7th International Conference on System-Integrated Intelligence (SysInt 2025)
04 - 06 June 2025, Bremen, Germany

Ultrasonic Guided Wave Signal Transformation with Polynomial Artificial Neural Networks for Data Augmentation

Christoph Polle[a,b,*], Stefan Bosse[c], David May[a,b]

[a] *Faserinstitut Bremen e. V., Am Biologischen Garten 2, Bremen 28359, Germany*
[b] *University of Bremen, Dept. of Production Technology, Germany*
[c] *University of Koblenz, Dept. of Computer Science, Koblenz, Germany*

Abstract

In this study we investigate signal transformation using Polynomial Artificial Neural Networks and the challenges correlated with their training. Measuring technologies based on Guided Ultrasonic Waves (GUW) can be used for Structural Health Monitoring (SHM). In recent years Machine learning (ML) algorithms are increasingly applied to GUW technologies to predict the presence and location of damages. A major challenge is providing sufficient training data. This is because experimental recording of measuring data with a broad parameter space is a time-intensive operation when using surface bonded transducers, since the transducers have to be attached and detached at different positions. A possible solution to this challenge is the use of scanning methods like air-coupled measured GUWs, which allow a fast measurement at different sensing positions. In SHM bonded transducers are typically used, so that a transformation between both methods is necessary. So far, it was shown in previous work that the transformation is generally possible, but due to the complexity of GUW signals, also prone to signal generation errors. In the present work, the class of Polynomial Artificial Neural Networks (PANN) is introduced to optimize the transformation process, where the polynomial coefficients are trainable parameters. Sigmoid and tanh functions are widely used due to their bound gradient, essential for stability in gradient-based training methods. Polynomial functions do not have a bound gradient. For that the coefficients are set before the model training in a way that the polynomials fit a GUW partially, e.g. ascending and descending flank of the wave. This facilitates the training of the model and minimizes the problem of a not bound gradient. We assume (but to be proven) that the signals of both measurement techniques should have a common feature vector, which can be approximated with an encoder-decoder architecture providing the transformation function between air-coupled and surface bond GUW signals.

Keywords: Guided Ultrasonic Waves, Piezo Electric Transducer, Machine Learning, Artificial Neural Networks, Polynomial Artificial Neural Networks, Trainable Activation Function, Sensor Data Augmentation/Transformation

1. Introduction

Guided Ultrasonic Waves (GUW) are waves propagating through plate-like structures and were first described mathematically by Horace Lamb [1]. Due to their interaction with inhomogeneities within a plate, GUW signals can be used for Structural Health Monitoring (SHM) applications [2]. And because of their ability to travel large distances, GUW signals can monitor a large area with a relatively small amount of sensors [3, 4]. The disadvantage of GUW signals for SHM applications is that they are influenced by environmental conditions like temperature [5, 6]. Also, the recorded GUW signal changes with the position of the used sensor, and with that, the sensitivity to the occurrence of damage. Therefore, the sensor placement has to be optimized to ensure a large coverage of the monitored specimen [7, 8]. Due to these influences, a large amount of data has to be processed to cover all non-damage-related signal changes in order to use GUW data for SHM applications. Since Machine Learning (ML) approaches can process these large amounts of data and are able to distinguish between the signal changes, ML has been used increasingly for GUW-based SHM in the past years [9]. But the time-consuming measurement of GUW signals under the different environmental and spatial changes remains a major challenge. To reduce the measurement time for experimental values, simulated data can be used for ML-training [10, 11, 12]. The drawback of this approach is that simulations often differ from real-world data, so there are limits to the transferability. Data augmentation techniques are another possibility to increase the amount of data available. In [13], for example, it was shown that signal features can be generated over a large temperature range.

Data augmentation can be performed commonly in three different ways:

1. Stochastic (Monte Carlo simulation-based) or geometrical recomposition by using experimental data;
2. Simulation;
3. Random process data generators, e.g., by using Generative Adversarial Networks or Variational Autoencoder architectures [BOS24A], trained from experimental or simulation data.

The first two methods are well understood but are too limited. Any stochastic or geometric augmentation (e.g., by combining parts of signals from different measurements) do not provide fully independent new data. The third approach is promising to generate independent new data, but fails due to physical incorrect signal generations, which are hard to be detected and assessed. Additionally, experiments in [14] showed a narrow parameter space of the generated GUW signals, not suitable for data augmentation with a broad parameter space (e.g., damage position, wave base frequency and so on). Moreover, the GAN architectures cannot be still controlled with respect to the parameter setting. These generative models producing artifacts like ripple or distorted signals, which are not be observed in real data. Generation is only possible within the training parameter space (interpolation, if any), but produces high errors outside the trained parameter space (extrapolation). GAN models produce artifacts that can have a significant impact (distortion) on data-driven predictor models, which should be trained with generated data only. The validation of the generated signals is a challenge, and classical correlation or distance measures are not suitable for signals containing weak features (to be detected). Instead, data-driven predictor models can be used to predict the parameter set of a generated signal. But the auxiliary predictor model is purely data-driven and need to be validated, too, which is not possible in most cases as long there is no ground-truth data set. Surrogate models show a good coverage of the trained parameter space, whereas GAN-based models tend to narrow the parameter space towards central average values. The control of the data generation of GAN models is still a challenge, and often only entangled parameter vectors are available (i.e., single parameters like the damage position cannot be set independently). Therefore, the method proposed in this work has a much higher potential to fill the parameter-data-space gap.

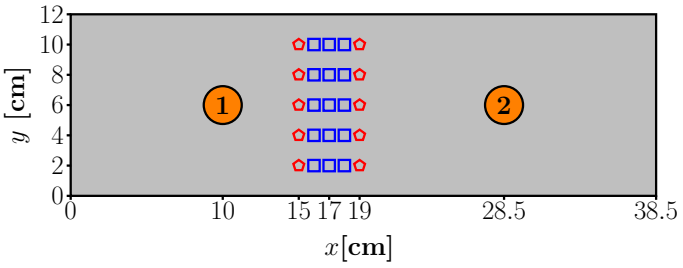
In past work [15], another approach was introduced by the authors of this article, where different neural network (NN) architectures were used to transform the GUW signals measured by the air-coupled measurement system (ACMS) to signals measured by surface-bonded piezoelectric wafer active sensors (PWAS). This has the advantage that one can perform fast AMCS measurements instead of time-consuming surface-bonded measurements where the PWAS has to be attached to the surface for every position. It was shown that in general it is possible to perform such a transformation, but that there were still some shortcomings, like large errors for some data and for all data a small basic error within a tolerable range.

In the current work, the approach from [15] will be refined by the introduction of Polynomial Artificial Neural Networks (PANN) [16]. In PANN, polynomials serve as activation functions (AF), where the polynomial coefficients are trainable parameters within the PANN. This makes PANN optimal for fitting complicated functions. Since in [15] it was shown that the autoencoder (AE) architecture worked best for the signal transformation, AE architecture is also used in the present work, where NN with tangens hyperbolicus (tanh) and polynomial (poly) were trained for comparison. Since PANN introduces new learnable parameters, the number of hyperparameters also increases. To choose the best hyperparameter, limited hyperparameter tuning is performed. For the experiments, the data pool from [15] was used. In the course of the study, an extended error analysis was conducted.

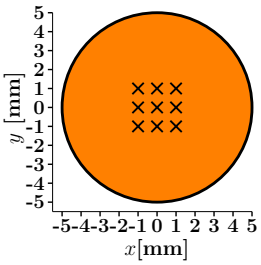
2. Materials and Methods

2.1. Experimental Setup

As mentioned above, the data used is the data from [15]. The GUW signals were measured on a small GLARE5-3/4 plate with dimensions $38.5 \times 12 \times 0.23$ cm. Two PWAS transducers (DuraAct from PI Ceramics) were attached to the plate for GUW signal actuation and surface-coupled measurements (SCM). To increase the number of training data, magnets were attached to the plate at different positions. The magnets introduce a local inhomogeneity that alters the measured GUW signals. A sketch of the plate and the positions of the transducer and the magnets can be seen in Figure 1a. As a sensor for the Air Coupled Measurements (ACM), a MEMS microphone (SPU0410LR5H-QB from Knowles Electronics) was used. The GUW signals were actuated and measured with a TiePie Handyscope HS5-540XMS, which was connected to the PWAS actuator and sensor as well as to the MESMS microphone. For every magnet position, as well as for the case where no magnet was attached, a GUW signal was actuated from transducer 1. The GUW signal then propagated through the plate and was recorded at transducer 2 for SCM. For the ACM, the MEMS microphone measured the GUW signal at 9 positions within the surface of the transducer; the scanning positions can be seen in Figure 1b. It should be mentioned that the scans were not performed directly on the transducer but on the other side of the plate to ensure that the original surface waves from the plate were captured. This measurement was also performed with transducer 2 as actuator and transducer 1 as sensor, but in the present work this data was not used.



(a) Sketch of GLARE-Plate used for GUW measurements. Orange circels donate transucers attached, squares donate magent positions used for model training, while polygons donate magent positions used as test data.



(b) Sketch of scanning positons at the sensor location.

Fig. 1. Sketches of experimental setup, a) plate and b) sensor.

The pitch signal was a 25 kHz Hanning windowed sine wave signal with five cycles. The frequency of 25 kHz was used because at this frequency the MEMS microphone is most sensitive [17].

2.2. Data Preprocessing

Before being used for training and testing the models, the GUW data was preprocessed. At first the data was normalized to the maximum so that the amplitude values were within [-1, 1]. In the next step, the data was shortened by cutting the last part of the GUW time signal, which reduced the number of data points from 6249 to 4000. This was done to save computational time and to cut off a large part of attenuating

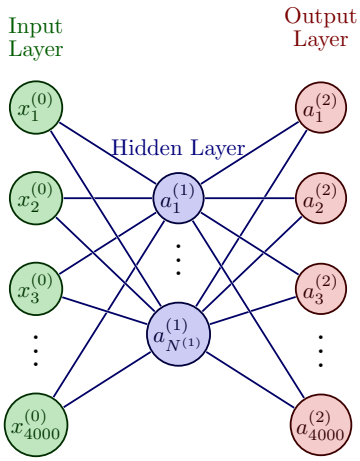


Fig. 2. Graph of the AE architecture used in this study. The architecture consists of one input and output layer with 4000 nodes each and a hidden layer with $N^{(1)}$ nodes, with $N^{(1)} = 200, 300$. The x_i denotes the input values, while the a denotes the node values after the activation function was applied to the node input.

reflections. Like described in chapter 2.1, the ACM consists, for every magnet position, of multiple GUW signals scanned at the position of the PWAS 2 (see Figure 1). Since the GUW signal measured with the PWAS is a summation of the wave displacements over the whole area of the PWAS, the ACM scans were averaged to one time signal to produce a signal that is also a summation of the wave displacements over a defined area.

Finally, to increase the number of training and test data, a Monte Carlo approach was used to add Gaussian noise to the data, whereby a maximum deviation of 2.5 % was permitted.

2.3. Neural Network Architecture and Training

Since in [15] it was found that among the analyzed architectures, the autoencoder (AE) was best for the signal transformation, in this work just AEs are used. In Figure 2, a sketch of the utilized AE architecture is shown. All AEs consisted of $L = 3$ layers with one input and one output layer with 4000 nodes each, which correspond to the number of data points in the GUW signals. The hidden layer of the AE had $N^{(1)}$ nodes, where $N^{(1)} = 200, 300$. The AE was fully connected with weights $w_{ij}^{(l)}$, where $l = [1, 2]$ is the index of the layer, while i and j are indices of the connected nodes, where i is the index of the node of layer l and j is the index of the node in layer $l - 1$.

The x_i in Figure 2 are the input values of the input layer. While the $a_i^{(l)}$ on the other hand denote the node values of layer l after the activation function $f_i^{(l)}(z_i^{(l)})$ was applied to the node input $z_i^{(l)}$. In this study two kinds of activation functions were used: polynomials with trainable coefficients and, for comparison, tanh, so that $a_i^{(l)}$ is defined as:

$$a_i^{(l)} = f_i^{(l)}(z_i^{(l)}) = \begin{cases} \sum_{d=0}^{D} c_d^{(l)} \cdot z_i^{(l)d}, & \text{if activation = poly} \\ \tanh(z_i^{(l)}), & \text{if activation = tanh} \end{cases} \tag{1}$$

where D is the degree of the polynomial and c_d are the trainable polynomial coefficients. Looking at 1, one can see that if the activation function is a polynomial, the activation function can be different for each node since the coefficients are trained separately for each node, while in the other case, the activation function is always the tanh function.

For the training of the polynomial coefficients, two approaches were used. At first the coefficients were trained in parallel with the weights $w_{ij}^{(l)}$ of the AE. In the second approach, at first the weights of the AE

were trained, and after that the polynomial coefficients were fine-tuned in a single training epoch. The first approach will be labeled as parallel and the second as split.

For the optimization of the polynomial coefficients, the algorithm proposed in [16] was used. In a classic backpropagation algorithm, inputs from a training data set are transferred to the NN during training, which are processed by the NN in a forward propagation step. The true results y associated with the input data are known. The outputs $\tilde{y}$ (in this case $a_i^{(3)} = \tilde{y}_i$) of the NN are then compared with the y value using an error function E (e.g., $E = \tilde{y} - y$). In the backpropagation step, all weights $w_{ij}^{(l)}$ are then updated using the chain rule:

$$\Delta z_i^{(l)} = \begin{cases} E & \text{if } l = L-1, \\ f'^{(l)}(a_i) \sum_j w_{ji}^{(l+1)} \Delta z_j^{(l+1)} & \text{else,} \end{cases} \tag{2}$$

$$\Delta w_{ij}^{(l)} = a_i^{(l-1)} \Delta z_i^{(l)}, \tag{3}$$

$$w_{ij}^{(l)\text{new}} = O_\text{w}\left(w_{ij}^{(l)\text{old}}, \Delta w_{ij}^{(l)}\right), \tag{4}$$

where $f'^{(l)}()$ denotes the derivative of the activation function $f^{(l)}()$. The weights w_{ij}^{l} are updated by optimization function O_w. In this work O_w corresponds to the ADAM optimizer with the hyperparameters $\alpha_\text{w} = 0.001, \beta_\text{w1} = 0.9$, and $\beta_\text{w2} = 0.99$.

For the training of a PANN, some extra steps have to be performed to optimize the polynomial coefficients $c_i^{(l)}$ with the backpropagation algorithm. In addition to the steps in equations 2, 3, and 4, the following steps must be taken. For a deeper description of the algorithm, the curious reader is referred to paper [16], which also provides evidence for the stability of the PANN:

$$p_i^{(l)} = \sum_{d=0}^{D} z_i^{(l)d}, \tag{5}$$

$$\Delta c_i^{(l)} = \sum_{i=1}^{N^{(l)}} \Delta z_i^{(l)} \cdot p_i^{(l)}, \tag{6}$$

$$c_i^{(l)\text{new}} = O_\text{c}\left(c_i^{(l)\text{old}}, \Delta c_i^{(l)}\right). \tag{7}$$

The coefficients are optimized by an optimization function O_c, in this case again the ADAM optimizer. As there are no empirical values for optimizing the coefficients, hyperparameter tuning was carried out for D, α_c, β_c1, and β_c2. Also, different architectures were used to optimize the results, so, as mentioned above, the number of nodes in the hidden layer changed, and also the use of polynomial activation in the hidden layer and output layer was investigated. In Table 1, all parameters with the investigated values are shown.

Table 1. Parameter values used for model training.

Parameter	Values
D	2, 3, 4
α_c	0.001, 0.0001
β_c1	0.4, 0.5, 0.6, 0.9
β_c2	0.59, 0.69, 0.99
N^1	200, 300
Activations $[f^{(1)}(), f^{(2)}()]$	[tanh,tanh],[tanh,poly],[poly,tanh],[poly,poly]

In order to optimize the training of the PANN, the polynomial coefficients were initialized by fitting a polynomial of degree D to a cosine function. The idea is that in this way the activation functions have at

least partially the form of a wave, and with that, a better chance to fit complicated waveforms like a GUW signal. In Figure 3, an example of the fits with the different cosine functions used in this study is shown.

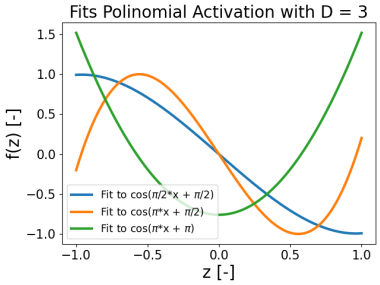


Fig. 3. Example of polynomial functions with $D = 4$ fitted to different cosine functions to determine initial coefficients.

3. Results and Discussion

In the following, only the models with the parameters that delivered the best results are presented. In Table 2 the parameter of these models are shown.

Table 2. Parameter of Models with the best Resultes

Parameter	Poly: $cos(\pi/2 * x + \pi/2)$		Poly: $cos(\pi * x + \pi/2)$		Poly: $cos(\pi * x + \pi)$		Tanh
	Parallel	Split	Parallel	Split	Parallel	Split	
D	3	4	3	3	4	4	-
α_c	0.001	0.0001	0.0001	0.001	0.001.	0.001	-
β_{c1}	0.6	0.9	0.6	0.6	0.4	0.4	-
β_{c2}	0.69	0.69	0.59	0.69	0.99	0.59	-
$N^{(1)}$	300	200	200	200	200	200	200
Activations	[tanh,poly]	[poly,tanh]	[tanh,poly]	[tanh,poly]	[tanh,poly]	[tanh,poly]	[tanh,tanh]

In Figure 4, an example of a generated PWAS signal in comparison to the original measured PWAS signal is shown. Looking at Figure 4, one can see that the transformation generally worked quite well over the whole time. However, there is a time span where the transformation produces strongly visible errors. To put the analysis a bit further, simulations with the dispersion calculator [18] were performed to estimate the beginning and end of the A_0 and S_0 modes. To simplify the analysis of the signals, colored areas were introduced in the plots to mark different time periods in the signals. The red area marks the time when just noise is present, and the end of this area also marks the beginning of the S_0 mode. The green areas mark the time spans at which the signal transformation worked quite well, while in the yellow area the transformation produced significant errors. Also, the end of the yellow area marks the end of the A_0 mode. The black lines mark the time in which A_0 and S_0 modes are simultaneously present. So that the left line marks the beginning of the A_0 mode and the right line marks the end of the S_0 mode. With this information, it seems that at times at which both modes are present, the transformation from ACM to SCM is problematic.

To further compare the results of the different trained models, the averaged relative error (ARE) was computed for different signal features F, like the signal envelope and the phase of the signal:

$$ARE = \frac{100}{N_g} \sum_{s=1}^{N_g} \frac{|F_s^g - F_s^o|}{|F_s^o|}, \tag{8}$$

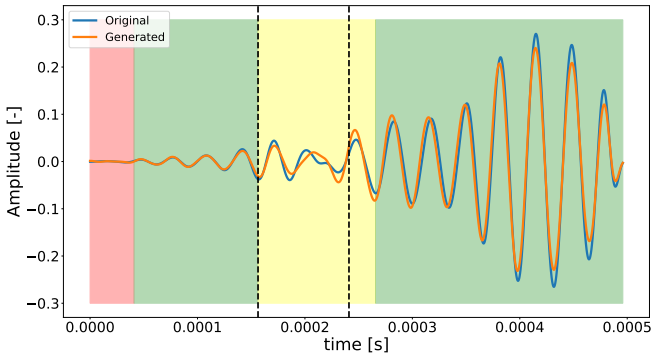


Fig. 4. Example of a generated PWAS signal in comparison with the measured PWAS signal. Both signals were normalized with respect to the signal maximum. The green areas mark the time when the transformation of the signal worked well. The yellow area, on the other hand, marks the time space where the transformation produced a large error, while the black lines in that area indicate the time when A_0 and S_0 modes are both present at the same time.

where F^{g} stands for the generated and F^{o} stands for the original signal, while s is the index of the current signals and N_{g} is the number of generated PWAS signals.

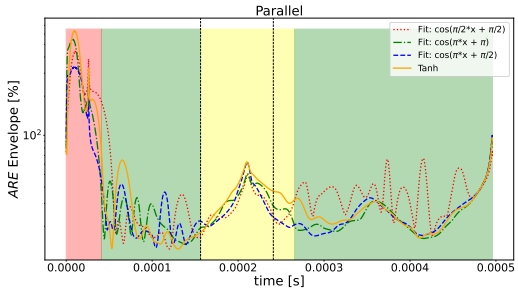
In Figure 5, the results of the *ARE* for the signal envelope are shown, with the same color area code as above. The results of the models where weights and coefficients were trained in parallel are shown in Figure 5a, while the results of the split training of the weights and coefficients are shown in Figure 5b. The first thing to notice is that in the red area the error is very high. This is due to the fact that here just noise is present, and with that the signal values are very small (range of approx. $\pm 2.5 \cdot 10^{-3}$), so that in this time range even small differences between the signals lead to a large *ARE*. At the end of the red area, the errors decrease so that the errors in the following (left) green area are quite small in comparison, but one can see that the error behavior in this area is very dynamic with a lot of maxima and minima for all models. The next thing one can observe is that for both training methods, the error in the yellow area increases, as has been seen in Figure 4. The error increases with the arrival of the A_0 mode and peaks at approx. the middle of the yellow area, which corresponds to the maximum amplitude of the A_0 mode. After that the error decreases again.

When comparing the different models with each other, one can see that the best results are given by the polynomial activation with the initial fit to the $cos(\pi * x + \pi/2)$ and $cos(\pi * x + \pi)$ function. But also the classic AE with the tanh activation gives good results.

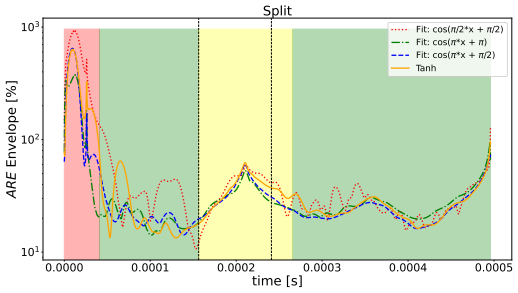
To investigate the errors further in Figure 6b the *ARE* of the signal phase is plotted. One can see that the error is again the biggest in the red area, which was expected. After a peak in the red area, the *ARE* decreases more or less constantly over the whole time range. Looking at the yellow area, one can see that the graphs of the phase *ARE* seem to change their slope close to the middle of the yellow area, which corresponds to the error peak in Figure 5 and the maximum of the A_0 mode. This behavior is more present in some of the models (e.g., the Figure 6b poly $cos(\pi * x + \pi)$ model) than in others (e.g., the AE model in Figure 6b). Thus, it can be concluded that the main sources of error of the transformation are in the regions where complex waveforms occur due to the increase of interference resulting from wave packet separation of the A_0 and S_0 modes. The high amplitudes of the errors are also due to the relatively low amplitude in these regions, which means that even small deviations for small amplitudes lead to a high *ARE*.

Comparing the parallel and split training approaches, one can see that the *ARE* are quite close to each other. Also, AREs are stable and quite low, so it seems that the transformation of the signal phase with all models is successful. The best results are given by the poly model with the $cos(\pi * x + \pi)$ fit.

To be able to better clarify which model produces the best transformation results, the *ARE* of the signal envelopes are averaged over time. However, in order to avoid a misleading picture, the red time range was omitted from the averaging process. The results are shown in Figure 7, where one can see the best results

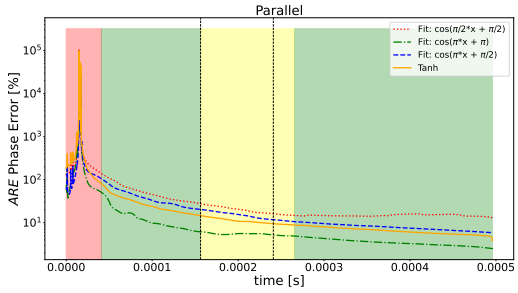


(a) Envelope error of parallel trained models in comparison with the classic AE model with tanh activation.

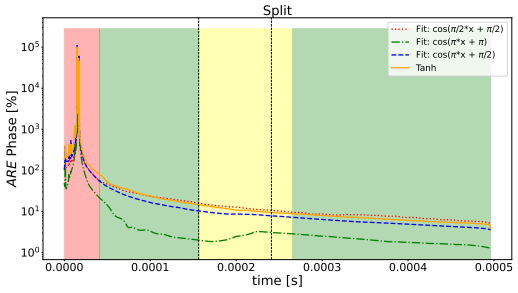


(b) Envelope error of split trained models in comparison with classic AE model with tanh activation.

Fig. 5. Signal envelope amplitude error over time. The red areas mark the time space in which just noise is present. The green areas mark the time when the transformation of the signal worked well. The yellow area, on the other hand, marks the time space where the transformation produced a large error, while the black lines in that area indicate the time when A_0 and S_0 modes are present at the same time.



(a) Phase error of parallel trained models in comparison with the classic AE model with tanh activation.



(b) Phase error of split trained models in comparison with the classic AE model with tanh activation.

Fig. 6. Signal Phase error over time. The red areas mark the time space in which just noise is present. The green areas mark the time when the transformation of the signal worked well. The yellow area, on the other hand, marks the time space where the transformation produced a large error, while the black lines in that area indicate the time when A_0 and S_0 modes are present at the same time.

are the polynomial activation with the initial fit to the $cos(\pi * x + \pi/2)$ and $cos(\pi * x + \pi)$ functions, so that it is shown the PANN approach can improve the transformation between ACM and SCM.

4. Summary and Conclusion

In the present work, the transformation between air-coupled and surface-coupled GUW measurements was investigated. In order to optimize the transformation, PANN was used with different initializations of the polynomial coefficients and model training approaches. To get the best results, hyperparameter tuning was performed. Due to the large number of parameters, the tuning was restricted to a maximum of four values per parameter. The results of the PANN models were then compared with a classic AE with tanh activations. It was shown that the PANN can optimize the transformation process. Another finding of this study is that the transformation error increases at points where interference in the wave signal is present. This interference led to phase changes and complex waveforms. But this shows that the transformation can be improved using wave signals without strong interference effects. Also, the hyperparameter tuning of the PANN could be improved in order to optimize the transformation. So that in future work on these approaches will be continued.

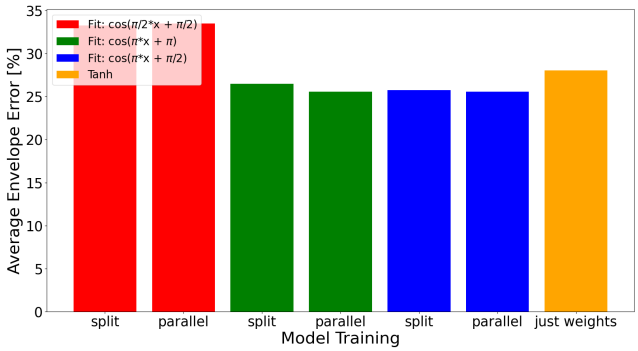


Fig. 7. Average *ARE* of the signal envelope.

Acknowledgements

The authors expressly acknowledge the financial support of the research work on this article within the Research Unit 3022 “Ultrasonic Monitoring of Fibre Metal Laminates Using Integrated Sensors” (Project number: 418311604) by the German Research Foundation (Deutsche Forschungsgemeinschaft (DFG)).

erences

References

[1] H. Lamb, On waves in an elastic plate, Proc. Roy. Soc. London Ser. A 93 (21) (1917) 114–128. doi:10.1098/rspa.1917.0008.

[2] V. Giurgiutiu, Structural Health Monitoring with Piezoelectric Wafer Active Sensors, 2014. doi:10.1016/C2013-0-00155-7.

[3] P. Cawley, D. Alleyne, The use of lamb waves for the long range inspection of large structures, Ultrasonics 34 (2) (1996) 287–290, proceedings of Ultrasonics International 1995. doi:https://doi.org/10.1016/0041-624X(96)00024-8.
URL https://www.sciencedirect.com/science/article/pii/0041624X96000248

[4] J. L. Rose, Ultrasonic guided waves in structural health monitoring 270 (2004) 14–21. doi:10.4028/www.scientific.net/KEM.270-273.14.

[5] F. Lanza di Scalea, S. Salamone, Temperature effects in ultrasonic lamb wave structural health monitoring systems, The Journal of the Acoustical Society of America 124 (1) (2008) 161–174. arXiv:https://doi.org/10.1121/1.2932071, doi:10.1121/1.2932071.
URL https://doi.org/10.1121/1.2932071

[6] K. J. Schubert, C. Brauner, A. S. Herrmann, Non-damage-related influences on lamb wave–based structural health monitoring of carbon fiber–reinforced plastic structures, Structural Health Monitoring 13 (2) (2014) 158–176. arXiv:https://doi.org/10.1177/1475921713513975, doi:10.1177/1475921713513975.
URL https://doi.org/10.1177/1475921713513975

[7] R. Soman, P. Kudela, K. Balasubramaniam, S. K. Singh, P. Malinowski, A study of sensor placement optimization problem for guided wave-based damage detection, Sensors 19 (8). doi:10.3390/s19081856.
URL https://www.mdpi.com/1424-8220/19/8/1856

[8] R. Soman, Multi-objective optimization for joint actuator and sensor placement for guided waves based structural health monitoring using fibre bragg grating sensors, Ultrasonics 119 (2022) 106605. doi:https://doi.org/10.1016/j.ultras.2021.106605.
URL https://www.sciencedirect.com/science/article/pii/S0041624X21002249

[9] Z. Yang, H. Yang, T. Tian, D. Deng, M. Hu, J. Ma, D. Gao, J. Zhang, S. Ma, L. Yang, H. Xu, Z. Wu, A review on guided-ultrasonic-wave-based structural health monitoring: From fundamental theory to machine learning techniques, Ultrasonics 133 (2023) 107014. doi:https://doi.org/10.1016/j.ultras.2023.107014.
URL https://www.sciencedirect.com/science/article/pii/S0041624X23000902

[10] A. De Fenza, A. Sorrentino, P. Vitiello, Application of artificial neural networks and probability ellipse methods for damage detection using lamb waves, Composite Structures 133 (2015) 390–403. doi:https://doi.org/10.1016/j.compstruct.2015.07.089.
URL https://www.sciencedirect.com/science/article/pii/S0263822315006352

[11] C. Sbarufatti, G. Manson, K. Worden, A numerically-enhanced machine learning approach to damage diagnosis using a lamb wave sensing network, Journal of Sound and Vibration 333 (19) (2014) 4499–4525. doi:https://doi.org/10.1016/j.

jsv.2014.04.059.
URL https://www.sciencedirect.com/science/article/pii/S0022460X14003459

[12] M. Rautela, S. Gopalakrishnan, Ultrasonic guided wave based structural damage detection and localization using model assisted convolutional and recurrent neural networks, Expert Systems with Applications 167 (2021) 114189. doi:https://doi.org/10.1016/j.eswa.2020.114189.
URL https://www.sciencedirect.com/science/article/pii/S0957417420309234

[13] C. Polle, S. Bosse, A. S. Herrmann, Damage location determination with data augmentation of guided ultrasonic wave features and explainable neural network approach for integrated sensor systems, Computers 13 (2). doi:10.3390/computers13020032.
URL https://www.mdpi.com/2073-431X/13/2/32

[14] S. Bosse, Data-driven parameterizable generative adversarial networks for synthetic data augmentation of guided ultrasonic wave sensor signals, in: Proceedings of the 11th European Workshop on Structural Health Monitoring (EWSHM 2024), Potsdam, Germany, 2024, 10–13 June 2024.

[15] C. Polle, S. Bosse, D. May, Transformation of guided ultrasonic wave signals from air coupled to surface bounded measurement systems with machine learning algorithms for training data augmentation, in: Proceedings of the 11th International Electronic Conference on Sensors and Applications, MDPI, Basel, Switzerland, 2024, 26–28 November 2024. doi:10.3390/ecsa-11-20448.

[16] J. Zhou, H. Qian, X. Lu, Z. Duan, H. Huang, Z. Shao, Polynomial activation neural networks: Modeling, stability analysis and coverage bp-training, Neurocomputing 359 (2019) 227–240. doi:https://doi.org/10.1016/j.neucom.2019.06.004.
URL https://www.sciencedirect.com/science/article/pii/S0925231219308252

[17] Knowles, Data sheet spu0410lr5h-qb, https://www.knowles.com/docs/default-source/model-downloads/spu0410lr5h-qb-revh32421a731dff6ddbb37cff0000940c19.pdf?Status=Master&sfvrsn=cebd77b1_4, available online: https://www.knowles.com/docs/default-source/model-downloads/spu0410lr5h-qb-revh32421a731dff6ddbb37cff0000940c19.pdf?Status=Master&sfvrsn=cebd77b1_4 (accessed on 14 October 2024) (2024).

[18] A. M. A. Huber, M. G. R. Sause, Dispersion calculator rev. 3.0, available at https://github.com/ArminHuber/Dispersion-Calculator/releases/tag/v3.0 (Last viewed 28 May 2025).