Causal dilated convolutional neural networks for automatic inspection of ultrasonic signals in Non-Destructive Evaluation and Structural Health Monitoring

Stefano Mariani[1], Quentin Rendu[1], Matteo Urbani[1,2] and Claudio Sbarufatti[2]

[1]Department of Mechanical Engineering, Imperial College, London SW7 2AZ, UK
[2]Politecnico di Milano, Dipartimento di Meccanica, Via La Masa 1, I-20156 Milano, Italy

Highlights

- This paper presents a deep learning network that inspects ultrasonic signals for defect detection.
- The network is an adaptation of WaveNet, hence is based on causal dilated convolutional neural networks and residual blocks.
- The network is shown to outperform both a widely used conventional analysis method (the optimal baseline selection + baseline signal stretch approach) and some competing deep learning algorithms (multilayer perceptron and recurrent neural networks with long short-term memory).
- The validation was performed on two datasets, one generated via finite element simulations on a steel plate, the other by actual experiments on a composite plate.
- Significant improvements over the conventional method are obtained especially when testing occurs at temperatures well outside the range in the training set of signals.

Abstract

This paper presents a deep learning network that performs automatic detection of defects by inspecting full ultrasonic guided wave signals excited in plate structures. The findings show that the algorithm, which is an adaptation of WaveNet, and hence is based on causal dilated convolutional neural networks, is effectively able to learn features and/or patterns related to the presence of waves scattered from damage, thus eliminating the need for any feature engineering to be performed by human operators. The network outperformed the widely used conventional approach that combines the optimal baseline selection and baseline signal stretch compensation methods when tested on two different datasets. The first dataset consisted of finite element simulated Lamb wave signals acquired in a pitch-catch configuration on a steel plate across a 50°C range of temperature variations, and the second was a publicly available experimental dataset of Lamb wave signals also acquired in pitch-catch mode on a composite plate with a 40°C range of variations. The improvements over the conventional approach are

particularly encouraging when analyzing signals at temperatures well outside the temperature range available in the set of baseline signals, hence suggesting that this class of algorithms can complement or substitute existing methods, especially when testing occurs at unseen environmental and operational conditions, or when the effects of sensor drift make conventional methods less effective.

Keywords: Ultrasound, guided waves, defect detection, deep learning, WaveNet, causal dilated CNN

1. Introduction

When ultrasonic testing is performed with guided waves, such as Lamb waves in plate-like structures [1], in a non-destructive evaluation (NDE) setting, signal interpretation is a challenging task due to the typical presence of multiple modes propagating in the test structure with different velocities and possibly interacting with each other. Reverberations of stress waves reflected at structural boundaries or from damage greatly add to the complexity of the problem. Recently, there has been a move toward structural health monitoring (SHM) configurations, where sensors are attached to the structure in a permanent installation; this has the advantage that later acquisitions can be compared to earlier ones, typically by simple subtraction of the latest 'current' measurement from an earlier one that is referred to as the 'baseline'. Ideally, the only change between the two signals is the possible formation of reflections from defects, which would then be clearly seen after baseline subtraction [2]. However, baseline subtraction suffers from the effects of environmental and operational conditions (EOCs), such as temperature, and those of sensor drift. The latter refers to a change of sensor behavior or of the mechanical properties of the adhesive layers used to bond sensors to the structure over time; as a consequence, the stress waves excited and received from the structure always differ from one measurement to another, hence producing non-zero baseline subtracted residuals across portions of the signal, or across the whole signal length. Even if extremely stable sensors and adhesives are employed, signals acquired at two different times are still likely to differ due to the presence of different EOCs effects. Among those, temperature effects have been the subject of several studies [3–6], and a number of temperature compensation techniques have been developed. Baseline signal stretch (BSS) uses one baseline signal as a reference to either compress or stretch in time any current measurement to minimize the residuals between the two [7,8], hence BSS fundamentally aims at compensating the effects of changing temperature on the wave speed and on thermal expansion of the test structure, which modify the propagation length [9]. However, since the different speeds of the various propagating modes are, in general, differently affected by a given change of temperature, and since other effects also combine to modify the received signals, such as those due to temperature-dependent sensor properties [9], the method is only partially effective. Optimal baseline selection (OBS) method, rather than only using one reference signal, stores a set of baselines so that the

one deemed most similar to the specific current measurement is used for baseline subtraction, often after applying BSS on the selected baseline (OBS+BSS) [7,10]. Temperature compensation in SHM is still a very active field of research, and other recently developed data-driven techniques include [6,11–15], all of which require the acquisition of either one or a set of baseline signals that form a reference for the compensation process. However, any of these methods suffers to various extents when unseen changes occur, such as those due to testing at temperatures significantly outside those experienced in the acquired set of baseline signals, to an uneven temperature profile across the spatial extent of the test structure, to the effects of other EOCs such as loads, or to the occurrence of sensor drift [16]. For these reasons, both the NDE and SHM communities can greatly benefit from the development of machine learning architectures that can be trained effectively to detect the reflection from defects independently from the given EOCs acting in the test structure and from the different sensor and/or adhesive behavior at the time of testing.

Since the 90s, researchers in the NDE field have investigated the use of multilayer perceptron (MLP) networks to process information extracted from ultrasonic signals acquired in test structures [17–21]. Also due to the limited hardware capabilities in early work, shallow MLP networks consisting of one [17–19,21] or two hidden layers [20] were the preferred choice, and the signals were typically pre-processed to reduce their dimensionality, i.e. by extracting features deemed to be sensitive to defect reflections either from the time-domain waveforms [21], or after operating FFT [17] or Wavelet transforms [18–20]. However, applications of MLP networks, or of other forms of artificial neural networks (ANN), to perform tasks of NDE have not found widespread application, partially due to the need for skillful operators to manually determine defect-sensitive features effective for each specific application.

In the last decade, with the advent of GPU-based machines and the increasing availability of large training datasets, there has been a proliferation of deep learning architectures, and some of these techniques have set the state-of-the-art to tackle a number of different applications in various fields, such as in computer vision, medical image analysis, speech recognition and natural language processing. Alongside deep MLP networks, two popular alternatives are recurrent neural networks (RNN) [22], especially in the improved forms of long short-term memory (LSTM) [23] and gated recurrent units (GRUs) [24], and convolutional neural networks (CNN) [25]. Traditionally, RNNs have been the method of choice for sequence modelling tasks [26], i.e. to interpret, make predictions or generate any type of sequential data, such as audio, although recently one-dimensional, ‘causal dilated’ CNNs such as WaveNet [27] and ByteNet [28] have been developed and have been shown to reach similar (or better) accuracies as those offered by RNNs in several applications, at the same time offering a number of other advantages, as described in section 5.1.

These recent successes resulted in a renewed interest towards deep learning methods from the NDE and SHM communities, including for ultrasound-based applications. Potentially, some of these algorithms can provide the modelling capabilities needed to interpret the complex ultrasonic wave propagation of guided waves in structures when taking the entire received waveform as input, hence eliminating the need for researchers or operators to manually select features. Two recent works have focused on cases of acoustic emission (AE), where stress waves are produced by crack growth events, hence requiring continuous monitoring [29]. Ebrahimkhanlou et al. [30] instrumented an aluminum panel including features such as doublers and rivet connections with a single AE transducer, and developed a deep learning algorithm using autoencoders aimed at detecting and localizing stress waves manually excited via Hsu-Nielsen pencil lead break tests. The main idea was to leverage the information contained in multiple reverberations of the stress waves reflected at geometrical boundaries and other structural features. Similarly, Hesser et al. [31] have considered the case of AEs produced by a steel ball dropping on an aluminum plate, and have used a network of MLPs to localize the hit location. Melville et al. [32] have tackled the case of Lamb waves excited in an aluminum plate by means of a piezoelectric actuator, although they used a laser Doppler vibrometer to acquire images of the full wavefield, then used a CNN network to inspect them as typically done in the computer vision field.

1.1. Scope of the paper

To the authors knowledge, no attempts have been reported to develop a machine learning architecture able to interpret raw guided wave signals in applications involving sensors both emitting and receiving A-scan measurements, hence being arranged either in a pulse-echo or pitch-catch configuration. Given the similarities existing between ultrasonic waveforms and audio signals, it seems reasonable that deep learning algorithms successfully used for audio sequence modelling can be adapted for tasks of defect detection and, possibly, localization based on interpretation of ultrasonic measurements. To this end, this article investigates the use of LSTM networks and, particularly, causal CNNs of the type presented in [27] for the binary classification task of defect detection, i.e. to flag presence or absence of defect reflections in a measured A-scan. To establish whether using these classes of architecture effectively offer improvements over MLP networks, the latter have also been investigated. As in the work of Ebrahimkhanlou et al. [30], one of the assumptions of the study is that the accuracy of defect detection can benefit from the use of signals acquired for a sufficiently long duration to include some stress wave reflections from structural boundaries, since this can give the opportunity to a trained algorithm to leverage additional information, i.e. to examine additional waves reflected from damage and reaching the receiver other than those travelling along the most direct path existing between the two. A second key

point of the research is that only minimum signal processing has to be applied to the signals before feeding them to the machine learning architectures. This only consists of the partial compensation of temperature effects via application of the BSS method [7,8], since that simply requires the manual selection of a reference signal and automatically stretches any other signal in an effort to minimize the temperature effects on the speed of wave propagation. In order to have full control of parameters such as number and position of sensors, testing temperature, type and position of defect, and noise affecting the measurements, a numerical dataset was generated via finite element (FE) simulations, which comprises simulated guided wave signals acquired from a steel plate. This dataset constitutes a valuable test-benchmark to investigate and compare the candidate machine learning architectures. Finally, the architecture deemed to be best suited to interpret ultrasonic data was tested on experimental signals acquired from a composite plate and made available in an open-source framework by Moll et al. [33].

Section 2 describes the numerical dataset and applies the conventional OBS+BSS method to characterize the difficulty of the problem. Section 3 discusses the MLP networks and applies them to the numerical dataset. RNNs based on LSTM are then described and tested on the numerical dataset in section 4. Section 5 introduces the concept of causal dilated CNNs and applies it to both the numerical and the experimental datasets. This is followed by the conclusions of the paper.

2. Generation of a numerical dataset to test the machine learning algorithms

2.1. Description of the finite element model

A 3D model of a 500x500x5 mm steel plate was created with Pogo [34], a FE package that performs explicit time-domain simulation of wave propagation problems in an efficient manner by using Compute Unified Device Architecture (CUDA) parallelization on graphics processing units (GPUs). The plate was discretized using 10 million linear brick elements with 0.5 mm-long edges. Steel was modelled as an isotropic non-attenuative material whose density, elastic modulus and Poisson's ratio were set based on literature values and were varied in different simulations in order to mimic testing at various temperatures [6]. Temperatures ranging from 20 to 69°C at steps of 1°C were considered, and the cited material properties across this range were varied using linear relations with temperature, the values at the extreme temperatures being provided in Table 1. Figure 1(a) shows the dispersion curves of the Lamb wave modes in terms of group velocity for the modelled 5 mm-thick plate at 20°C, those at higher temperatures being similar. By inspection of Figure 1(a), it was decided to conduct the simulated testing using the fundamental A0 flexural mode at 300 kHz, for which limited dispersion is expected due to the derivative of its group velocity being close to zero at the chosen frequency, and that frequency being just below the

A1 mode cutoff. A similar choice would be commonly made in practice, since it provides a good balance between the desirability of increasing the frequency, hence enhancing the defect sensitivity, and reducing the complexity of the signal analysis by limiting the number of modes excited in the test structure. It is possible that well-designed ML algorithms might actually take advantage of the information contained in multiple propagating modes that would be generated by choosing a higher excitation frequency, but this was not investigated here. The advantage of the A0 mode compared to the extensional S0 mode is that the former propagates axisymmetrically from an excitation point, hence providing better omnidirectional coverage of the plate structure. An actuator was simulated by selecting the node in the middle of the upper face of the plate (i.e. the face lying in the positive, outward z-direction considering the Cartesian coordinate system shown in Figure 1(b)) and by assigning to it a force time-history with the shape of a 5-cycle, Hanning windowed toneburst at 300 kHz; the force was directed normally to the plate, hence predominantly exciting the flexural A0 mode and only to a much lesser extent the extensional S0 mode. Similarly, a sensor was modelled by selecting a node at the desired location on the same face of the plate and by recording the displacement time-history experienced by the node in the normal direction to the plate at each simulation step. Overall, this provides a reasonable simulation of the stress waves that would be excited and sensed by e.g. small piezoelectric discs attached to the plate. A more rigorous characterization of piezo sensors would also be possible in FE [35], but the extra sophistication was deemed unnecessary for the purpose of these simulations.

It has been shown experimentally that the signals excited and received by piezoelectric sensors subjected to different temperatures typically undergo temperature-dependent changes such as variations in amplitude and in the frequency spectrum of the excited toneburst, the most notable effect being a significant phase change [6]. Amplitude changes can typically be dealt with by normalizing each measurement to the maximum amplitude of the direct arrival (in pitch-catch mode) or of the reflection from a known structural boundary (in pulse-echo mode), though this can be difficult if signals overlap or multiple modes are present, and can cause errors if a defect grows in the direct arrival path. However, phase changes make signal analysis much more challenging since they modify the shape of the received signals, hence also worsening the results obtained via BSS temperature compensation and baseline subtraction [6]. This adverse effect was included in the FE simulations by linearly varying the phase of the excited toneburst by a total of 100° in the 50°C temperature range, this being consistent with previous experimental findings [6].

Table 1. Material properties for the finite element models of the 5-mm thick steel plate at the extreme temperatures of the considered range. Group velocity and wavelength for the fundamental A0 and S0 modes at 300 kHz are also given.

T (°C)	Density (kg/m3)	E (GPa)	Poisson's ratio (-)	A0 at 300 kHz		S0 at 300 kHz	
				Gr. vel. (m/s)	λ (mm)	Gr. vel. (m/s)	λ (mm)
20	7,908	219	0.2892	3,297	11	4,936	16.5
69	7,895	215.5	0.2905	3,272	10.9	4,843	16.1

The mesh size of 0.5 mm and the simulation time-step of 50 ns were chosen in order to satisfy the accuracy and stability requirements typical of elastodynamic simulations via FE at the chosen excitation frequency of 300 kHz. For accuracy, it is recommended to use at least 10 elements per shortest propagating wavelength [36]. The lowest phase velocity propagating in the model, and hence the shortest wavelength, is that of the A0 mode at 69°C, namely 2,606 m/s for a wavelength of 8.7 mm. Considering the most unfavorable case of propagation along the diagonal of a cubic element, the accuracy requirement was met by offering more than 10 elements per shortest wavelength. For stability, the Courant-Friedrichs-Lewy (CFL) condition dictates that the fastest propagating wave, this being the bulk longitudinal wave at about 6 km/s, must not travel more than one element in a single time-step [37]; by setting a simulation step of 50 ns the CFL requirement was also met.

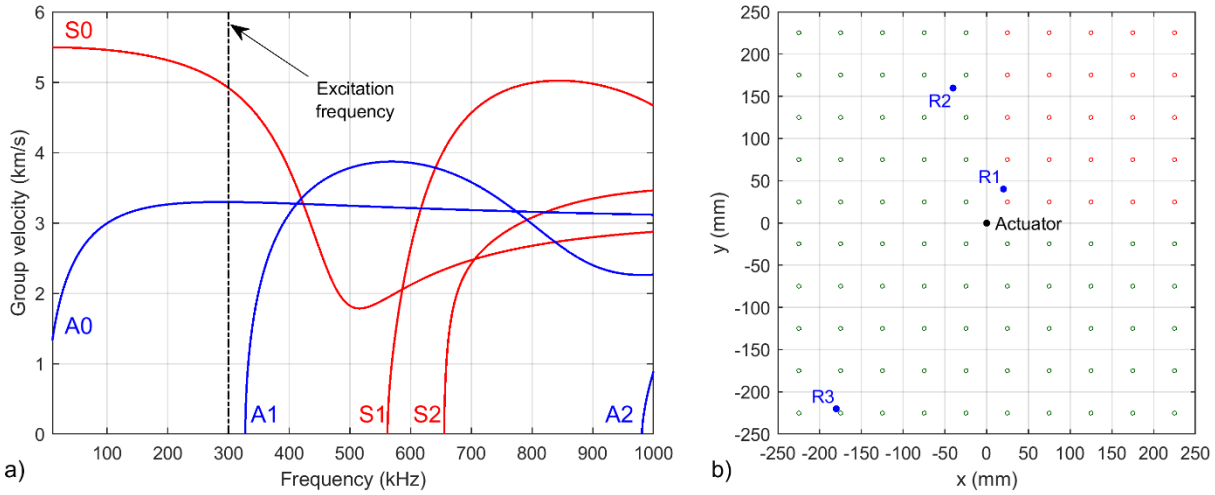

Figure 1. (a) Lamb wave group velocity dispersion curves for the modelled 5 mm-thick steel plate at 20°C. (b) Schematic of the simulated plates and tests; actuator and receivers (monitoring nodes) used in this study are indicated with black and blue dots, respectively. The positions of the 100 defects are shown in red and green circles; the 25 red circles correspond to 25 actual models that have been solved, each model including a single defect; simulated measurements acquired by the three receivers when a defect was introduced at one of the 75 green circles were obtained in post-processing by using 'mirrored' receivers around x- and y-axes.

To create a damaged plate, a through thickness hole was introduced at the desired nodal location by removing elements whose mid-point lay within a circle centered at the chosen nodal point and having 4.6 mm diameter (~40% of the A0 wavelength given in Table 1). This created a hole with staircase-shaped edges of equivalent area to a smooth circular hole of the same diameter. Note that the goal of the

simulations was to create a benchmark dataset to test different ML architectures designed to detect the presence of a reflector, rather than to determine the smallest defect that a given architecture would be able to detect, hence this damage size and shape was deemed satisfactory; in actual experimental settings, the difficulty of detection will mostly depend on the available signal-to-noise ratio (SNR) of the measurements. In order to test the ML algorithms at increasing levels of difficulty, three different levels of noise were added in post-processing by superposition to the simulation results, and this is described in the next section.

A total of 25 damaged models were generated as the through thickness defect was placed at 25 different locations in the quadrant defined by positive x- and y-coordinates in Figure 1(b); as seen in the figure, the locations were defined by a grid where the defect closest to the actuator was at position (25,25) mm and the furthest at (225,225) mm, and steps of 25 mm were used in both x- and y-directions. The case of pitch-catch testing using one actuator and one receiver was investigated in this study; since the relative positions of actuator, defect and receiver are expected to have an influence on the accuracy of defect detection, three different cases have been considered with the monitoring node (i.e. the simulated receiver) at different distances from the actuator, namely at locations (20,40), (-40,160) and (-180,-220) mm, as shown in Figure 1(b). The central position of the actuator and the choice of receivers not lying in either x- or y-axes enabled symmetry to be exploited to increase fourfold the available number of damage locations without having to run further simulations. This was achieved by inserting nine further monitoring nodes in the models, which were placed at all points of symmetry about the x- and y-axes with respect to the actual receiver locations (three symmetric monitoring nodes for each simulated receiver). The signals received by the mirrored monitoring nodes were then reordered to create three datasets, one for each simulated receiver, comprising simulated measurements from 100 different damaged plates, where the extra 75 damage locations were obtained by mirroring the results of the actual 25 models. Therefore, a total of 1,300 simulations were performed (26 models, one of which being undamaged, at 50 temperatures). The duration of each simulation was set to 0.5 ms, a time during which the A0 mode propagates for ~1.6 m from the actuator position; this is sufficient for A0 to be reflected from any defect found in the various models and for the defect reflection to reach either of the three chosen receivers, typically also including multiple reflections from the plate boundaries. By setting the sampling rate at 4 MHz, a single waveform was composed of 2,000 samples. Each simulation required about 10 minutes of processing on one of the NVIDIA® Quadro RTX™ 6000 GPU cards offered by the Imperial College Research Computing Service [38].

2.2. Post-processing of FE results: application of BSS method and introduction of noise

The BSS method [8] was applied to the signals acquired by each of the three considered receivers using the specific time-trace acquired in the undamaged plate at 45°C as baseline. Figure 2(a-c) plot the two signals recorded by receiver 2 (labelled 'R2' in Figure 1(b)) in the undamaged modelled plate at 45 and 69°C, the latter signal being time-stretched with a factor of 1.0042 after the application of BSS; this is higher than the actual ratio between the A0 mode velocities at the two temperatures, namely 1.0037, or that between S0 mode velocities (1.0031), suggesting that the computed stretch factor is significantly affected by the phase change between the signals, as explained below. The signals in the figure are normalized to the maximum envelope of the direct arrival of the A0 mode, seen roughly between 50 and 65 μs, while the direct arrival of the weaker S0 mode is at ~35 μs. The wave packets seen after these two direct arrivals are due to successive reflections from the plate boundaries. Perfect alignment of the RF wave components across the entire signal length is not possible, both due to the different phases of the excited toneburst and, to a lesser extent, to the different ratios between the speeds of the two modes (this was tested by performing an additional simulation at 69°C using an unmodified excitation phase from the 45°C case, in which case BSS returned a more accurate stretch factor of 1.0035, which is intermediate between the two ratios of velocities relative to A0 and S0 modes). The difference between the two signals shown in Figure 2(d) has peak values higher than 50%, indicating that it will be very difficult to detect damage growth reliably. More advanced methods of temperature compensation that combine the stretching procedure with capabilities of phase compensation, such as [6], are expected to yield better alignment throughout, and hence lower residuals after subtraction.

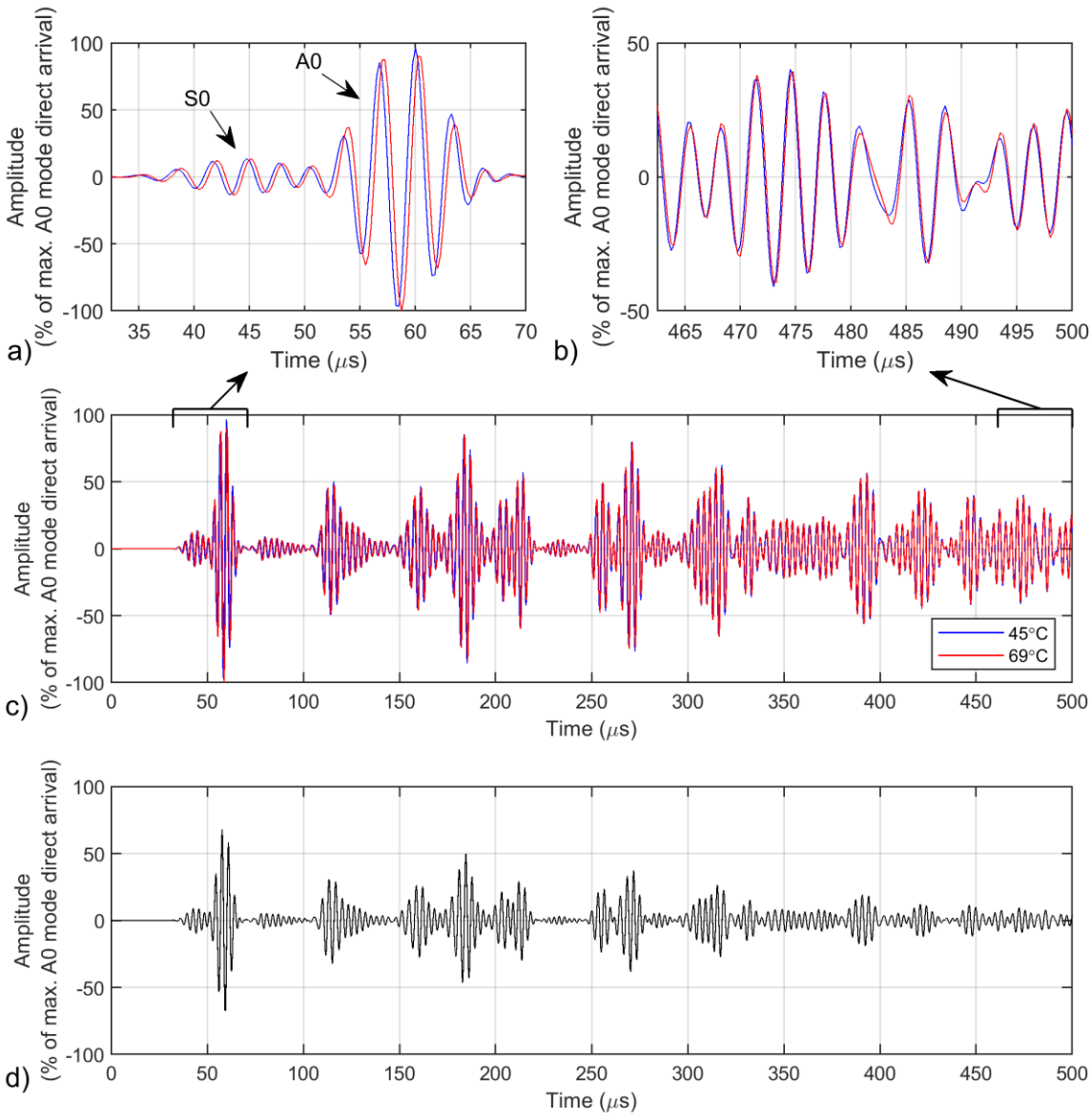

Figure 2. (a-c) Signals recorded by receiver 2 in the undamaged modelled plate at the baseline 45°C and at 69°C after BSS compensation; (a) and (b) are zoomed toward the initial and the final parts of the signals, respectively, which are shown in their entirety in (c). (d) Results of baseline subtraction.

Incoherent noise was superposed to the FE simulated measurements to mimic the effects of instrumentation noise, using a procedure similar to that employed in [15], which is described briefly in the following. In typical ultrasonic measurements, frequency components outside a band around the center frequency of the excited signal can be easily filtered out via hardware or software filters, while noise at and around the frequency of excitation is much more problematic. Therefore, noise time-traces were produced by generating a normally distributed random sequence of the same length as each individual signal, then band-pass filtering the sequence via a Butterworth filter of order 4 and cutoff frequencies of

230 and 370 kHz, and finally multiplying the filtered sequence by a desired factor before superposition to an individual measurement. The noisy traces were added to the BSS-compensated signals, hence saving computational time by only performing BSS once, rather than after every different realization of noise, although the latter approach would be more realistic. However, since the RF amplitudes of the coherent signals are much larger than those due to noise at realistic SNR levels, the chosen approach made negligible difference to the computed stretch factors. Three noise multiplying factors were considered, labelled SNR1 to SNR3 with decreasing SNR values, where the factor for the most favorable case (SNR1) was set to be at a level where conventional compensation still offers perfect accuracy, but is about to become problematic; this is discussed further in section 2.4. The multiplying factors used for SNR2 and SNR3 were then set as twice and four times that used in SNR1, respectively. Note that in practical settings the SNR can be improved e.g. by taking more averages of any given acquisition, although that comes at a cost of more energy needed to carry out testing, this being particularly important when the acquisition system is battery-powered, especially given that the SNR only improves with the square root of the number of averages. Furthermore, averaging is only possible in stationary conditions, which becomes increasingly questionable as a large number of averages is used.

To show the effects of noise at the chosen SNR values, Figure 3(a) plots the differences between the blue signal of Figure 2(a) and that recorded by the same receiver from the model with damage at the position (75,-125) mm at the same temperature of 45°C. As expected, the noise-free signals show some differences due to the interaction of the propagating modes with the through hole; for this defect, the maximum amplitude of residuals is of the order of 10%. After introduction of noise at the SNR1 level, the residuals in Figure 3(b) are visibly noisier, and, for example, before 100 μs a noise packet with amplitude close to 10% and unrelated to any damage reflection is seen. The adverse effects of the simulated instrumentation noise are more pronounced when the higher SNR2 and SNR3 levels are considered, as shown in Figure 3(c,d), respectively, and the correct detection of defect reflections in these cases is evidently more challenging. Section 2.4 quantitatively analyzes the accuracy of defect detection that would be achieved by using the conventional OBS+BSS temperature compensation method when considering all possible defect locations, temperatures and receivers.

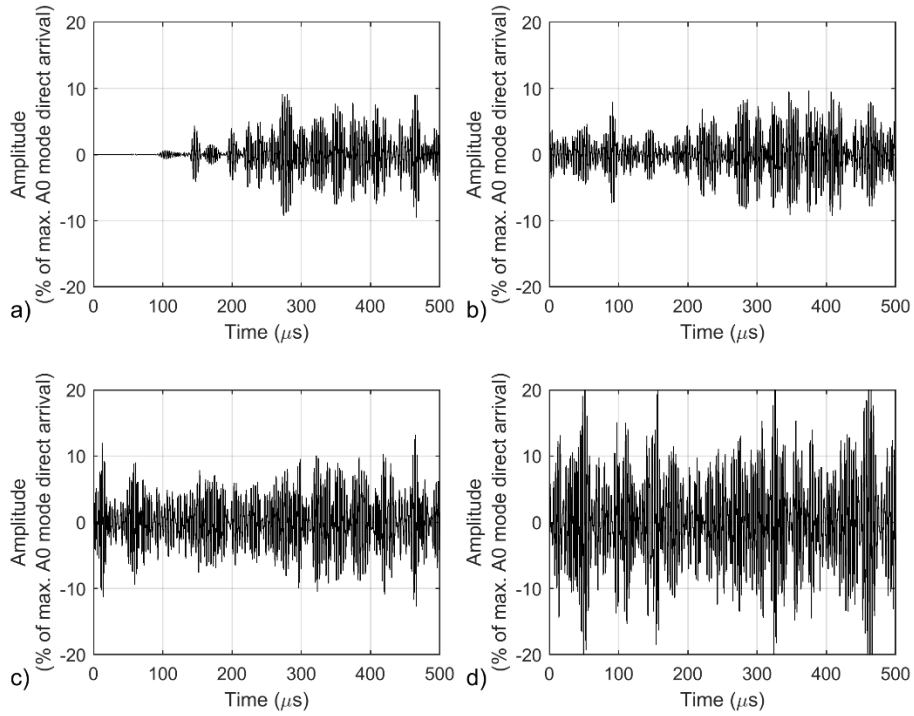

Figure 3. Differences between the signals recorded by receiver 2 at 45°C in the undamaged modelled plate and in the model including the defect located at (75,-125) mm. (a) noiseless simulation; (b-d) noise added at SNR1, SNR2 and SNR3 levels, respectively.

2.3. Dataset split - training, validation and testing sets

To train a supervised ML algorithm, the available labelled dataset is typically split into three portions referred to as training, validation and testing sets. The training set is the data effectively used to update the parameters of a given architecture during the training phase, e.g. to update weights and biases in the case of a MLP network, and the 'epochs' indicate the number of passes completed by the ML algorithm on the entire training dataset, as further discussed in section 3.1. At the end of each epoch, the accuracy of the current ML model is tested on the validation set. When the training is fully concluded, the model parameters that gave the best performance in terms of a desired parameter on the validation set are retained, and the corresponding trained version of the ML model is finally tested on the testing set, hence providing a true unbiased evaluation of a given model. Therefore, the accuracy on the testing set was the feature used in this study to compare the performance of the competing models.

The dataset was split into training, validation and testing sets using a 80/10/10% ratio, and Figure 4 illustrates the division employed for each of the three receivers considered here; this dataset division will be referred to as scenario 1. Twenty defects were randomly chosen, half of which were assigned to the

validation set and the other half to the testing set, as seen in Figure 4(a); all simulation results from any given defect (50, one from each modelled temperature) were assigned to either the training, validation or testing set, as indicated in Figure 4(b). This ensures that at testing time the ML model has never analyzed signals reflected by any of the ten red defects shown in Figure 4(a). The 50 simulations on the undamaged modelled plate were assigned to the three sets under the scheme shown in the bottom row of Figure 4(b). Therefore, the undamaged training set is formed by measurements taken every 2°C from 20 to 68°C, for a total of 25 signals, while the validation and testing sets have 13 and 12 of such signals, respectively.

There is a significant mismatch between the number of simulations from the plate in damaged conditions versus those from the undamaged plate. Following a procedure that is frequently employed to augment a dataset for ML training, multiple copies of the undamaged signals were used to establish a 50/50% division between damaged and undamaged cases in each dataset. Note that the copies of a measurement at any specific temperature are not exact replicas since a different realization of noise was added to each signal via the procedure described in the previous section. The number of copies and the total number of signals available in each set and for each analyzed receiver are summarized in Table 2.

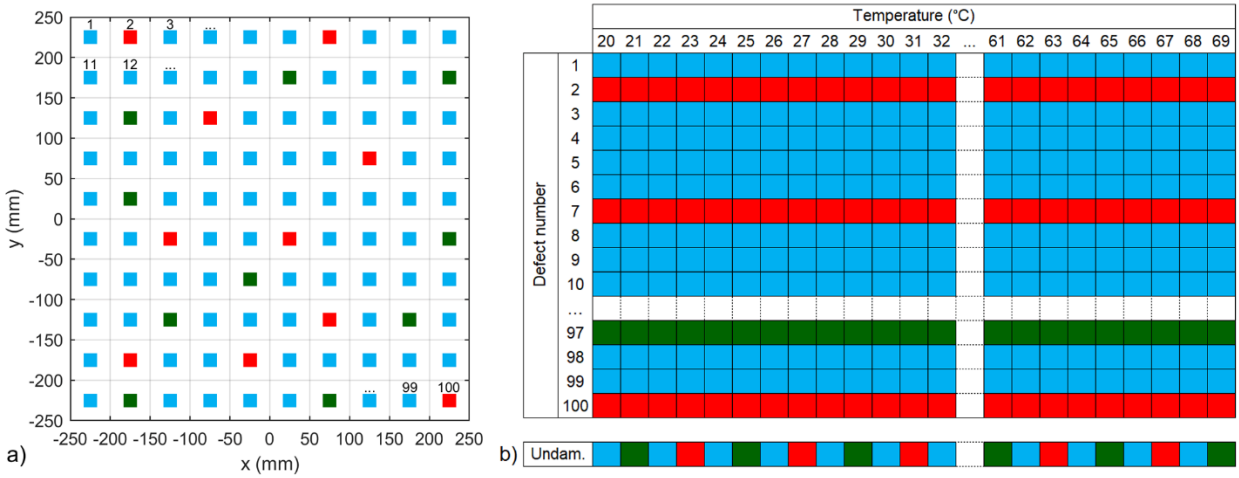

Figure 4. Scenario 1: division of dataset into training (cyan), validation (green) and testing (red) sets. (a) Plate schematic showing the defects assigned to training, validation and testing sets. (b) For the models with added defect, all simulated measurements at any temperature are assigned to the same set; for the undamaged modelled plate the measurements at different temperatures are assigned to the three sets using the scheme shown in the bottom row.

The dataset division described in Figure 4 was used to evaluate and compare the performance offered by the three tested ML architectures, as described in the next sections. The architecture deemed most capable of interpreting ultrasonic measurements was further trained and tested on three different subsets formed by only including measurements taken between 20 and 39°C in both training and validation sets, while simulations at higher temperatures were only assigned to the testing set. In this case, the BSS compensation described in the previous section was recomputed using the undamaged signals at 30°C as

baselines, rather than those at 45°C, since that is outside the temperature range of training data and might favor the ML algorithms when evaluated on the testing sets. The three further scenarios (defined as scenarios 2 to 4) consider testing sets comprising signals acquired at (1) 40 to 49°C, (2) 50 to 59°C and (3) 60 to 69°C, hence up to 10, 20 and 30°C outside the training range, respectively. For each scenario, the dataset was split in a 85/10/5% ratio, as detailed in Table 2, and the scheme used for each of the three sub-datasets is shown in Figure 5, which refers to the most challenging case (scenario 4). The same twenty defects as before were assigned to validation and testing sets; measurements of training and validation defects above 39°C were discarded. Similarly, signals from defects belonging to the testing set but acquired outside the considered 60 to 69°C range were neglected. Undamaged signals from 22 to 37°C at steps of 3°C were given to the validation set, the remaining ones in the range 20 to 39°C to the training set, while the ten undamaged cases in the temperature range of testing were assigned to the testing set. The dataset division used for scenarios 2 and 3 is analogous.

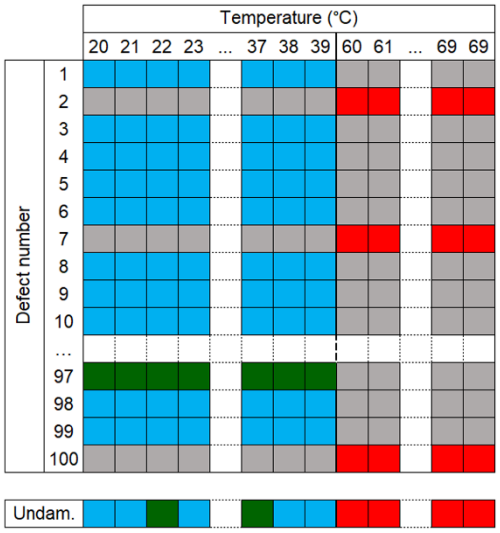

Figure 5. Scenario 4: subset of signals used to evaluate the ML performance when the testing set includes measurements up to 30°C outside the training set. Colors as in Figure 4; discarded signals are in grey.

Table 2. Number of signals used in training, validation and testing sets for the four considered scenarios.

	Entire finite element dataset (scenario 1)			Sub-datasets to test accuracy at temperatures outside the training range (scenarios 2 to 4)		
	Undamaged + damaged (% of total signals in the three sets)	Undamaged (number of unique FE models x number of copies)	Damaged	Undamaged + damaged (% of total signals in the three sets)	Undamaged (number of unique FE models x number of copies)	Damaged
Training	8,000 (80%)	4,000 (25 x 160)	4000	3,196 (84.2%)	1,596 (14 x 114)	1600
Validation	994 (10%)	494 (13 x 38)	500	398 (10.5%)	198 (6 x 33)	200
Testing	1,004 (10%)	504 (12 x 42)	500	200 (5.3%)	100 (10 x 10)	100

2.4. Signal analysis using the OBS+BSS method

This section investigates the performance given by the OBS temperature compensation method [7,10] that is a commonly used technique for ultrasonic signals, when applied to the scenarios described in the previous sections. In OBS, a criterion needs to be chosen to characterize the similarity between a given testing measurement and any signal in the baseline dataset, thus selecting the best match. A sensible criterion is to minimize the mean square deviation [7], which was adopted in this study. For each current (testing) measurement, once the optimal baseline was selected, that was used as reference to also apply the BSS method (i.e. employing an OBS+BSS approach [7,10]). The stretched current measurement was then subtracted from the specific baseline, and the parameter monitored for defect detection was the root mean square (RMS) of the residual signal.

Although care was taken to adhere to similar conditions as those existing for ML training, when drawing a comparison between the results given by the OBS+BSS approach with those obtained from the tested ML architectures, one needs to consider that the former does not make any use of the information contained in the damaged signals. Nevertheless, the results yield by OBS+BSS give a clear indication of the difficulty in detecting the defects included in the testing sets in the various scenarios and at the considered SNR values. For better comparison with scenario 1 (dataset split of Figure 4), the set of baselines for OBS was formed by including undamaged signals at all temperatures except those of the testing set. 100 copies of those 38 signals were produced and different realizations of noise were superposed as described in section 2.2, yielding a total of 3,800 signals in the baseline set (close to the 4,000 undamaged signals in the ML training set as seen in Table 2). The testing set for OBS was formed in the same manner as that for ML training. To analyze scenarios 2 to 4 shown in Figure 5, 80 copies of the undamaged signals of both the training and validation sets were used to form the OBS baseline set, hence obtaining 1,600 signals in the baseline set; again, the OBS testing set was left identical to that used for ML.

Receiver operating characteristic (ROC) curves were computed as the threshold for defect calling was increased; at each threshold level, the number of residual signals whose RMS value exceeded the given threshold when considering either damaged or undamaged sets of measurements was counted, and both probabilities of detection (PD) and of false alarms (PFA) were then estimated by dividing each of the two computed numbers by the population size of each set. Figure 6 shows the ROC curves obtained when using each of the three receivers of Figure 1(b) for some representative cases, namely scenario 1 at (a) SNR2 and (b) SNR3, and (c) scenario 3 at SNR1. Since the accuracy of detection of the ML architectures tested in the next sections will be given as percentage of overall accuracy, i.e. with no distinction between PD and PFA values, each plot also gives the corresponding indication referred to the point of the specific

ROC curve for which that measure of accuracy is maximized; finally, Table 3 summarizes the accuracies computed for any considered receiver, scenario and SNR. The table reveals that in scenario 1 at SNR1 OBS is perfectly able to distinguish between damaged and undamaged signals, since the three performances are at 100%; Figure 6(a) shows only a minor drop in performance when the SNR deteriorates to SNR2, but at SNR3 the higher noise level significantly hinders the defect detection, and, as seen in Figure 6(b), a relatively large number of false alarms is obtained if the threshold is set to enable detection of a large portion of the defects, the accuracies being all below 90%. Also, it appears that in scenario 1 receiver 1 is the best positioned to detect the signal changes due to reflections from the considered defects, then followed by receiver 2. As seen in Table 3, when OBS is tested on current measurements acquired at temperatures outside the baseline range (scenarios 2 to 4) the accuracy of detection is very poor at all SNRs considered, even for the most favorable case of scenario 2; indeed in Figure 6(c), the ROC curves for scenario 3 at SNR1 are close to the diagonal representing the random guess performance. Notably, in the unfavorable conditions of scenarios 2 to 4 receiver 1 typically yields the worst results.

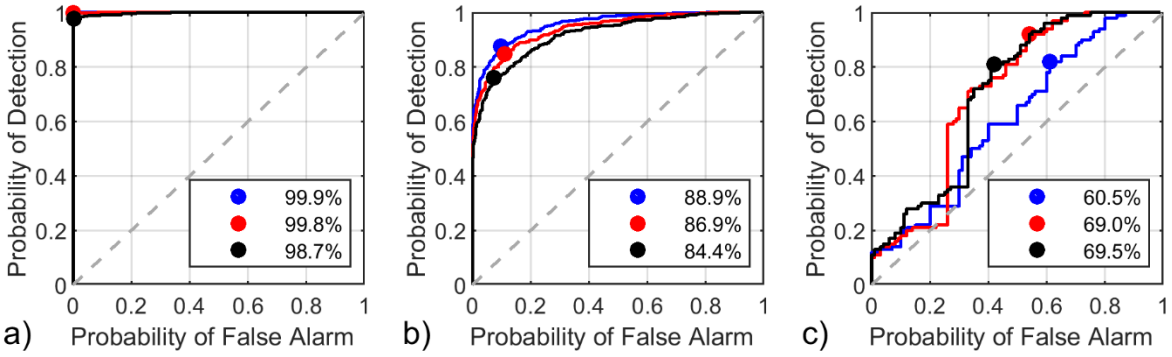

Figure 6. Receiver operating characteristic curves on the results of OBS+BSS for scenario 1 at (a) SNR2 and (b) SNR3, and (c) for scenario 3 at SNR1. Receivers 1, 2 and 3 are plotted in blue, red and black, respectively. The maximum accuracy of detection for every receiver and for every considered setting is given by the colored dots and shown as a percentage in the key. The light gray dashed line is the random guess performance.

Table 3. Accuracy of detection obtained with the conventional OBS+BSS method for every receiver and for every considered scenario and SNR, given as a percentage. Perfect detection (100% accuracy) is highlighted.

		Scenario 1			Scenario 2			Scenario 3			Scenario 4		
		R1	R2	R3	R1	R2	R3	R1	R2	R3	R1	R2	R3
OBS+BSS	SNR1	**100.0**	**100.0**	**100.0**	75.5	88.5	83.5	60.5	69.0	69.5	59.5	59.5	64.0
	SNR2	99.9	99.8	98.7	74.0	83.0	81.0	60.0	68.0	69.0	59.5	58.5	59.0
	SNR3	88.9	86.9	84.4	70.0	77.0	75.0	59.5	64.5	66.5	56.5	55.5	58.5

3. Multilayer perceptron network

3.1. Background

The perceptron algorithm was introduced by Rosenblatt in 1958 [39] and, with only few modifications, is still found in most ANN architectures used today. It consists of a layer of artificial neurons, where each neuron k accepts a total of N numeric input(s) x and returns a numeric output z_k as:

$$z_k = \varphi\left(\sum_{j=0}^{N} w_{kj}x_j + b_k\right) \tag{1}$$

where w_{kj} are weights applied to the inputs before summation, and typically the extra-term b_k is also included, which is referred to as the bias; the sum is then passed through a non-linear activation function $\varphi(\cdot)$ often being either the sigmoid (σ), the hyperbolic tangent (tanh) or the Rectified Linear Unit (ReLU) [40], which are plotted in Figure 7(b) where their equations are also given. A MLP network consists of multiple perceptrons, where each perceptron can comprise a different number of neurons. As seen in Figure 7(a), the first layer of a MLP consists of the input number(s) that are passed as input to each neuron of the second layer (this being referred to as the first 'hidden' layer), which in turn produces outputs that are passed to the next layer, and so on. The last layer of the network represents the final output(s). When MLP is used for a binary classification task, as in this work, a good choice for the activation function of the last hidden layer, which gives the final output, is the sigmoid (blue curve in Figure 7(b)), since it produces an output bounded between 0 and 1 that can be interpreted as the estimated probability of the positive class, which in this work was assigned to the 'damaged' conditions. For a more thorough review of MLP and other ANN architectures that have found widespread application in various fields, the book [41] is an excellent resource.

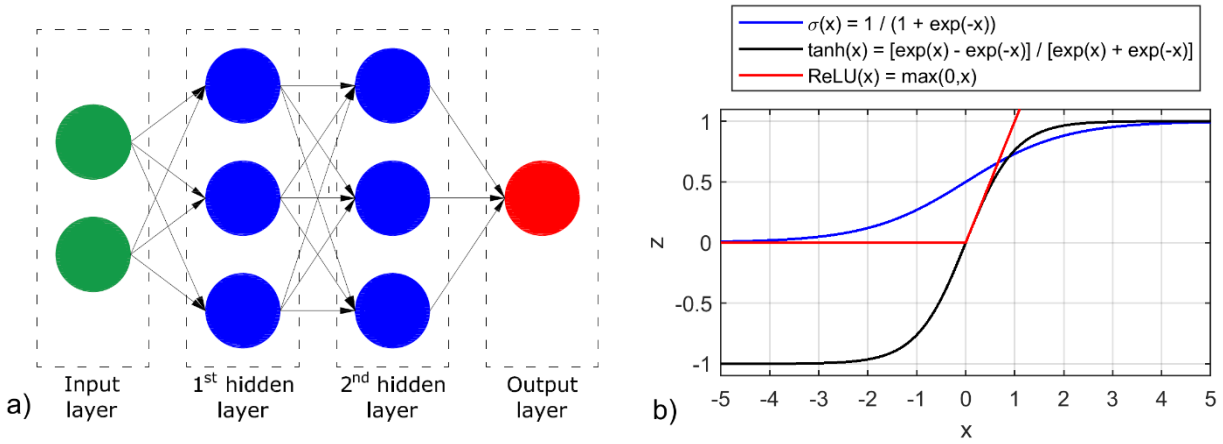

Figure 7. (a) Architecture of a multilayer perceptron with two inputs, two hidden layers each comprising three neurons, and one output neuron; the bias neurons are not shown. (b) Sigmoid, tanh and ReLU activation functions.

The training of MLPs (and of most other ML architectures) is performed iteratively by 'optimizer' algorithms that typically make use of 'backpropagation' [22] to update the parameters of the model, until a minimum for the problem-specific 'loss' function L is found. This process occurs by splitting the training set into small random sets of instances called 'mini-batches', on which at each iteration i the optimizer makes a prediction, measures the error via the loss function and then moves backward from the output layer to the earlier layers while computing the gradient of the loss function with respect to each weight and bias via the chain rule; the computed gradients are then used to update each weight and bias as:

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i - \alpha \nabla L(\boldsymbol{w}_i) \tag{2}$$

where the 'learning rate' α governs the step size of the change at each iteration (the equation for the bias terms is obtained by substituting $\boldsymbol{w}$ with $\boldsymbol{b}$). Note that advanced optimizers such as Adam [42] have the ability to adaptively change α as the training proceeds further.

Furthermore, ML networks possessing a very large number of parameters can often learn to fit the training dataset very well while giving poor results on the validation dataset. This is known as overfitting and indicates that the model was not able to extract the fundamental patterns contained in the training dataset. Various explicit 'regularizer' techniques exist to mitigate this issue, some of which are designed to modify the loss (e.g. 'L1' [43] and 'L2' [41]), others to modify the architecture during training (e.g. 'dropout' [44]). Although the description of these techniques is beyond the scope of this article, their use has been investigated in this work, and it is worth highlighting that the application of any of these three methods requires the definition of one dedicated hyperparameter.

3.2. Implementation

Some characteristics of the MLP networks tested in this study are shared with the other architectures described in the next sections, namely: (1) the input layer consisted of the 2,000 scalar values forming one BSS-compensated, FE-simulated measurement; (2) a sigmoid (blue curve in Figure 7(b)) was used as the activation function of the last layer of the model, hence producing a final output being a scalar between 0 and 1; (3) the model parameters were randomly initialized at the start of training; (4) the Adam optimizer [42] with an initial learning rate α of 0.001 was used; (5) a mini-batch size of 64 was employed; and (6) the 'cross-entropy log-loss' was chosen as the loss function, which is very well suited for problems of binary classification and which is defined as:

$$L(\boldsymbol{q}) = -\frac{1}{M}\sum_{1}^{M}\left[y_i \cdot \log \hat{y}_i\,(\boldsymbol{q}) + (1 - y_i) \cdot \log\left(1 - \hat{y}_i(\boldsymbol{q})\right)\right] \tag{3}$$

where M is the mini-batch size, $\boldsymbol{q}$ are the current parameters of the model (weights and biases in MLP), $\hat{y}_i$ is the output predicted from the current model and y_i is the true target, which is either 0 or 1 for undamaged or damaged conditions, respectively. Note that these choices of optimizer, learning rate and mini-batch size were made after performing several iterations of a randomized search over the full possible 'parameter space', though the details of this search are not discussed here for brevity.

Other hyperparameters that must be set in a MLP network are number of layers, number of neurons per layer, activation function used in the hidden layers, number of training epochs (i.e. number of passes through the entire training dataset), and, possibly, the parameter of a chosen regularizer. It is clearly not feasible to test every possible combination of these parameters, hence some choices need to be made to restrict the search field. The general idea used in this study regarding the dimension of the model was to start with a low complexity and then to gradually increase it while keeping track of possible improvements in the accuracy, e.g. by using only one hidden layer at first, then adding extra layers to investigate whether that added any benefit. When multiple hidden layers were used, the pattern employed to set the number of neurons in each layer was to form pyramids with fewer and fewer neurons at each layer, which is a common choice under the rationale that many lower level features can coalesce into far fewer higher level features while moving toward the final output [41].

Tanh and ReLU (black and red curves in Figure 7(b), respectively) were both tested as activation function for all hidden layers, although neither was determined to yield consistently better results than the other. To address possible issues of 'exploding and vanishing gradients' [45], the 'batch normalization' technique [46] was implemented before the activation function of each hidden layer. L1, L2 and dropout have been independently tested, and randomized searches of the parameter governing each of these techniques were performed; generally, the best results have been obtained by using L2, with a parameter that depended on the dimension of the tested model. Since training these networks was relatively quick on one NVIDIA® Quadro RTX™ 6000 GPU card, the number of epochs was set to be relatively large at 1,500 (this took about 30 minutes in the model that produced the results of Figure 8), and, as explained in section 2.3, the trained model was set to be the one which yielded the minimum loss on the validation set according to eq. (3).

3.3. Results

The MLP network was investigated on the dataset of scenario 1 at the three considered SNR levels. The hyperparameter search was significantly more challenging than those performed for the architectures described in the next sections, since the MLP models were generally very prone to overfitting, and the

learning curves consistently showed high scatter about the trendline, as seen in Figure 8(b). Many of the tested models were typically able to reach 100% accuracy on the training set, but the same level of accuracy was never reached for either validation or testing sets, hence suggesting that this type of architecture is not well suited to detect features and patterns in the raw time-domain signals that can unequivocally determine the presence of defect reflections. The network dimension that was found to give the best results consisted of four hidden layers formed by 2,000, 1,000, 500 and 250 neurons, respectively, with tanh activation function and the L2 regularizer sets to operate with a governing parameter of 10^{-6}; Figure 8(a) plots the accuracies yielded by this network in the testing set, which are also given in Table 5, while Figure 8(b) plots the accuracies measured in the training and validation sets during the first 200 epochs of training for the case of receiver 2 at SNR1. As seen in Table 5, the performance was always worse than that offered by the conventional OBS+BSS in scenario 1, thus scenarios 2 to 4 have not been investigated.

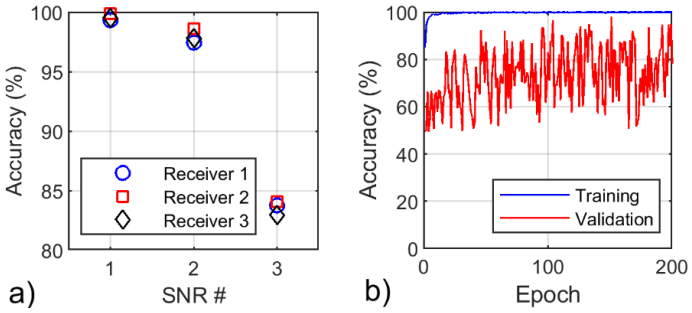

Figure 8. Case of multilayer perceptron (MLP) networks in scenario 1. (a) Accuracies in the testing set yielded by a MLP with four hidden layers formed by 2,000, 1,000, 500 and 250 neurons, respectively. (b) History of accuracy obtained during the first 200 epochs of training by the network of (a) for the case of receiver 2 at SNR1.

4. Recurrent Neural Network with LSTM

4.1. Background

RNNs are a class of architectures designed to handle sequential data, such as audio samples. In its basic form, and considering an input being an ultrasonic waveform sampled at a certain number of time steps, a RNN is composed of a 'recurrent neuron' that at each time step t receives the input value $x(t)$ at the same step as well as the 'memory state' output from the previous time step, $\boldsymbol{h}(t-1)$, where the bold character indicates that $\boldsymbol{h}$ can be a either a scalar or a vector, and produces both an updated memory state $\boldsymbol{h}(t)$ and an output $\boldsymbol{y}(t)$. This is shown schematically in Figure 9(a) in its 'unrolled' version, where the three neurons shown are actually the same recurrent neuron that is iteratively applied at each time step (note that the array indication $\boldsymbol{x}(t)$ in figure generalizes the problem to the case of a multidimensional input

array). The output sequence ***y(t)*** can either be used as final output of the model or can be given as input to another recurrent neuron, and so forth, hence obtaining deep RNNs. The training of these models is analogous to that of a MLP network, i.e. first a prediction is made, then backpropagation is run in the backward direction [22].

Thanks to the information passed over multiple time steps via the memory state ***h(t)***, theoretically at each time step the RNN can learn patterns from all past observations. However, practically, the information is gradually attenuated as it passes through successive steps, and completely disappears after only a few tens of data samples. To address both this issue and the possible formation of exploding gradients (i.e. ever-increasing gradients as backpropagation runs through a large number of steps), LSTM was developed in 1997 [23], which has been a game-changer in the field of sequence modelling, hence resulting in a vast array of RNNs based on LSTM. In LSTM the memory state is split into two vectors, ***h(t)*** and ***c(t)***, where essentially the former possesses a short-term memory (as in the standard RNN) and the latter a long-term one. Additionally, in LSTM the output ***y(t)*** is set to be identical to the short-term memory state ***h(t)***. Figure 9(b) shows the operations that take place in a LSTM cell in order to process the inputs and to produce the output and the updated memory states. A precise description of the operations and formulae are beyond the scope of this work and can be found in specialized textbooks [41], but essentially the information contained in both the short-term memory of the previous time step ***h(t-1)*** and the input data at the current time *t* is used to: (1) first eliminate some memories from the long-term memory state ***c(t-1)***, then (2) give it some new information, hence producing an updated state ***c(t)***; (3) produce an updated short-term memory state ***h(t)*** (and hence the output ***y(t)***) after filtering with the updated ***c(t)***. As a result of these operations, LSTM has the ability to learn which are the most important inputs, to store them in the long-term memory state for longer than a typical RNN (thanks to the forget gate that eliminates less useful memories), and to use them when needed.

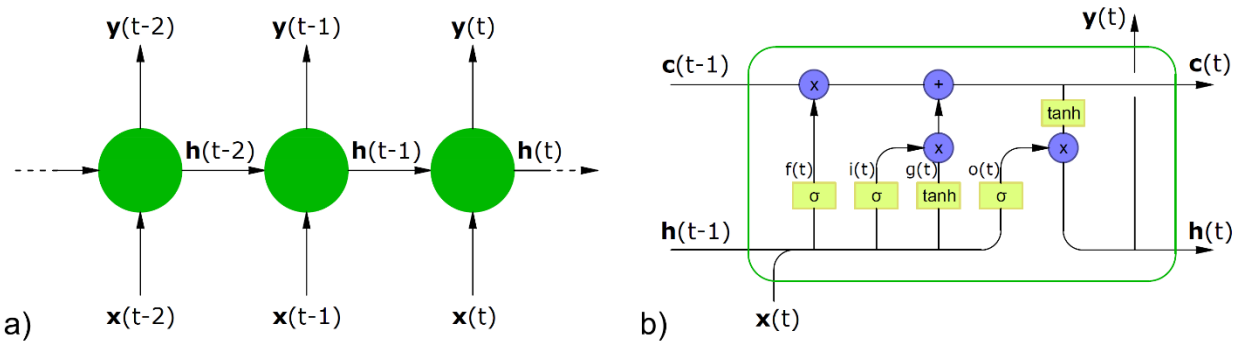

Figure 9. (a) Architecture of a recurrent neural network algorithm that sequentially processes the input ***x(t)*** and produces both a memory state ***h(t)*** and the output ***y(t)***. (b) Operations occurring in a long short-term memory (LSTM) cell, which splits the memory state into two vectors ***h(t)*** and ***c(t)***, where the former is also the output of the

model ***y(t)***; in the blue 'operator' circles, 'x' indicates element-wise multiplication and '+' indicates addition; sigmoid and tanh activation functions are also applied element-wise.

Despite the significant improvement over traditional RNNs, it has been observed that even by using LSTM networks, or other advanced versions of RNNs such as GRUs [24], the ability to learn patterns in previous samples is limited to a few hundred time steps [41], which might be insufficient in some NDE/SHM applications e.g. to track multiple reflections bouncing from geometrical boundaries of the waves scattered by a defect when the test geometry is large or when high testing frequencies, and hence high sampling rates, are used. A further drawback of this class of algorithm is that GPU-based parallelization during training is challenging due to their sequential nature, hence training time can be very long when the input signals are on the order of e.g. few thousand steps.

4.2. Implementation

The classic LSTM cell shown in Figure 9(b) was used as a base to test few different architectures. Although it would be technically possible to use only the output information at the last time step (in this study ***y(t=2,000)***) to predict whether a given ultrasonic signal is damaged or undamaged, so discarding the output produced by the last LSTM layer at all previous time steps, this was deemed disadvantageous due to the limited length of memory retained by these networks as discussed in the previous section. Therefore, it was decided to use one or more additional layers of either MLP or CNN to collect the output of the last LSTM layer at all time steps (***y(t)***) and to let this additional network learn which information and which time steps are more relevant for the task.

As for the MLP networks discussed in section 3.2, the general idea was to test LSTM architectures at increasing complexities, and to stop adding components once no further improvements were seen. Specifically, initial tests were performed by using one LSTM layer (i.e. one cell as in Figure 9(b), which can be seen unrolled as in Figure 9(a)), and by setting each cell to return a multidimensional output ***y(t)***, with dimension (at each sample) of either 1 (scalar output), 10, 50, 100 or 500, this dimension being denoted *LSTM_width*. The output array, which had total dimensions of (2,000 x *LSTM_width*), was then given as input to a further network based on convolutional layers identical to that described in section 5.2 and shown in Figure 12, where it is labelled 'output network'. However, these models were not able to find convergence on the training dataset and were discarded. It was then decided to stack one more LSTM layer above the first one, such that the input of the second layer was of dimensions (2,000 x *LSTM_width*) and the output was a mono-dimensional vector ***y(t)***. The latter was then passed through a perceptron consisting of a single neuron with sigmoid activation that returned a final scalar output between 0 and 1. This architecture showed much better capabilities of learning patterns of reflections from a defect within

the input ultrasonic signals, as suggested by the learning curves that often converged toward accuracies close to 100% in both training and validation sets, with the best performance found when using a *LSTM_width* of 100. The good performance on the validation set also showed that the network already possessed good generalization properties, so it was not necessary to add explicit regularizers to the model. Further tests were conducted on a deeper network obtained by inserting an additional LSTM hidden layer between the first and the final layers, where the additional layer were given the same dimensions as the first one; since this did not offer improvements from the architecture with two layers, while the training time was significantly increased (the time was observed to essentially increase linearly with the number of LSTM layers), no further tests were performed by inserting additional layers, and the results offered by the 2-layers network with *LSTM_width* of 100 are those shown in the next section. The training was set to last 180 epochs, which required about 24 hours of processing on one NVIDIA® Quadro RTX™ 6000 GPU card.

4.3. Results

Figure 10(a) summarizes the performance given by the LSTM network described in the previous section when tested on the dataset of scenario 1 at the three considered SNR levels; detailed results are also given in Table 5. Figure 10(b-c) plot the histories of both accuracy and loss on the training and validation sets during the 180 epochs of training for the case of receiver 2 at SNR1. This architecture showed a good improvement over the MLP networks of section 3, and the accuracies at all receivers were better than those of the conventional OBS+BSS method at the most unfavorable noise level of SNR3. However, the performance at SNR1 and SNR2 was either identical or slightly worse than that of OBS+BSS. Possibly, higher accuracies could have been achieved by increasing the number of training epochs, since in a few cases (including that of Figure 10(b-c)) the training and validation losses did not plateau by the end of the 180 epochs. However, the long training time itself already constitutes a significant drawback of the method, so this architecture was not analyzed further on the most challenging cases of scenarios 2 to 4.

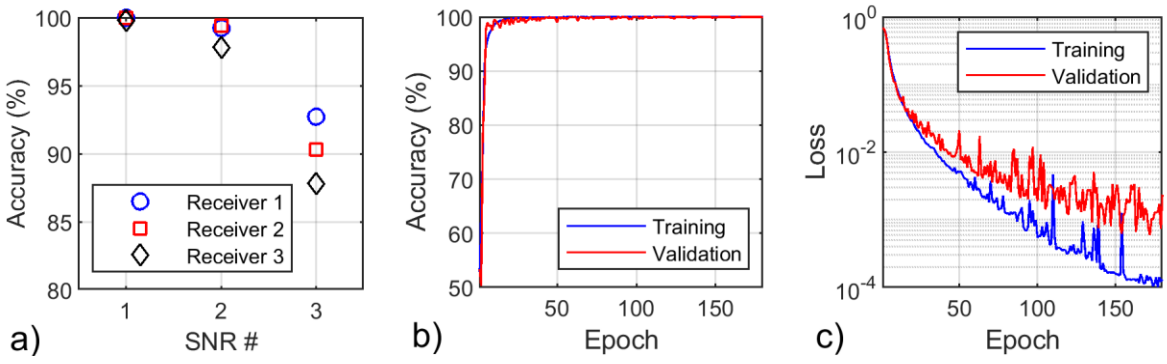

Figure 10. Case of long short-term memory (LSTM) networks in scenario 1. (a) Accuracies in the testing set yielded by a LSTM with two layers, where the dimensions of the array output by the layers were (2,000 x 100) and (2,000 x 1), respectively, and a perceptron unit with sigmoid activation received the latter array as input and produced an output scalar between 0 and 1. (b-c) Histories of (b) accuracy and (c) loss obtained during the 180 epochs of training by the network of (a) for the case of receiver 2 at SNR1.

5. Causal dilated convolutional neural network

5.1. Background

WaveNet was presented in 2016 [27] as a one-dimensional (1D) CNN designed to model sequences of, potentially, thousands of data samples. This was made possible by applying: (1) causal convolutions to the input that ensure that the network cannot violate the ordering in which the data is modelled, i.e. information from future data samples cannot affect the analysis at any given time step, as shown in Figure 11; (2) a 'dilation' scheme where a convolutional filter is applied over an area larger than its length by skipping input values at a given step, which allows the receptive field of the network to be increased exponentially with respect to the number of dilated convolutional layers [27,47], as also shown in Figure 11; (3) 'gated activation units' applied to the output of the dilated convolutions, as seen in Figure 12, of the form:

$$\mathbf{z} = \tanh\left(W_{f,k} * \boldsymbol{x}\right) \odot \sigma\left(W_{g,k} * \boldsymbol{x}\right) \tag{4}$$

where W indicates the convolution filter, f and g stand for filter and gate, $\mathbf{x}$ and $\mathbf{z}$ are input and output, $*$ and $\odot$ are the convolution and the element-wise multiplication operators and k is the index of the layer; (4) a residual learning framework containing additional 1D convolutional layers with filters of size 1 [48] and (5) skip connections [49] (both shown in Figure 12), which enable the information to jump over some layers and which have the effects of easing the training of the network and of speeding up convergence by addressing both the vanishing gradient [45] and the so-called 'degradation' issue [49]. The latter indicates a tendency of very deep networks to worsen their accuracy as the number of layers is increased [49].

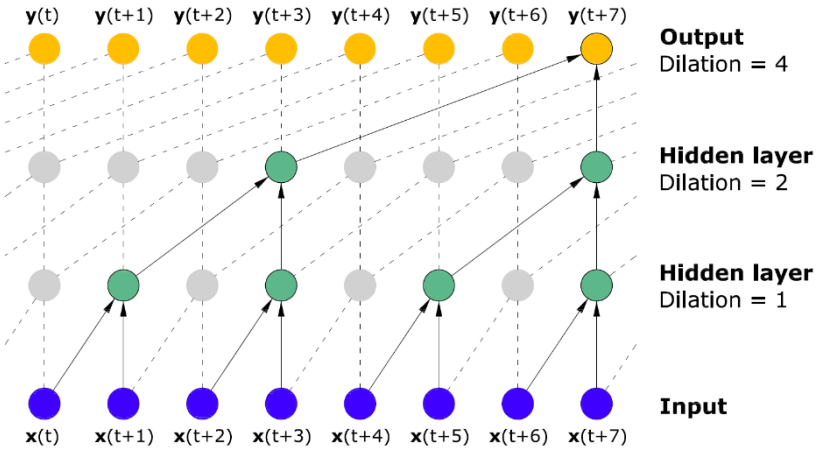

Figure 11. Example of causal dilated convolutional layers of the type employed in WaveNet [27]; the dilation rate is doubled at each layer, while the length of the filter (also known as 'kernel size') is fixed at 2. A blue cell represents the input value at time *t*; the arrows indicate the inputs used to perform a convolution, whose output is indicated with a green cell when that has occurred in a hidden layer and with a yellow cell when that has produced an output at time *t*. Analogous convolutions also occur at the grey cells via the grey dashed lines, though they are greyed out for clarity.

Thanks to the increasingly dilated convolutional layers, the first layers are exposed to a smaller number of data samples, and hence are expected to learn short-term patterns, while the latest layers can benefit from large receptive fields, and so can learn long-term trends. Another benefit of the method is that convolutional filters can be more efficiently parallelized, thus enabling a faster GPU-based training than what is possible in a RNN setting, which treats the data sequentially.

Both in the original paper [27] and in later works, WaveNet has been shown to obtain state-of-the-art performance on a number of audio tasks, such as for text-to-speech and speech/music generation. Given the obvious similarities between audio and ultrasonic signals, one would expect that a similarly good performance can be obtained for tasks typical of NDE and/or SHM, such as defect detection or localization; to the authors knowledge, this is the first attempt to produce and test a WaveNet-based network for NDE/SHM purposes.

5.2. Implementation

As done previously, the hyperparameter search was performed by gradually increasing the complexity of the model. This mostly involved varying the number of residual blocks as well as the dilation rate and the kernel size used in each block. The search was easier than that required for any of the other considered architectures, since the training time was relatively fast, and also, in general, it was possible to obtain convergence to good or very good accuracies for a large number of tested combinations of hyperparameters, hence suggesting that the model is not overly sensitive to a particular dimension.

Furthermore, as for the case of LSTM, it appeared that it was not necessary to use explicit regularizer techniques. As a result of the hyperparameter search, the architecture that has been retained for final testing is shown schematically in Figure 12 and described in detail in Table 4. A causal convolutional layer of kernel size (i.e. length of the filter) 2 is applied to the input signal, then the kernel size is fixed at 3 and the dilation rate is increased to 3, 9, 27 and 81 through four residual blocks. Finally, an output network was designed to collect the output of the residual blocks and to process it via two convolutions and two average pooling layers (see Table 4 for details), before applying a final sigmoid to produce a scalar between 0 and 1. This is the same output network tested in combination with LSTM that was mentioned in section 4.2, though it did not provide good results in that setting. While the residual blocks retain the original scheme used in WaveNet, though different parameters are used for kernel size and dilation rates, the output network is conceptually different than the original formulation of WaveNet, which was mainly designed for tasks involving data generation rather than binary classification. Early tests showed that a training duration of 900 epochs was typically sufficient to reach plateaus of both training and validation losses, and this only required about 30 minutes of processing on one NVIDIA® Quadro RTX™ 6000 GPU card when training on the full dataset of scenario 1, and about 15 minutes on the smaller datasets of scenarios 2 to 4.

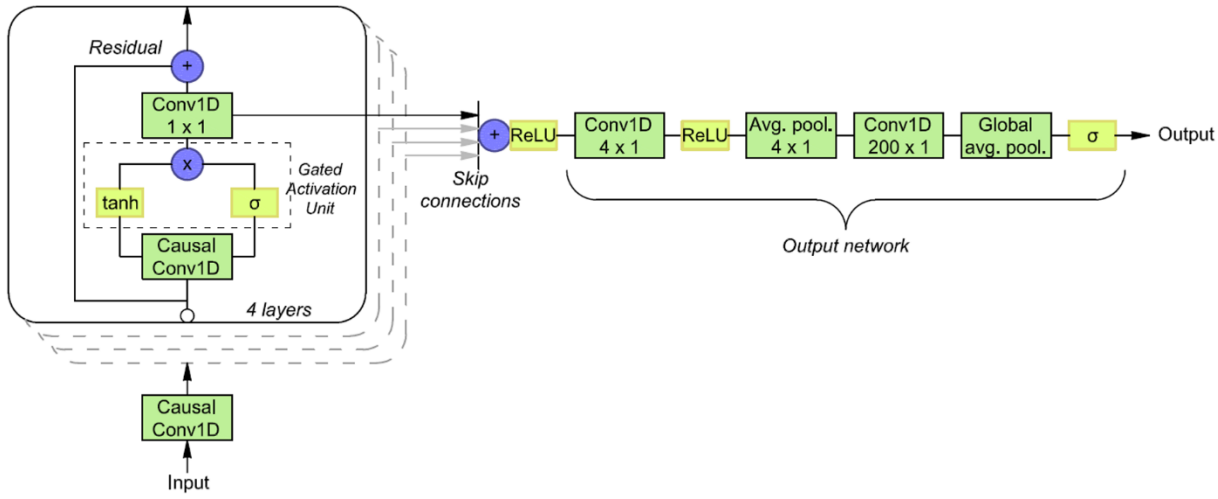

Figure 12. Schematic of the WaveNet-based architecture that was implemented for the task of defect detection; in the blue 'operator' circles, 'x' indicates element-wise multiplication and '+' indicates addition; sigmoid, tanh and ReLU activation functions are also applied element-wise; the ensemble of operations defined as 'output network' collects the output of the residual blocks and finally produces a scalar output between 0 and 1. Table 4 describes the operations in greater detail.

Table 4. Operations occurring in the implemented WaveNet-based architecture, which has a total of 11,649 trainable parameters.

Block	Layer	Type	Inputs	Filters	Kernel size	Stride	Dilation rate	Padding	Activation	Output size
Initial	In	Input	-	-	-	-	-	-	-	2,000 x 1
	C0	Convolution	In	16	2	1	1	Causal	Linear	2,000 x 16
Residual 1	C1t	Convolution	C0	16	3	1	3	Causal	Tanh	2,000 x 16
	C1s	Convolution	C0	16	3	1	3	Causal	Sigmoid	2,000 x 16
	M1	Multiply	C1t, C1s	-	-	-	-	-	-	2,000 x 16
	S1	Convolution	M1	16	1	1	1	Valid	Linear	2,000 x 16
	A1	Add	C0, S1	-	-	-	-	-	-	2,000 x 16
Residual 2	C2t	Convolution	A1	16	3	1	9	Causal	Tanh	2,000 x 16
	C2s	Convolution	A1	16	3	1	9	Causal	Sigmoid	2,000 x 16
	M2	Multiply	C2t, C2s	-	-	-	-	-	-	2,000 x 16
	S2	Convolution	M2	16	1	1	1	Valid	Linear	2,000 x 16
	A2	Add	A1, S2	-	-	-	-	-	-	2,000 x 16
Residual 3	C3t	Convolution	A2	16	3	1	27	Causal	Tanh	2,000 x 16
	C3s	Convolution	A2	16	3	1	27	Causal	Sigmoid	2,000 x 16
	M3	Multiply	C3t, C3s	-	-	-	-	-	-	2,000 x 16
	S3	Convolution	M3	16	1	1	1	Valid	Linear	2,000 x 16
	A3	Add	A2, S3	-	-	-	-	-	-	2,000 x 16
Residual 4	C4t	Convolution	A3	16	3	1	81	Causal	Tanh	2,000 x 16
	C4s	Convolution	A3	16	3	1	81	Causal	Sigmoid	2,000 x 16
	M4	Multiply	C4t, C4s	-	-	-	-	-	-	2,000 x 16
	S4	Convolution	M4	16	1	1	1	Valid	Linear	2,000 x 16
Skip connections	AF	Add	S1, S2, S3, S4	-	-	-	-	-	-	2,000 x 16
	ACT	Activation	AF	-	-	-	-	-	ReLU	2,000 x 16
Output network	C5	Convolution	ACT	16	4	1	1	Same	ReLU	2,000 x 16
	P1	Average pooling	C5	-	4	4	-	Valid	-	500 x 16
	C6	Convolution	P1	1	200	1	1	Same	Linear	500 x 1
	P2	Global avg. pooling	C6	-	-	-	-	-	-	1
	Out	Activation	P2	-	-	-	-	-	Sigmoid	1

5.3. Results

The WaveNet-based architecture described in the previous section was tested on all four scenarios and at the three considered SNR levels. Figure 13(a) and Figure 14(a) summarize the performance on scenario 1 and on scenarios 2 to 4, respectively; detailed results are given in Table 5. Figure 13(b-c) plot the histories of accuracy and loss on the training and validation sets during the 900 epochs of training for the case of receiver 2 at SNR1 on scenario 1. Both histories of loss start to converge almost immediately and quickly reach plateaus after about 300 epochs. The peak seen at about 50 epochs is probably due to the occurrence

of a ‘bad mini-batch’ characterized by high losses, which can temporarily destabilize the convergence when using optimizer methods based on the stochastic gradient descent such as Adam [50]. Figure 14(b-c) are analogous to Figure 13(b-c) but for the case of scenario 3, and also show losses that convergence quickly, plateauing after about epoch 500.

As seen in Table 5, this ML architecture outperformed the conventional OBS+BSS in all considered cases, except when both methods provided perfect 100% accuracy. Notably, this is the only method that gave 100% accuracy at the SNR2 level, though this was limited to the signals received by receiver 1 on scenarios 1 and 2. On scenario 1, the performance at all receivers at the highest considered noise of SNR3 is significantly better than that of OBS+BSS. Even more remarkable is the performance when testing occurs at temperatures outside the range of the training set, i.e. scenarios 2 to 4, where 100% detection is obtained at all receivers at SNR1 up to the highest temperature excursion considered in this study (this being the 30°C maximum difference existing in scenario 4). This suggests that this ML network is not simply learning to perform operations analogous to those of OBS, i.e. to compare the test signal to measurements acquired at similar temperatures found in the training set, but has the ability to learn patterns unequivocally related to the presence of waves scattered by damage, possibly making use of the existence of multiple reflections of those waves from the plate edges, since the recording duration was set to be long enough to include some of them. Furthermore, when considering the performances at SNR2 and at SNR3, only a minor drop in accuracy is seen as the temperature excursion in the testing set increases from the case of scenario 2 (up to 10°C) to that of scenario 4 (up to 30 °C). Finally, the results of Table 5 clearly indicate that receiver 1 is the best positioned to detect the considered defects, since it consistently yielded the best performance among the three receivers at all SNR levels and at all scenarios; this is in agreement with the OBS+BSS performance on scenario 1.

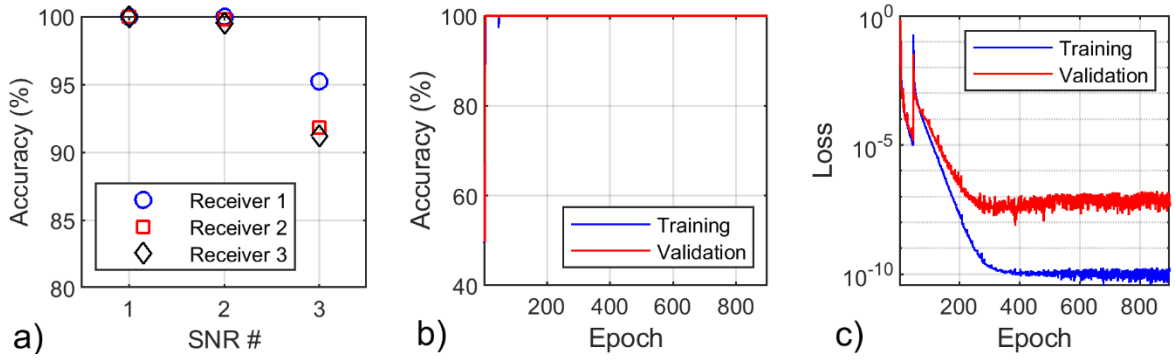

Figure 13. Case of the WaveNet-based architecture schematized in Figure 12 applied to scenario 1. (a) Accuracies in the testing set and histories of (b) accuracy and (c) loss obtained during the 900 epochs of training for the case of receiver 2 at SNR1.

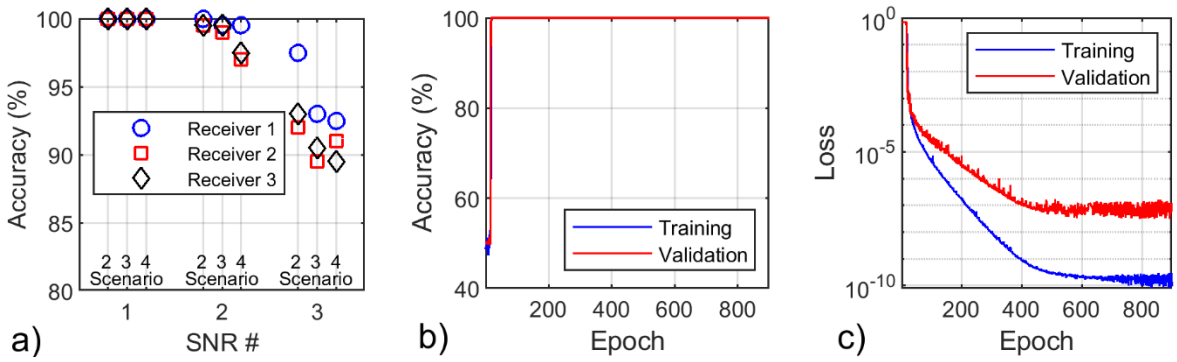

Figure 14. Case of the WaveNet-based architecture schematized in Figure 12 applied to scenarios 2 to 4. (a) Accuracies in the testing set. (b-c) Histories of (b) accuracy and (c) loss obtained during the 900 epochs of training for the case of receiver 2 at SNR1 in scenario 3.

Table 5. Accuracy of detection on the testing set that was obtained with the various tested methods for every receiver and for every considered scenario and SNR, given as a percentage. The results for OBS+BSS are copied here from Table 3 to facilitate comparisons. Perfect detection (100% accuracy) is highlighted.

		Scenario 1			Scenario 2			Scenario 3			Scenario 4		
		R1	R2	R3	R1	R2	R3	R1	R2	R3	R1	R2	R3
OBS+BSS	SNR1	**100.0**	**100.0**	**100.0**	75.5	88.5	83.5	60.5	69.0	69.5	59.5	59.5	64.0
	SNR2	99.9	99.8	98.7	74.0	83.0	81.0	60.0	68.0	69.0	59.5	58.5	59.0
	SNR3	88.9	86.9	84.4	70.0	77.0	75.0	59.5	64.5	66.5	56.5	55.5	58.5
MLP	SNR1	99.3	99.9	99.5	-	-	-	-	-	-	-	-	-
	SNR2	97.4	98.6	97.8	-	-	-	-	-	-	-	-	-
	SNR3	83.8	84.0	83.0	-	-	-	-	-	-	-	-	-
LSTM	SNR1	**100.0**	**100.0**	99.8	-	-	-	-	-	-	-	-	-
	SNR2	99.2	99.4	97.8	-	-	-	-	-	-	-	-	-
	SNR3	92.7	90.3	87.8	-	-	-	-	-	-	-	-	-
WaveNet (CNN)	SNR1	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
	SNR2	**100.0**	99.8	99.5	**100.0**	99.5	99.5	99.5	99.0	99.5	99.5	97.0	97.5
	SNR3	95.2	91.8	91.2	97.5	92.0	93.0	93.0	89.5	90.5	92.5	91.0	89.5

5.4. Application to experimental signals acquired on a composite plate

The WaveNet-based architecture described in the previous sections was further tested on a set of experimental Lamb wave signals acquired on a 500x500x2 mm composite plate, which was made available through the Open Guided Waves project [51]. Note that the experimental tests were not performed by the authors of this article. Here only a short summary of the testing procedure is given, while a detailed description of both mechanical properties of the plate and experimental setup can be found in [33,51,52]. Figure 15(a) shows a schematic of the test plate, which was composed of Hexply® M21/34%/UD134/T700/300 carbon pre-impregnated fibers and had a quasi-isotropic layup with a

stacking sequence of $[45/0/-45/90/-45/0/45/90]_S$; the plot indicates the positions of the pair of piezoelectric transducers used as actuator and receiver in a pitch-catch configuration, these being the same pair denoted $T_4 – T_{10}$ in the analysis of Moll et al. [33], and the locations where simulated damage was introduced (only one damage at a time in any test conducted in damaged conditions). This was realized by mounting an aluminum disc on the surface of the specimen via tacky tape. The damage locations are denoted D_{04}, D_{12}, D_{16}, and D_{24} to be consistent with [33,51,52], and represent the same four damaged conditions analyzed by Moll et al. [33]. The excitation signal was a 5-cycle, Hanning windowed toneburst at 40 kHz. The plate was instrumented with two temperature probes and was installed in an environmental chamber where it was subjected to temperature cycling between 20 and 60°C; signals were acquired every 0.5°C during both heating and cooling phases, for a total of 161 acquisitions per full cycle (only a single acquisition was taken at 60°C). Two temperature cycles were performed for the undamaged condition and one for each damaged location, though only the first cycle of the undamaged case was used for the analysis described in this work; therefore, for every plate condition, two signals are available at any temperature (except at 60°C), one being acquired during the heating phase and the other during cooling.

All signals were high-pass filtered using a Butterworth filter of order 6 and a cutoff frequency of 10 kHz and were down-sampled by a factor of 6 from the original 10 MHz sampling rate; then the BSS compensation method was applied using one of the two measurements acquired at 40°C in the undamaged plate as baseline, so obtaining signals ready to be fed to the ML algorithm. Figure 16(a) compares two signals acquired at 20°C in the plate in undamaged and in D_{04}-damaged conditions, respectively, each signal being composed of 2,185 data samples at the employed sampling rate. The direct arrivals of the S0 and A0 modes are labelled in figure, although the wave packets are probably overlapped with later arrivals due to reflections from the boundaries. Figure 16(b) shows the difference between the red and blue signals of Figure 16(a), where the multiple packets seen after 400 μs are essentially due to the interaction of the A0 mode with the surface damage, and the maximum amplitude seen at about 600 μs is approximately 8% of the peak value in Figure 16(a). Figure 16(c) is analogous to Figure 16(b) though obtained using damaged and undamaged signals at 57°C rather than 20°C, and shows that the effect of the simulated damage at higher temperatures is weaker, now being on the order of 4% of the peak value recorded in the A-scans at the same temperature (not shown for brevity, but also with a peak amplitude of ~0.2V as the signals of Figure 16(a)). This is possibly due to the effects of higher temperatures on the mechanical properties of the tacky tape.

Due to the significant differences existing between the FE-simulated signals analyzed in the previous sections and these experimental measurements (notably the very different excitation frequencies), the

WaveNet-based architecture schematized in Figure 12 and described in Table 4 had to be retrained on the newer dataset. To this purpose, as seen in Figure 15, all measurements acquired on D_{16} and D_{24} were assigned to the training set, those of D_{12} to the validation set, and those of D_{04} to testing. Therefore, at testing time the trained model had never seen any signal acquired on the D_{04}-damaged plate. The 161 measurements from the undamaged plate were assigned to each of the three sets following the scheme in the bottom row of Figure 15(b). As seen in Table 6, any measurement from the undamaged plate was used four times in order to establish a 50/50% division between damaged and undamaged signals in each dataset; this is different than the case of section 2.3, where the later addition of random noise actually produced a different (simulated) signal at each copy. The procedure resulted in a 50/25/25% ratio between training, validation and testing sets, as also seen in Table 6. Clearly, the rather low number of signals available in the training set does not represent an ideal scenario for training of a ML model such as that of Table 4, which possesses more than 10,000 trainable parameters, although it has been extensively observed that many deep learning algorithms have generalization capabilities that have not yet been explained [53].

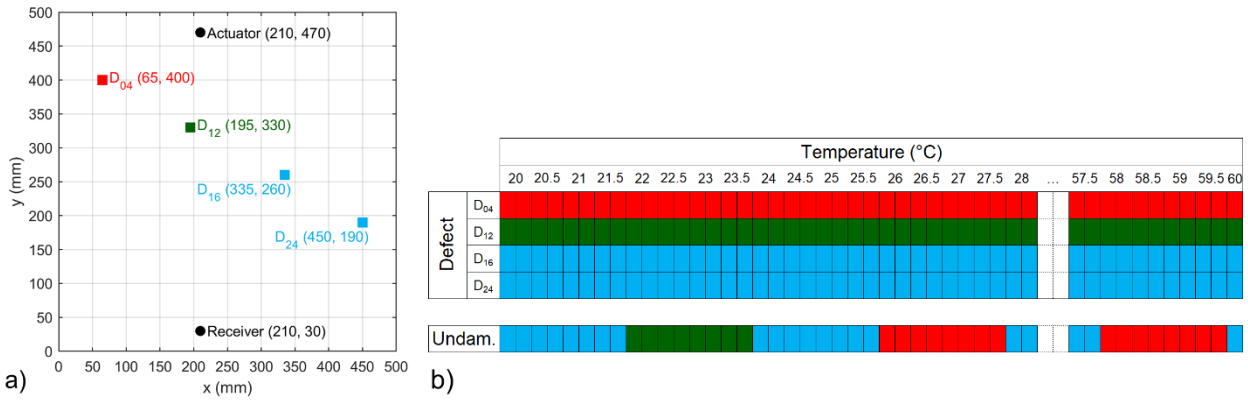

Figure 15. Experimental signals: division of dataset into training (cyan), validation (green) and testing (red) sets. (a) Schematic of the composite plate; the positions of the transducers and of the aluminum disc mounted on the plate to simulate damage are given in mm. Note that the same coordinate system as in the work of Moll et al. [33] is used. (b) All 161 measurements from a specific damage are assigned to the same set. For the undamaged plate, the 161 measurements at different temperatures are assigned to the three sets using the scheme shown in the bottom row.

Table 6. Experimental tests on the composite plate: number of signals used in training, validation and testing sets.

	Undamaged + damaged (% of total signals in the three sets)	Undamaged (number of unique tests x number of replicas)	Damaged
Training	646 (50.2%)	324 (81 x 4)	322
Validation	321 (24.9%)	160 (40 x 4)	161
Testing	321 (24.9%)	160 (40 x 4)	161

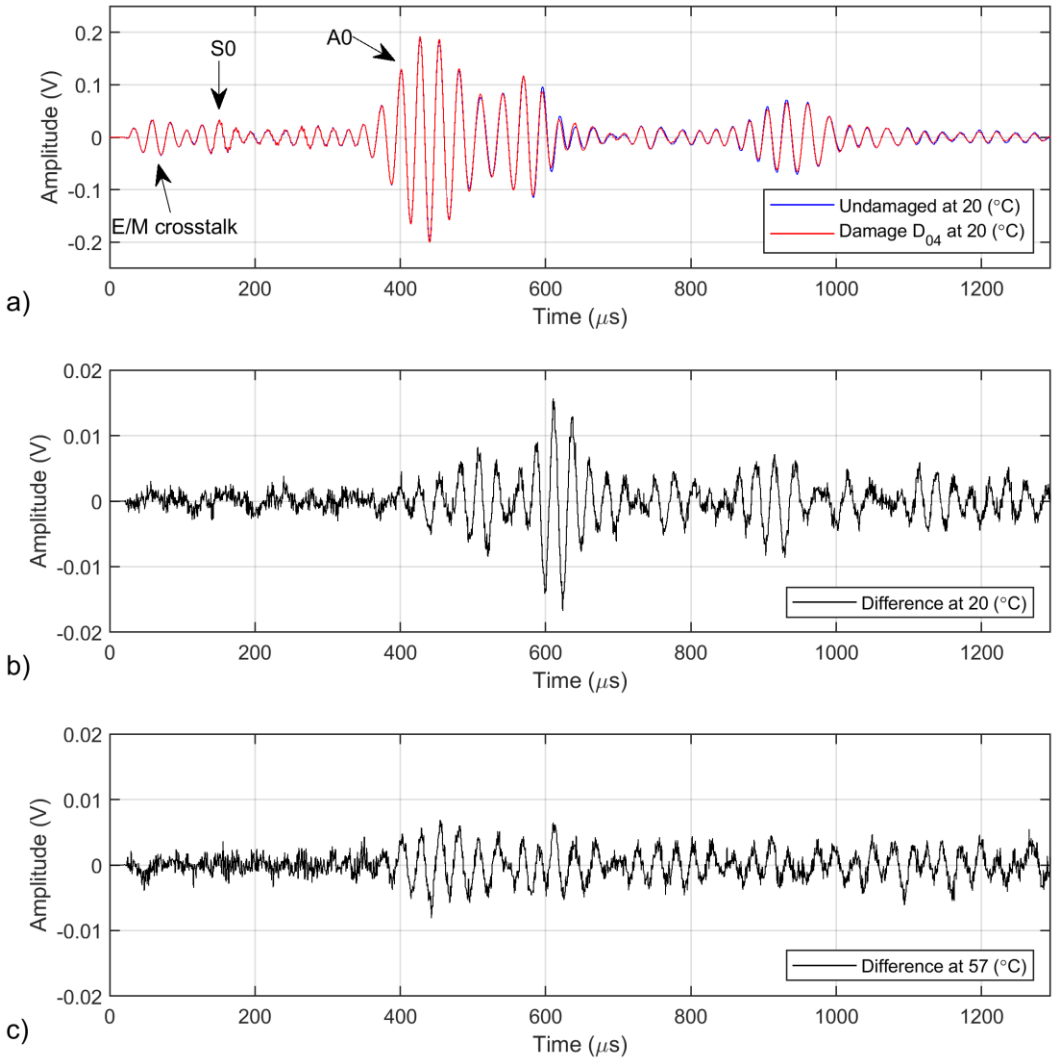

Figure 16. (a) Signals recorded at 20°C in the undamaged plate and in that including the damage labelled D_{04} in Figure 15(a); both signals are the result of BSS compensation using one of the two signals acquired in the undamaged plate at 40°C as baseline; the first arrivals of S0 and A0 modes are indicated. (b) Difference between the two signals of (a). (c) Analogous to (b), but using two signals recorded at 57°C.

As in section 2.4, the conventional OBS+BSS approach was applied to the dataset described in Figure 15, where the set of baselines for OBS included the undamaged signals at all temperatures except those of the testing set; the testing set was identical to that used for ML training (hence also adopting four replicas of the 40 unique undamaged measurements), and the monitored parameter for the computation of a ROC curve was the RMS of each residual signal. Figure 17(a) shows that OBS+BSS is essentially able to discriminate between damaged and undamaged measurements, though it did not achieve perfect detection accuracy, that being measured at 99.4%.

The training of the WaveNet-based architecture was quick due to the small number of signals in the training dataset, hence the number of epochs was set to 2,000, which only required about 5 minutes of

processing on one NVIDIA® Quadro RTX™ 6000 GPU card. The histories of accuracy and loss on the training and validation sets are given in Figure 17(b-c), which shows that after about 500 epochs the parameters of the model started to converge, after which both training and validation loss kept decreasing at a good pace, though did not plateau yet by the end of training. Nevertheless, the trained model was tested on the testing set and achieved 100% detection accuracy, so demonstrating that the proposed ML architecture can be successfully applied to experimental signals. Clearly, in this case the OBS+BSS approach was already very successful, since it gave an accuracy of 99.4%, so the improvement over that result can only be limited. The results of section 5.3 suggest that the improvement can be much more substantial when testing is performed at temperatures well outside the range covered in the training set; unfortunately testing this would require more experimental signals than are available in [51].

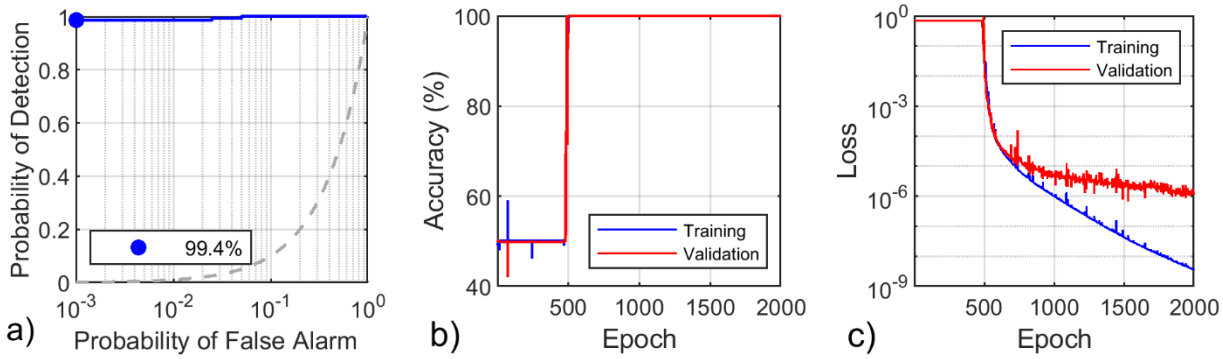

Figure 17. Analysis of the experimental dataset via (a) OBS+BSS and (b-c) the WaveNet-based architecture of Figure 12. (a) Receiver operating characteristic curve on the results of OBS+BSS in the testing set; the maximum accuracy of detection is given in the key; note that the plot is on semi-logarithmic scale, and the light gray dashed line is the random guess performance. (b-c) Histories of (b) accuracy and (c) loss obtained during the 2,000 epochs of training by the WaveNet-based architecture.

6. Conclusions

Supervised deep learning architectures potentially capable of inspecting ultrasonic guided wave signals to automatically detect the presence of defect signatures have been investigated. (1) MLP networks have been shown to be inherently incapable of modelling the ultrasonic sequences, and their best performance was inferior to that offered by the conventional and widely used OBS+BSS approach. (2) Recurrent neural networks in the form of LSTM showed better capabilities of learning features and/or patterns to flag the presence of defects, although their performance remained at best similar to that of OBS+BSS; also, they are affected by the significant drawback of requiring a very long time for parameter training. Finally, (3) an architecture based on causal dilated convolutional neural networks was shown to outperform both the other deep learning algorithms and the conventional OBS+BSS approach. Notably, its performance on Lamb wave signals obtained on a simulated steel plate remained almost unaltered

when the testing signals were acquired at temperatures up to 30°C higher than those of the training range, a condition in which the OBS+BSS was shown to be practically ineffective. Training this algorithm was shown to be relatively fast on GPU-based machines; for instance, it took about 30 minutes to train the network on the largest dataset considered in this study.

The proposed architecture was also benchmarked on a publicly available experimental dataset where, despite the limited number of signals available for training (on the order of hundreds), it gave 100% accuracy. This was in line with the results produced by the OBS+BSS approach, which gave an accuracy of 99.4%, indicating that the test case was not very demanding; nevertheless, it served to demonstrate that the proposed ML architecture can be successfully applied to experimental signals. The results from simulated data suggested that more substantial improvements may be expected when testing occurs at temperatures well outside the training range, but unfortunately the number of available experimental signals was insufficient to test this satisfactorily.

It has to be noted that conventional signal analysis methods such as OBS+BSS only need a set of baseline signals acquired on the undamaged structure, while the proposed supervised algorithm also requires measurements including reflections from damage, which might not always be practical to produce. Possible solutions to the problem can be to simulate damage in actual experimental settings, e.g. as done in the experimental dataset used in this study, or to complement actual measurements with FE simulations, e.g. by adding synthetic damage to experimental undamaged signals [54]. 'Transfer learning' techniques, which would involve training a large portion of the model on some specific testing configurations, such as that of the FE simulations performed in this study, and to re-train only the latest layers based on a limited number of measurements acquired on the specific testing setup of interest, can also alleviate the problem, if successfully applied. Clearly, the ability of the network to generalize to types of damage different than those used for training needs to be investigated. A further limitation of almost any ML algorithm is that it is very challenging, if not impossible, to interpret the logic and the computations occurring in the model itself, which acts as a 'black box'. Future work will be aimed at investigating the inner mechanisms that activate within the network when inspecting the ultrasonic signals, which will be performed by means of activation paths, adversarial samples and further parametric studies.

Finally, although the ultrasonic signals analyzed in this article were all acquired by sensors employed in a pitch-catch configuration and exploiting Lamb wave modes, it is expected that the proposed architecture would be equally applicable to tests performed in pulse-echo mode, hence using the same receiver as actuator and receiver, and possibly in passive monitoring, e.g. using the acoustic emissions technique; similarly, the use of Lamb wave modes per se does not seem to be a requirement, although it is arguably

advantageous for the network to exploit modes and a structural geometry that can produce multiple reflections from a single damage location in the received signals. Future work will be aimed to test the validity of these assumptions as well as to extend the methodology to perform tasks of defect localization within the test structure.

Acknowledgements

The first author is extremely thankful to Giuseppe Barbalinardo, currently at the University of California, Davis, for fruitful discussions on deep learning methods for sequence modelling, and to Prof. Peter Cawley for fruitful discussions during the preparation of the paper. The first three authors are grateful to Dr. Charles Wood, currently at the University of Portsmouth, for the fruitful discussions on machine learning in general, and specifically on convolutional neural networks.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] J.L. Rose, Ultrasonic guided waves in solid media, Cambridge university press, 2014.

[2] A.J. Croxford, P.D. Wilcox, B.W. Drinkwater, G. Konstantinidis, Strategies for guided-wave structural health monitoring, Proc. R. Soc. A Math. Phys. Eng. Sci. 463 (2007) 2961–2981. https://doi.org/10.1098/rspa.2007.0048.

[3] R.L. Weaver, O.I. Lobkis, Temperature dependence of diffuse field phase, Ultrasonics. 38 (2000) 491–494. https://doi.org/10.1016/S0041-624X(99)00047-5.

[4] H. Sohn, Effects of environmental and operational variability on structural health monitoring., Philos. Trans. A. Math. Phys. Eng. Sci. 365 (2007) 539–60.

https://doi.org/10.1098/rsta.2006.1935.

[5] T. Clarke, F. Simonetti, P. Cawley, Guided wave health monitoring of complex structures by sparse array systems: Influence of temperature changes on performance, J. Sound Vib. 329 (2010) 2306–2322. https://doi.org/10.1016/J.JSV.2009.01.052.

[6] S. Mariani, S. Heinlein, P. Cawley, Compensation for temperature-dependent phase and velocity of guided wave signals in baseline subtraction for structural health monitoring, Struct. Heal. Monit. 19 (2020) 26–47. https://doi.org/10.1177/1475921719835155.

[7] Y. Lu, J.E. Michaels, A methodology for structural health monitoring with diffuse ultrasonic waves in the presence of temperature variations, Ultrasonics. 43 (2005) 717–731. https://doi.org/10.1016/J.ULTRAS.2005.05.001.

[8] J.B. Harley, J.M.F. Moura, Scale transform signal processing for optimal ultrasonic temperature compensation, IEEE Trans. Ultrason. Ferroelectr. Freq. Control. 59 (2012) 2226–2236. https://doi.org/10.1109/TUFFC.2012.2448.

[9] A. Raghavan, C.E.S. Cesnik, Effects of Elevated Temperature on Guided-wave Structural Health Monitoring, J. Intell. Mater. Syst. Struct. 19 (2008) 1383–1398. https://doi.org/10.1177/1045389X07086691.

[10] G. Konstantinidis, P.D. Wilcox, B.W. Drinkwater, An Investigation Into the Temperature Stability of a Guided Wave Structural Health Monitoring System Using Permanently Attached Sensors, IEEE Sens. J. 7 (2007) 905–912. https://doi.org/10.1109/JSEN.2007.894908.

[11] C. Fendzi, M. Rébillat, N. Mechbal, M. Guskov, G. Coffignal, A data-driven temperature compensation approach for Structural Health Monitoring using Lamb waves, Struct. Heal. Monit. An Int. J. 15 (2016) 525–540. https://doi.org/10.1177/1475921716650997.

[12] A.B. Zoubi, V.J. Mathews, Temperature compensation for Lamb waves using a nonlinear model, in: Proc. Int. Work. Struct. Heal. Monit., Palo Alto, CA, 2019.

[13] A.C.S. Douglass, J.B. Harley, Dynamic Time Warping Temperature Compensation for Guided Wave Structural Health Monitoring, IEEE Trans. Ultrason. Ferroelectr. Freq. Control. 65 (2018) 851–861. https://doi.org/10.1109/TUFFC.2018.2813278.

[14] S. Mariani, S. Heinlein, P. Cawley, Location specific temperature compensation of guided wave signals in structural health monitoring, IEEE Trans. Ultrason. Ferroelectr. Freq. Control. 67 (2020) 146–157. https://doi.org/10.1109/TUFFC.2019.2940451.

[15] S. Mariani, P. Cawley, Change detection using the generalized likelihood ratio method to improve

the sensitivity of guided wave structural health monitoring systems, Struct. Heal. Monit. (2020) 147592172098183. https://doi.org/10.1177/1475921720981831.

[16] S. Mariani, Y. Liu, P. Cawley, Improving sensitivity and coverage of structural health monitoring using bulk ultrasonic waves, Struct. Heal. Monit. (2020) 147592172096512. https://doi.org/10.1177/1475921720965121.

[17] M. Thavasimuthu, C. Rajagopalan, P. Kalyanasundaram, B. Raj, Improving the evaluation sensitivity of an ultrasonic pulse echo technique using a neural network classifier, NDT E Int. 29 (1996) 175–179. https://doi.org/10.1016/0963-8695(96)80001-5.

[18] S. Legendre, D. Massicotte, J. Goyette, T.K. Bose, Neural classification of lamb wave ultrasonic weld testing signals using wavelet coefficients, IEEE Trans. Instrum. Meas. 50 (2001) 672–678. https://doi.org/10.1109/19.930439.

[19] F. Bettayeb, T. Rachedi, H. Benbartaoui, An improved automated ultrasonic NDE system by wavelet and neuron networks, Ultrasonics. 42 (2004) 853–858. https://doi.org/10.1016/j.ultras.2004.01.064.

[20] S. Sambath, P. Nagaraj, N. Selvakumar, Automatic defect classification in ultrasonic NDT using artificial intelligence, J. Nondestruct. Eval. 30 (2011) 20–28. https://doi.org/10.1007/s10921-010-0086-0.

[21] C. Sbarufatti, G. Manson, K. Worden, A numerically-enhanced machine learning approach to damage diagnosis using a Lamb wave sensing network, J. Sound Vib. 333 (2014) 4499–4525. https://doi.org/10.1016/j.jsv.2014.04.059.

[22] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature. 323 (1986) 533–536. https://doi.org/10.1038/323533a0.

[23] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Comput. 9 (1997) 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

[24] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, in: EMNLP 2014 - 2014 Conf. Empir. Methods Nat. Lang. Process. Proc. Conf., Association for Computational Linguistics (ACL), 2014: pp. 1724–1734. https://doi.org/10.3115/v1/d14-1179.

[25] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Backpropagation Applied to Handwritten Zip Code Recognition, Neural Comput. 1 (1989) 541–551. https://doi.org/10.1162/neco.1989.1.4.541.

[26] S. Bai, J.Z. Kolter, V. Koltun, An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, (2018). http://arxiv.org/abs/1803.01271 (accessed June 16, 2020).

[27] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, WaveNet: A Generative Model for Raw Audio, (2016). http://arxiv.org/abs/1609.03499 (accessed June 16, 2020).

[28] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. van den Oord, A. Graves, K. Kavukcuoglu, Neural Machine Translation in Linear Time, (2016). http://arxiv.org/abs/1610.10099 (accessed June 16, 2020).

[29] P. Cawley, Structural health monitoring: Closing the gap between research and industrial deployment, Struct. Heal. Monit. An Int. J. (2018) 147592171775004. https://doi.org/10.1177/1475921717750047.

[30] A. Ebrahimkhanlou, B. Dubuc, S. Salamone, A generalizable deep learning framework for localizing and characterizing acoustic emission sources in riveted metallic panels, Mech. Syst. Signal Process. 130 (2019) 248–272. https://doi.org/10.1016/j.ymssp.2019.04.050.

[31] D.F. Hesser, G.K. Kocur, B. Markert, Active source localization in wave guides based on machine learning, Ultrasonics. 106 (2020) 106144. https://doi.org/10.1016/j.ultras.2020.106144.

[32] J. Melville, K.S. Alguri, C. Deemer, J.B. Harley, Structural damage detection using deep learning of ultrasonic guided waves, in: AIP Conf. Proc., American Institute of Physics Inc., 2018: p. 230004. https://doi.org/10.1063/1.5031651.

[33] J. Moll, C. Kexel, S. Pötzsch, M. Rennoch, A.S. Herrmann, Temperature affected guided wave propagation in a composite plate complementing the Open Guided Waves Platform, Sci. Data. 6 (2019) 191. https://doi.org/10.1038/s41597-019-0208-1.

[34] P. Huthwaite, Accelerated finite element elastodynamic simulations using the GPU, J. Comput. Phys. 257 (2014) 687–707. https://doi.org/10.1016/j.jcp.2013.10.017.

[35] A. Raghavan, C.E.S. Cesnik, Finite-dimensional piezoelectric transducer modeling for guided wave based structural health monitoring, Smart Mater. Struct. 14 (2005) 1448. https://doi.org/10.1088/0964-1726/14/6/037.

[36] D. Alleyne, P. Cawley, A two-dimensional Fourier transform method for the measurement of propagating multimode signals, J. Acoust. Soc. Am. 89 (1991) 1159–1168. https://doi.org/10.1121/1.400530.

[37] R. Courant, K. Friedrichs, H. Lewy, On the Partial Difference Equations of Mathematical Physics, IBM J. Res. Dev. 11 (2010) 215–234. https://doi.org/10.1147/rd.112.0215.

[38] Research Computing Service | Imperial College London, (n.d.). https://doi.org/10.14469/HPC/2232 (accessed July 13, 2020).

[39] F. Rosenblatt, The perceptron: A probabilistic model for information storage and organization in the brain, Psychol. Rev. 65 (1958) 386–408. https://doi.org/10.1037/h0042519.

[40] X. Glorot, A. Bordes, Y. Bengio, Deep Sparse Rectifier Neural Networks, in: Proc. Fourteenth Int. Conf. Artif. Intell. Stat., 2011: pp. 315–323. http://proceedings.mlr.press/v15/glorot11a.html (accessed July 3, 2020).

[41] A. Géron, Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow, 2nd edition, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2019.

[42] D.P. Kingma, J.L. Ba, Adam: A method for stochastic optimization, in: 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc., International Conference on Learning Representations, ICLR, 2015. https://arxiv.org/abs/1412.6980v9 (accessed July 3, 2020).

[43] R. Tibshirani, Regression Shrinkage and Selection Via the Lasso, J. R. Stat. Soc. Ser. B. 58 (1996) 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x.

[44] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, (2012). http://arxiv.org/abs/1207.0580 (accessed July 5, 2020).

[45] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proc. 13th Int. Conf. Artif. Intell. Stat., 2010: pp. 249–256. http://www.iro.umontreal. (accessed July 6, 2020).

[46] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: 32nd Int. Conf. Mach. Learn. ICML 2015, International Machine Learning Society (IMLS), 2015: pp. 448–456. https://arxiv.org/abs/1502.03167v3 (accessed July 6, 2020).

[47] F. Yu, V. Koltun, Multi-Scale Context Aggregation by Dilated Convolutions, 4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc. (2015). http://arxiv.org/abs/1511.07122 (accessed July 10, 2020).

[48] K. He, X. Zhang, S. Ren, J. Sun, Identity Mappings in Deep Residual Networks, Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 9908 LNCS

(2016) 630–645. http://arxiv.org/abs/1603.05027 (accessed July 10, 2020).

[49] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016: pp. 770–778. http://image-net.org/challenges/LSVRC/2015/ (accessed July 10, 2020).

[50] J.M. Ede, R. Beanland, Adaptive learning rate clipping stabilizes learning, Mach. Learn. Sci. Technol. 1 (2020) 15011. https://doi.org/10.1088/2632-2153/ab81e2.

[51] Open Guided Waves, (n.d.). http://openguidedwaves.de/ (accessed July 12, 2020).

[52] J. Moll, J. Kathol, C.-P. Fritzen, M. Moix-Bonet, M. Rennoch, M. Koerdt, A.S. Herrmann, M.G. Sause, M. Bach, Open Guided Waves: online platform for ultrasonic guided wave measurements, Struct. Heal. Monit. 18 (2019) 1903–1914. https://doi.org/10.1177/1475921718817169.

[53] C. Zhang, B. Recht, S. Bengio, M. Hardt, O. Vinyals, Understanding deep learning requires rethinking generalization, in: 5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc., International Conference on Learning Representations, ICLR, 2019. https://arxiv.org/abs/1611.03530v2 (accessed July 13, 2020).

[54] C. Liu, J. Dobson, P. Cawley, Efficient generation of receiver operating characteristics for the evaluation of damage detection in practical structural health monitoring applications, Proc. R. Soc. A Math. Phys. Eng. Sci. 473 (2017). https://doi.org/10.1098/rspa.2016.0736.