p_z(z)$
- $G(z)$ is the fake sample generated by the Generator from noise z.
- $D(x)$ is the Discriminator's output probability that x is real.
- $D(G(z))$ is the Discriminator's output probability that $G(z)$ is real.

The standard DCGAN architecture, depicted in Fig. 5, aims to maximize the discriminator's ability to differentiate between real and generated images. This framework can also be adapted for data augmentation of one-dimensional sound signals. During adversarial training, the generator attempts to produce AE signals that are sufficiently realistic to deceive the discriminator, while the discriminator endeavors to distinguish between the fake AE signals generated by the generator and the authentic AE signals. This competitive and adversarial interaction drives the two networks to progressively enhance their performance, ultimately enabling the generator to create highly realistic AE signals.

The structure of the generator and discriminator is detailed in Table II. The discriminator D comprises five convolutional layers, each followed by a batch normalization layer and an activation function layer. The convolutional layers are defined by four key parameters: the number of filters, filter size, stride length, and padding. In this configuration, the stride length is set to 2 for all layers except the first, which uses a stride length of 1. The choice of a stride length of 2 in most layers is intended to retain more detailed features of the signal, thereby enhancing the model's performance. As the feature map size is halved after each convolution with a stride of 2, the number of filters in each successive layer is doubled to preserve the representational capacity. Zero padding is applied to the input of each convolutional layer to prevent alterations in the output size due to the convolution process. Larger filters are used in the first layer to capture broader information, while the last layer employs a sigmoid activation function for binary classification (0 or 1). The generator G consists of five transposed convolutional (inverse convolutional) layers, each also followed by a batch normalization layer and an activation function layer. Each transposed convolutional layer, except for the final one, corresponds to a convolutional layer in the discriminator. The rectified linear unit (ReLU) activation function is applied to all layers in G except the output layer,

TABLE II
ARCHITECTURE OF DCGAN GENERATOR AND DISCRIMINATOR

Block	**Generator** G	**Discriminator** D
1	DeConv-BN-ReLU Filter: (512, 64, 1)	Conv-BN-LReLU Filter: (64, 4, 2)
2	DeConv-BN-ReLU Filter: (256, 4, 2)	Conv-BN-LReLU Filter: (128, 4, 2)
3	DeConv-BN-ReLU Filter: (128, 4, 2)	Conv-BN-LReLU Filter: (256, 4, 2)
4	DeConv-BN-ReLU Filter: (64, 4, 2)	Conv-BN-LReLU Filter: (512, 4, 2)
5	DeConv Filter: (1, 4, 2)	Conv-Sigmoid Filter: (1, 64, 1)

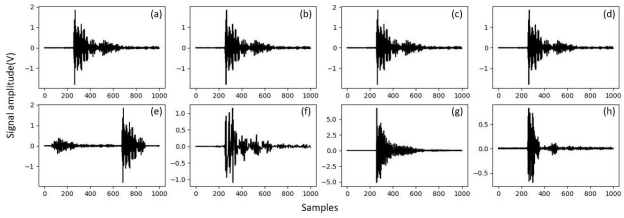


Fig. 6. Augmented AE data waveform examples using (a) jittering; (b) scaling; (c) magnitude warping; (d) time warping; (e) permutation; (f) random sampling; (g) Gaussian noise; (h) DCGAN.

as models incorporating ReLU are typically easier to optimize. Conversely, the discriminator D uses the leaky ReLU activation function with a negative slope factor of 0.2 to prevent dying ReLUs and to allow a small gradient when the unit is not active. The training framework employed is TF, and the model is optimized using the Adam optimizer, with a learning rate set to 0.0002.

Figure 6 presents a graphical representation of the data augmentation technique applied to AE signals, derived from the carbon fiber panels used in this study. It is important to note that the results of the permutation technique exhibit significant deviations from the original signal due to the random segmentation and cropping involved. Consequently, this method has been excluded from further consideration in subsequent analyses within this paper.

C. Robustness and Accuracy Evaluation

In terms of accuracy, the F1-score is a widely used metric in classification tasks, particularly effective for imbalanced datasets. It balances Precision and Recall, offering a single measure that considers both false positives and false negatives. Mathematically, the F1-score is defined in (20).

$$F1 - score = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{20}$$

Where Precision and Recall are given by (21).

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}. \tag{21}$$

Here, TP represents true positives, FP false positives, and FN false negatives.

The F1-score uses the harmonic mean of Precision and Recall, ensuring sensitivity to both metrics. This makes it particularly suitable for scenarios where a balance between identifying all relevant instances (high Recall) and minimizing false positives (high Precision) is critical.

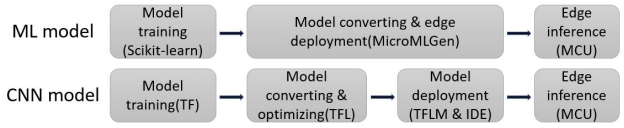


Fig. 7. TinyML deployment toolchain.

Regarding the impact of data augmentation, except the permutation techniques, we applied seven different data augmentation techniques to enhance the model's generalization ability and robustness. These techniques include jittering, scaling, magnitude warping, time warping, random sampling, adding Gaussian Noise and DCGAN. The primary objective was to simulate real-world variations in input data, enabling the model to adapt to diverse operational conditions.

Specifically, each data augmentation technique was applied to half of the original dataset (104 samples) to generate 2000 augmented samples, with 1000 corresponding to delamination and 1000 to matrix cracking. The augmented datasets were then split into training (1600 samples) and validation (400 samples) subsets in an 80%–20% ratio. Subsequently, each augmented dataset was used to train a CNN model and an optimal model selected via AutoML. This process resulted in the training of seven CNN models and seven AutoML-optimized models under identical training conditions. A 10-fold cross-validation approach was employed during training. Upon completion of the training phase, all models were evaluated on the remaining half of the original dataset (the test dataset - 103 samples) ten times to assess their average performance on non-augmented data.

For robustness evaluation, the trained models were further tested under noisy environments. Gaussian noise with varying signal-to-noise ratio (SNR) levels of 95 dB, 85 dB, 75 dB, 65 dB, 55 dB, 45 dB, 35 dB, 25 dB, and 15 dB was added to the original test dataset. The models used for evaluation were trained and validated on the datasets augmented with the seven techniques. The objective of introducing Gaussian noise was to assess the model's recognition accuracy and resilience under different noise levels, providing insights into its real-world applicability. Additionally, to ensure evaluation reliability, each model trained on augmented datasets underwent ten independent test runs, and the results were averaged to obtain representative performance metrics.

D. Tiny Machine Lerning

TinyML represents a critical intersection where embedded ML applications, hardware infrastructure, software frameworks, and advanced algorithms converge. This field enables the deployment and execution of ML and DL models on MCUs, digital signal processors (DSPs), and specialized ultra-low-power processors. The toolchains for TinyML comprise several key platforms, including Google's TensorFlow Lite for

TABLE III
TECHNICAL SPECIFICATIONS OF MCU PLATFORM

Test board	Arduino Nano 33 BLE
MCU	nRF52840
CPU core	Arm Cortex M4
CPU Frequency	64MHz
SRAM	256KB
Flash	1MB
Operating voltage	3.3V
Current consumption	52 μA/MHz

Microcontrollers[4] (TFLM), STM32Cube.AI[5], and the open-source ML deployment tool MicroMLGen [6].

Figure 7 illustrates the toolchains utilized in this study. ML model training was conducted using Scikit-learn, an open-source Python machine learning library. The trained ML algorithms were then deployed onto the MCU using MicroMLGen, which converts trained ML algorithms into C-code. For CNN models, the study employed the TFLM framework, specifically designed to deploy DL models on MCUs with constrained memory resources. Notably, TFLM operates without reliance on an operating system, standard C/C++ libraries, or dynamic memory allocation. The deployment process using TFLM involves model training with TF, conversion to the TensorFlow Lite (TFL) format, and the optimization technique used in this study is post-training quantization. It converts a Float32 model into an Int8 model, specifically, the weights, biases, inputs, outputs, and activations of the model are quantized to Int8 to reduce memory footprint (about a quarter of the original), inference time, and energy consumption.

The target deployment platform in this study is the Arduino Nano 33 BLE, with the Arduino Integrated Development Environment (IDE) serving as the primary development environment. Table III outlines the key technical specifications of the Arduino Nano 33 BLE.

During deployment, the process is divided into two parts: one for the CNN model and the other for the ML algorithm. For both approaches, we used half of the original dataset without data augmentation (103 samples) as the test dataset to evaluate performance. The model inference was executed solely on the MCU, where we recorded key metrics such as inference time, power consumption, and memory usage. For the CNN model, we validated the consistency of inference results obtained from the TFLM framework on a PC and the results of the model converted to C-code and deployed on the MCU (comparing the probability values for two types of damage). Since the consistency was confirmed, inference results were collected using the TFLM framework on the test dataset. For the ML model, we similarly verified the consistency between the inference results of the C-code generated by MicroMLGen running on the MCU and those obtained using Scikit-learn on a PC. Therefore, performance data collection for RF was conducted based on Scikit-learn inference on the PC.

[4]https://ai.google.dev/edge/litert/microcontrollers/overview
[5]https://stm32ai.st.com/stm32-cube-ai/
[6]https://github.com/eloquentarduino/micromlgen

IV. RESULTS AND DISSCUSSION

This section presents the results of the study and discusses their implications in the context of AE-based damage classification for carbon fiber panels. The primary focus is on evaluating the proposed methodology in terms of model selection, the impact of data augmentation, and the feasibility of deploying models on an resource-constrained MCU.

First, the selection of the optimal ML algorithm is discussed by comparing the performance of 12 widely used classifiers. This establishes the baseline for subsequent experiments and highlights the best-suited algorithm for AE signal classification. Second, the impact of various data augmentation techniques, including both traditional signal transformations and deep learning-based methods, is analyzed. The performance of the models trained on augmented datasets is assessed in terms of accuracy and robustness to noise. Finally, the on-device performance of the selected models is evaluated, including their deployment on an MCU. This involves analyzing key metrics such as inference time, memory usage, energy consumption, and classification accuracy to validate the practicality of the proposed solution for real-time SHM.

A. ML Algorithm Performance

To identify the optimal machine learning (ML) algorithm for AE-based damage classification, 12 widely used classifiers were evaluated using cross-validation on the original dataset. Table IV presents the performance metrics of the ML models for the classification task. Among these models, the Random Forest (RF), Gradient Boosting, AdaBoost, and Extra Trees classifiers demonstrate strong performance across all metrics, indicating that they are among the top-performing models for classification. These models exhibit high accuracy, strong recall, and a good balance between precision and recall. The Random Forest model achieves the highest accuracy of 95.76% and the highest F1 score of 0.9613 among these four methods. It also performs well in terms of recall (0.9857) and precision (0.9450). While Gradient Boosting, AdaBoost, and Extra Trees classifiers also achieve high accuracies of 93% to 94%, their overall performance is somewhat lower than that of the Random Forest model.

The Decision Tree Classifier, Linear Discriminant Analysis, K Neighbors Classifier, Naive Bayes, Logistic Regression, and Ridge Classifier exhibit moderately good performance, each demonstrating distinct advantages in specific metrics. For example, the Ridge Classifier achieves the highest precision at 0.9467. However, these models do not perform as well as the first-tier classifiers, such as Random Forest, in terms of accuracy. For instance, the Decision Tree Classifier, which is the best among this group, achieves an accuracy of only 92.24%. Additionally, these models lag behind in other performance metrics, indicating potential limitations in handling complex relationships within the data or the need for further optimization to improve their performance.

The third tier includes Quadratic Discriminant Analysis and Support Vector Machine, all of which perform relatively poorly. Although Quadratic Discriminant Analysis achieves the highest recall score of 1.0, its other scores, such as

TABLE IV
PERFORMANCE AND HYPERPARAMETERS OF ML AND CNN MODELS (HYPERPARAMETERS NOT SHOWN ARE SET TO DEFAULT).

#	Model	Hyperparameters	Accuracy	Recall	Precision	F1 Score
1	Random Forest	`n_estimators=100, min_samples_split=2, min_samples_leaf=1, max_features=sqrt`	**0.9576**	0.9857	0.9450	**0.9613**
2	Gradient Boosting	`n_estimators=100, learning_rate=0.1, max_depth=3, subsample=1.0, min_samples_split=2`	0.9429	0.9714	0.9260	0.9455
3	Ada Boost	`n_estimators=50, learning_rate=1.0, algorithm=SAMME.R, estimator=None`	0.9362	0.9589	0.9307	0.9404
4	Extra Trees	`n_estimators=100, max_depth=None, min_samples_split=2, max_features=sqrt`	0.9362	0.9857	0.9146	0.9443
5	Decision Tree	`criterion=gini, max_depth=None, min_samples_split=2, min_samples_leaf=1`	0.9224	0.9304	0.9307	0.9237
6	Linear Discriminant	`solver=svd, covariance_estimator=None, shrinkage=None, n_components=None`	0.9081	0.9161	0.9289	0.9043
7	K Neighbors	`n_neighbors=5, weights=uniform, leaf_size=30, algorithm=auto`	0.8671	0.7911	0.9357	0.8516
8	Naive Bayes	`priors=None, var_smoothing=1e-09`	0.7990	0.6536	0.9383	0.7542
9	Logistic Regression	`penalty=l2, C=1.0, intercept_scaling=1, solver=lbfgs, max_iter=1000`	0.7414	0.5250	0.9464	0.6565
10	Ridge Classifier	`alpha=1.0, class_weight=None, copy_X=True, fit_intercept=True, solver=auto`	0.7281	0.4857	**0.9467**	0.6335
11	Quadratic Discriminant	`reg_param=0.0, priors=None, store_covariance=False, tol=0.0001`	0.5000	**1.0000**	0.5000	0.6664
12	Support Vector Machine	`alpha=0.0001, kernel=Linear, max_iter=1000, n_iter_no_change=5`	0.4795	0.5304	0.2879	0.3730
13	CNN in [24]	`learning_rate=0.0001, optimizer=Adam, loss=sparse_categorical_cross-entropy`	Overfitting			

accuracy, precision, and F1, are notably low. Meanwhile, the Support Vector Machine fail to perform the classification task effectively, struggling to balance accuracy and recall and facing challenges in correctly identifying positive samples while maintaining precision.

In summary, the results of AutoML indicate that the Random Forest Classifier is the most suitable algorithm for the AE dataset of carbon fiber panels. Conversely, the CNN model suffers from overfitting due to the limited amount of data, preventing effective training and making it unsuitable for comparison at this stage. In the following sections, data augmentation methods will be employed to enhance CNN training, enabling performance comparisons.

B. Data Augmentation Impact

To address the limitations of the small AE dataset, various data augmentation techniques were applied to enhance the training data and improve model performance. These techniques included seven traditional signal transformations—such as jittering, scaling, and time-warping—and a deep learning-based method using DCGAN. The augmented datasets were used to train both RF and CNN model, and their performance was compared to models trained on the original dataset.

Figures 8 and 9 illustrate the impact of the data augmentation techniques on the robustness of the CNN and RF algorithms, specifically in terms of accuracy. The line plots

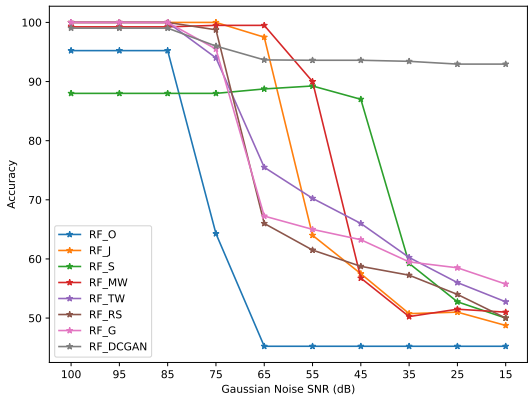


Fig. 8. RF model accuracy as a function of SNR after data augmentation.

representing the RF algorithm show that it is capable of maintaining an accuracy of 95.23% when utilizing the original small dataset at SNR of 100, 95, and 85 dB. However, as the noise level increases, the accuracy declines rapidly, nearly eliminating the model's classification ability. Conventional time-series signal data augmentation techniques, such as jittering, magnitude warping, time warping, random sampling, and Gaussian noise, enhance the robustness of the RF algorithm

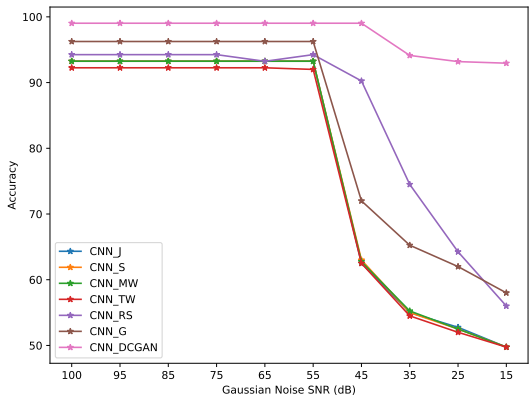


Fig. 9. CNN model accuracy as a function of SNR after data augmentation.

to a certain extent. These techniques delay the model's failure from SNR=85 dB to a range of SNR=75, 65, and 55 dB. However, their accuracy significantly decreases at an SNR of 55 dB. The scaling algorithm, due to its inherent random signal scaling, maintains its noise immunity up to SNR=35 dB, but the accuracy drops significantly beyond that point. Ultimately, DCGAN, a widely used deep learning-based data augmentation method, demonstrates its superiority in enhancing both robustness and accuracy. Under low noise conditions, DCGAN achieves an accuracy of 99.03%, surpassing the unaugmented model's accuracy of 95.23%. RF models trained on datasets augmented by this method retain relatively high accuracy even as the noise level increases. For instance, the model maintains a 93% recognition accuracy as the SNR decreases from 55 to 15 dB, indicating that DCGAN is highly effective in maintaining accuracy under increasing noise levels and in improving the performance of RF models.

For the CNN model, the use of data augmentation methods has proven to be a crucial solution to mitigate the overfitting challenges inherent in smaller datasets. Each data augmentation strategy effectively facilitates the training of CNN networks, resulting in accuracy rates exceeding the 90% threshold. Furthermore, the original structure of the CNN model, tracing back to [24], highlights its strong generalization potential, as evidenced by its successful application to the carbon fiber panel AE dataset. This success underscores the model's adaptability for various AE-based material damage identification applications. The integration of different data augmentation methods with the CNN algorithm not only ensures excellent accuracy but also enhances the robustness of the recognition results. Techniques such as jittering, scaling, magnitude warping, time warping, random sampling, and Gaussian noise perform commendably, maintaining accuracy between 93.25% and 96.25% until the SNR exceeds 55 dB. However, as SNR decreases further, a rapid decline in accuracy occurs, compromising the model's effective classification ability. In contrast, reflecting its performance in the RF algorithm, the DCGAN method's accuracy drops from 99.03% to 93.42% at SNR=35 dB and stabilizes around 93% thereafter. Notably, DCGAN demonstrates a remarkable capacity to enhance the model's robustness compared to all other data augmentation techniques used. This ability to maintain accuracy despite decreasing signal-to-noise levels highlights DCGAN's efficacy in bolstering the CNN model's resistance to noise-induced interference.

TABLE V
DEPLOYMENT PERFORMANCE OF RF AND CNN MODELS ON MCU

Model	**Accuracy** (%)	**RAM** (KB)	**Flash** (KB)	**Time** (ms)	**Energy per Inference**(mJ)
RF	99.03	45.88	111	0.46± 0.01	0.0015
CNN(Q)	99.03	67.8	31.1	165± 0.31	0.548

In conclusion, after comparing the performance of various data augmentation techniques in RF and CNN models, DCGAN emerges as a superior method compared to all traditional augmentation techniques. It significantly enhances the resilience of both RF and CNN models while also improving the accuracy of RF models before and after augmentation. Furthermore, the application of data augmentation techniques substantiates the substantial generalization ability of the CNN model proposed in [24] for material damage recognition based on AE techniques. Ultimately, both the CNN model and the RF algorithm effectively address the classification problem. However, the CNN model demonstrates greater robustness compared to the RF algorithm.

C. On-device Performance

This section summarizes the results of RF and CNN models on-device performance, as presented in Table V. The float32 precision CNN models cannot be deployed on the target MCU due to the number of parameters, the memory footprint is too large. Therefore the post training quantization is utilized to the model with TFL support. This process converts the model from a float32 format to an int8 format, thereby reducing both RAM and flash memory consumption. The quantized model requires only 31.1 KB of flash and 67.8 KB of RAM, representing a 64% and 73% reduction from the pre-quantized state, respectively.

Comparisons between the inference results from the TFL interpreter on the development board and a PC indicate that the CNN model produces identical outputs in both environments. The quantized model was then tested for accuracy using the TFL interpreter, achieving an accuracy of 99.03%. This demonstrates that the quantization process did not impair the model's performance compared to the floating-point version. The ten times average inference time on the MCU was recorded at 165 ms with a 0.31 ms standard deviation. For the RF model, since it was executed as a floating-point model, its PC-based results were consistent with the MCU inference results, also achieving 99.03% accuracy. According to the Arduino IDE, the RF model utilized 45.88 KB of RAM and 111 KB of flash memory. However, its ten times average inference time was only 0.46 ms with a 0.01 ms standard deviation, which is significantly faster than that of the CNN model.

Based on the average inference times of 165 ms and 0.46 ms, the energy required for a single inference, with a clock frequency of 64 MHz and MCU active current consumption of $52\,\mu\mathrm{W\,MHz^{-1}}$, is calculated to be $0.548\,\mathrm{mJ}$ and $0.0015\,\mathrm{mJ}$, respectively. Assuming the use of a coin cell battery with a nominal capacity of 250 mAh and an output voltage of 3 V, the battery can theoretically provide a total energy of 2,700 J. This energy is sufficient to support the MCU in performing 4,927 cycles of CNN inference or 1,800,000 cycles of RF inference.

In the context of AE-based carbon fiber plate damage identification, both RF and CNN models are effective. However, determining which method is superior is challenging. On one hand, RF offers significant advantages due to its fast inference speed and low energy consumption, making it ideal for time-sensitive and energy-limited applications. On the other hand, while CNN inference is slower, it outperforms RF in low SNR environments, and quantized CNNs can be more easily deployed on lower-cost MCUs due to their reduced RAM and flash memory requirements. The choice of approach depends on the specific needs and trade-offs of the particular industrial sector.

V. Conclusions and Future Work

This paper investigated the application of ML algorithms and CNNs for AE-based non-destructive fault diagnosis in carbon fiber panels. Thirteen common ML algorithms were evaluated using an automated pipeline, with RF emerging as the most suitable algorithm for this task. Addressing the challenges of limited AE data, the study further explored data augmentation techniques, including traditional time-series methods and the deep learning-based DCGAN approach. The results demonstrated that DCGAN significantly enhanced model accuracy and robustness, outperforming traditional augmentation methods.

Subsequently, RF and CNN models were deployed on a low-power resource-constrained MCU using TinyML to evaluate their on-device performance. Both models achieved an impressive accuracy of 99.03%, while maintaining low energy consumption per inference: $0.548\,\mathrm{mJ}$ for the CNN and $0.0015\,\mathrm{mJ}$ for the RF. Memory usage was similarly efficient, with the CNN requiring 31.1 KB of RAM and 67.8 KB of flash memory, and the RF using 45.88 KB of RAM and 111 KB of flash memory. These results highlight the feasibility of deploying ML models on low-power, resource-constrained devices for real-time SHM.

For AE-based non-destructive fault diagnosis in carbon fiber composite panels, the integration of ML algorithms and DCGAN-based data augmentation techniques offers a robust and efficient solution. Deploying these models on low-cost embedded systems enables continuous monitoring of structural damage and provides timely alerts for repair or replacement, supporting the proactive maintenance of critical engineering structures. Achieving long-term, autonomous operation of these systems will require the use of low-power TinyML models powered by batteries or energy harvesting technologies. This study represents an important step toward this vision, demonstrating the potential for scalable and cost-effective SHM solutions.

While the findings are promising, several avenues remain for future research. First, a complete SHM solution would require the integration of AE signal acquisition and real-time communication of recognition results, which may increase the overall system power consumption. Future work should explore techniques to optimize power usage in such integrated systems. Secondly, this study only considers the impact of different data augmentation techniques on the robustness and accuracy of CNN and RF algorithms. Future research should explore their effects on all common ML algorithms to provide a more comprehensive understanding of AE signal noise resistance. Additionally, future studies should investigate the applicability of the proposed approach to other composite materials, as well as examine the potential impact of domain shift and data drift on model performance. These efforts will further refine the deployment of efficient, reliable, and scalable SHM solutions.

Acknowledgment

This research was financially supported by the Knowledge Foundation under grant NIIT 20180170 and 20240029-H-02 (TransTech2Horizon).

References

[1] M. Norouzi and N. Masoumi, "Crasen: A phase variation passive sensor node for metallic structural health monitoring," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–11, 2023.

[2] P. Nagulapally, M. Shamsuddoha, G. Rajan, M. Mohan, and B. G. Prusty, "Distributed fiber optic sensor-based strain monitoring of a riveted bridge joint under fatigue loading," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–10, 2021.

[3] D. Crivelli, M. Guagliano, M. Eaton, M. Pearson, S. Al-Jumaili, K. Holford, and R. Pullin, "Localisation and identification of fatigue matrix cracking and delamination in a carbon fibre panel by acoustic emission," *Composites Part B: Engineering*, vol. 74, pp. 1–12, Jun. 2015.

[4] S. Song, X. Zhang, Y. Chang, and Y. Shen, "An improved structural health monitoring method utilizing sparse representation for acoustic emission signals in rails," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–11, 2023.

[5] X. Li, H. Yan, K. Cui, Z. Li, R. Liu, G. Lu, K. C. Hsieh, X. Liu, and C. Hon, "A novel hybrid yolo approach for precise paper defect detection with a dual-layer template and an attention mechanism," *IEEE Sensors Journal*, vol. 24, no. 7, pp. 11 651–11 669, 2024.

[6] Y. Zhang, L. S. Martinez-Rau, B. Oelmann, and S. Bader, "Enabling autonomous structural inspections with tiny machine learning on uavs," in *2024 IEEE Sensors Applications Symposium (SAS)*, 2024, pp. 1–6.

[7] J. Cao, S. Tian, Z. Yang, G. Teng, H. Li, R. Jin, R. Yan, and X. Chen, "Blade tip timing signal filtering method based on sampling aliasing frequency map," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–12, 2022.

[8] M. Chai, Z. Zhang, and Q. Duan, "A new qualitative acoustic emission parameter based on Shannon's entropy for damage monitoring," *Mechanical Systems and Signal Processing*, vol. 100, pp. 617–629, Feb. 2018.

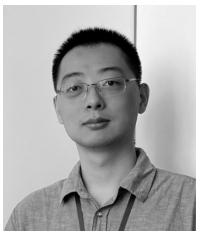
[9] R. Yan, Z. Shang, H. Xu, J. Wen, Z. Zhao, X. Chen, and R. X. Gao, "Wavelet transform for rotary machine fault diagnosis:10 years revisited," *Mechanical Systems and Signal Processing*, vol. 200, p. 110545, Oct. 2023.

[10] Y. Qiao, H. Wang, J. Cao, and Y. Lei, "Sound-vibration spectrogram fusion method for diagnosis of rv reducers in industrial robots," *Mechanical Systems and Signal Processing*, vol. 214, p. 111411, 2024.

[11] X. Li, Y. Lin, W. He, R. Liu, A. L. Oliveira, T. Qian, J. Zheng, and C. Hon, "Enhancing the interpretation of spirometry: Joint utilization of n-order adaptive fourier decomposition and deep learning techniques," *IEEE Transactions on Instrumentation and Measurement*, vol. 74, pp. 1–14, 2025.

[12] I. Hoyer, O. Berg, L. Krupp, A. Utz, C. Wiede, and K. Seidl, "Hardware accelerators for a convolutional neural network in condition monitoring of cnc machines," in *2023 IEEE SENSORS*, 2023, pp. 1–4.

[13] B. An, S. Wang, Z. Zhao, F. Qin, R. Yan, and X. Chen, "Interpretable neural network via algorithm unrolling for mechanical fault diagnosis," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–11, 2022.
[14] Y. Zhang, S. Bader, and B. Oelmann, "A Lightweight Convolutional Neural Network Model for Concrete Damage Classification using Acoustic Emissions," in *2022 IEEE Sensors Applications Symposium (SAS)*, Aug. 2022, pp. 1–6.
[15] J. Liu, Y. Xu, M. Cao, F. Gao, J. He, and J. Lin, "Fatigue crack size evaluation using acoustic emission signals for wire and arc additive manufactured material," *Mechanical Systems and Signal Processing*, vol. 204, p. 110819, Dec. 2023.
[16] S. Ju, D. Li, and J. Jia, "Machine-learning-based methods for crack classification using acoustic emission technique," *Mechanical Systems and Signal Processing*, vol. 178, p. 109253, Oct. 2022.
[17] S. Sengupta, A. K. Datta, and P. Topdar, "Structural damage localisation by acoustic emission technique: A state of the art review," *Latin American Journal of Solids and Structures*, vol. 12, no. 8, pp. 1565–1582, Aug. 2015.
[18] Y. Liu, M. Huo, Q. Li, H. Zhao, Y. Xue, J. Yang, and N. Qi, "Imbalanced source-free adaptation diagnosis for rotating machinery," *IEEE Transactions on Instrumentation and Measurement*, vol. 73, pp. 1–11, 2024.
[19] F. Zakaryapour Sayyad, I. Shallari, S. J. Mousavirad, and M. O'Nils, "Model Evaluation and Selection for Robust and Efficient Advertisement Detection in Print Media," in *Advances in Computing and Data Sciences*. Cham: Springer Nature Switzerland, 2025, pp. 211–224.
[20] X. Li, W. Zhang, Q. Ding, and J.-Q. Sun, "Intelligent rotating machinery fault diagnosis based on deep learning using data augmentation," *Journal of Intelligent Manufacturing*, vol. 31, no. 2, pp. 433–452, Feb. 2020.
[21] T. T. Um, F. M. J. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić, "Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks," in *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, ser. ICMI '17. New York, NY, USA: Association for Computing Machinery, Nov. 2017, pp. 216–220.
[22] W. Fu, R. Zhou, and Z. Guo, "Concrete acoustic emission signal augmentation method based on generative adversarial networks," *Measurement*, vol. 231, p. 114574, 2024.
[23] U. Muthumala, Y. Zhang, L. S. Martinez-Rau, and S. Bader, "Comparison of tiny machine learning techniques for embedded acoustic emission analysis," in *2024 IEEE 10th World Forum on Internet of Things (WF-IoT)*, 2024, pp. 444–449.
[24] Y. Zhang, V. Adin, S. Bader, and B. Oelmann, "Leveraging Acoustic Emission and Machine Learning for Concrete Materials Damage Classification on Embedded Devices," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–8, 2023.
[25] Y. Zhang, L. S. Martinez-Rau, Q. N. P. Vu, B. Oelmann, and S. Bader, "Survey of quantization techniques for on-device vision-based crack detection," *arXiv preprint arXiv:2502.02269*, 2025.
[26] V. Adın, Y. Zhang, B. Oelmann, and S. Bader, "Tiny Machine Learning for Damage Classification in Concrete Using Acoustic Emission Signals," in *2023 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, May 2023, pp. 1–6.
[27] L. S. Martinez-Rau, Y. Zhang, B. Oelmann, and S. Bader, "Tinyml anomaly detection for industrial machines with periodic duty cycles," in *2024 IEEE Sensors Applications Symposium (SAS)*, 2024, pp. 1–6.
[28] C. U. Grosse, H. W. Reinhardt, M. Motz, and B. Kroplin, "Signal conditioning in acoustic emission analysis using wavelets," *NDT. net*, vol. 7, no. 9, pp. 1–9, 2002.
[29] R. Finlayson, "Acoustic emission testing," *Handbook of Nondestructive Evaluation, C. Hellier, Ed. New York: McGraw-Hill*, pp. 10–1, 2003.



Yuxuan Zhang (Graduate Student Member, IEEE) received the MSc degree in Embedded Systems Engineering from the University of Leeds, UK in 2019. Since 2021, he is a Ph.D. candidate of Electronics at the Department of Computer and Electrical Engineering, Mid Sweden University, Sweden. His research interests include machine learning & signal processing on low-power resource-constrained embedded devices and on-device learning.

Rhys Pullin was born in Newport, Wales in 1975. He received the Ph.D. degree from Cardiff University in 2001. He is currently a Professor and the Head of the Department of Mechanical and Medical Engineering at Cardiff University working in the area of nondestructive evaluation. His primary area of research is structural health monitoring using acoustic emission and acousto-ultrasonics in metallic and composite aerospace components. Dr. Pullin has completed research programs for Airbus, Boeing, EDAS, the Ministry of defence, the Centre for Defence Enterprise, and SKF. Finally, He is a member of the National Committee for the British Society of Strain Measurement and the Health Management and Prognostics National Technical Committee.

Bengt Oelmann received the Doctor of Technology degree in electronics from the Royal Institute of Technology, Stockholm, Sweden, in 2000. He is currently a Full Professor in electronics system design with Mid Sweden University, Sundsvall, Sweden. His current research interests include low-energy embedded system design, energy harvesting, and embedded sensor technology.

Sebastian Bader (Senior Member, IEEE) received the Ph.D. degree in electronics from Mid Sweden University, Sundsvall, Sweden, in 2013, and the Dipl.-Ing. degree from the University of Applied Sciences, Wilhelmshaven, Germany. He is currently an Associate Professor of embedded systems with the Department of Electronics Design, Mid Sweden University. His research interests focus on energy aspects of embedded systems, including energy harvesting, low-power sensing systems, and machine learning on resource-constrained devices.