Data-driven Parameterizable Generative Adversarial Networks for Synthetic Data Augmentation of Guided Ultrasonic Wave Sensor Signals

Stefan Bosse[1,2]
[1]University of Bremen, Dept. Mathematics & Computer Science, Bremen, Germany
[2]University of Siegen, Dept. Mechanical Engineering, Siegen, Germany

Abstract. Synthetic data generation and augmentation is often the only solution to create data-driven robust and generalized damage predictor models. Ideally, the predictor model can be trained with synthetic data only, finally applied to real measured data. In this work, we present and evaluate different synthetic data generation methodologies using model-free data-driven generator models like Generative Adversarial networks (GAN) using model-based analytical functions, physical simulation, and experimental data. We will demonstrate the challenges and pitfalls of synthetic data generation using Machine Learning for the generation of Guided Ultrasonic Waves (GUW) time signals.

1. Introduction

Detecting and characterizing hidden damages in composite materials like Fibre-Metal Laminates (FML) remains a challenge. Guided Ultrasonic Waves (GUW) or X-ray imaging are commonly used to detect these damages, but their interpretation remains limited, in applications like Non-destructive Testing (NDT) and Structural Health Monitoring (SHM) as well. Data-driven predictor models can detect damages in structures using GUW time-dependent signals, but the experimental training data lacks variance, statistical strength, and sufficient coverage of the hyper-parameter space. Often experimental data lacks ground truth annotations of target parameters. Synthetic data augmentation is often the only solution to create robust and generalized damage predictor models. Synthetic sensor data can be generated using model-based, model-assisted, or model-free methods. However, numerically computed GUW signals by applying Finite-Element Methods or by solving field equations show poor reality conformance due to too much constraints and simplifications, especially in non-homogeneous materials, composites, and laminates. Recent developments in data-driven generative models, e.g, Generative Adversarial (Neural) Networks (GAN) [1], commonly driven by a random generation process, include deterministic style vectors to generate specific signal data [2], determining constraints such as damage size, position, transducer positions, material and environmental properties. These new architectures aim to reduce the impact of environmental changes on data-driven damage predictor models by using parameterizable synthetic data generation. At least in theory. In practice, these controllable GAN architectures must be validated to ensure sufficient reproduction of signal features related to damages and defects, commonly a small fraction of an entire signal. In [1]. the authors claim to create B-scan wave images in accordance with experiments using pseudo defects. In [3], a GAN is composed of bi-directional state-based recurrent LSTM cell networks to generate ECG signals. Both applications differ from this work because here the relevant damage feature are hidden in small temporal and amplitude fractions compared with the base-line signal making training of generative models a challenge. The generation of GUW signals comparable to this work was reported in [4] using a styled GAN and experimental data from OpenGuidedWaves data set. This data set contains measurements with pseudo defects and only 8 different defect positions. the authors claim to generate GUW signals in accordance with real data by controlling the generation process. The shown signals show artifacts which are observed by our experiments, too, and considered as critical for training ML predictor models.

We will elaborate these new architectures and discuss the possibilities and challenges of using such GAN models for data generation of US signals in GUW-based SHM applications, as originally proposed in [1]. We will have a focus on simplified measuring data used for training of data-driven predictor models addressing homogeneous and composite materials (e.g., but not limited to, FML) with hidden damages, and show some preliminary results and fundamental discussions. This paper is a review as well as experimental elaboration.

We will conduct and evaluate simple experiments with pitch (generator) and catch (sensor) time-resolved GUW signals, commonly used in NDT and SHM applications. The pitch signals are created with an analytical model, the catch signals are computed by a 2-dim simulation using a visco-elastic wave propagation model and a simulator (based on SimNDT [5]) using a elastodynamic finite integration technique. We present a model zoo that is in principle suitable for signal generation, discussing the pro and cons of the models using experiments or more theoretical considerations.

2. Methodology and Goals

The basic idea is to train data-driven classification and regression models suitable for NDT and SHM applications by using synthetic data only, finally applying the models to real data without loss of accuracy and quality, perhaps by fine-tuning the models. The overall taxonomy of the methodology for synthetic data generation addressed in this work is shown in Figure 1. The synthetic data could be generated in simulation, but due to restrictions and the reality gap this can be an error prone task. Generative models can be trained by using real, real and simulated, or simulated data. The generative models should be controllable by a parameter vector optimally with independent model parameters, like in any conventional signal generator. Additionally, the generations process should be randomize with respect to Monte Carlo simulation to created sufficient variance in the generated signals.

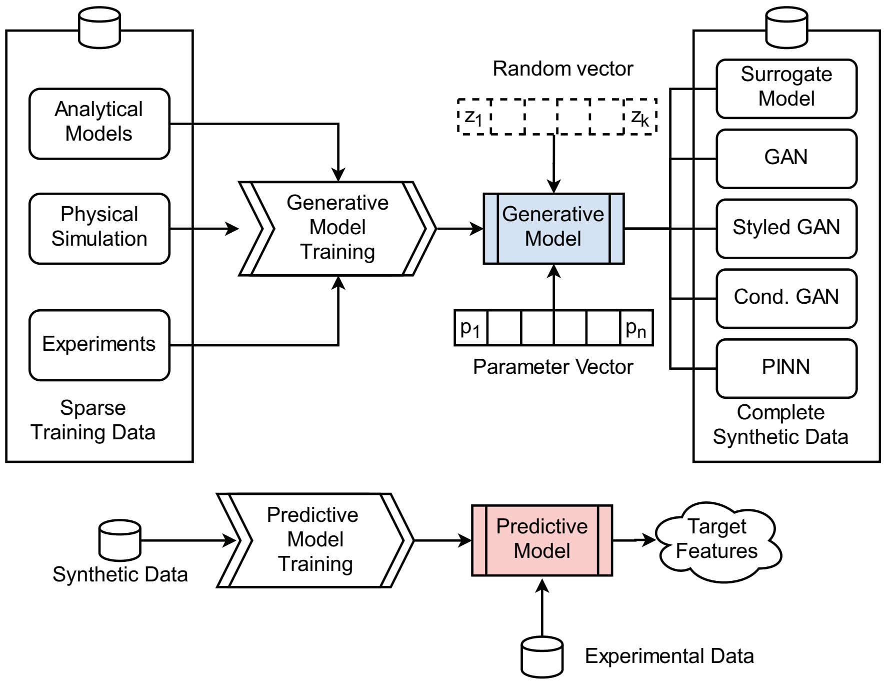


Figure 1. Taxonomy of the methodology for synthetic data generation addressed in this work. Data from a controllable but randomized generator model is used to train a predictor model. Different architectures are investigated in this work.

The assessment of the quality of the synthetic signal generator is critical and a challenge. In many works, distance measure functions are used or correlation analysis. Both methods can only applied to signals with strong features, i.e., the modifications of a signal with respect to an anomaly, damage, or defect of the material to be examined. GUW signals used in NDT and SHM applications show typically only weak features (response) with respect to structural or material deviations. One possible outcome is the application of a feature predictor model trained with experimental or simulated data to the synthetic data to validate synthetic signals. This is the approach we have chosen in this work.

3. Models

The generative models considered in this work should be used to augment the sparse experimental data set of GUW time-resolved signals. There are three classes of generative models:

1. Surrogate models without a random latent input space (pure functional) controlled by a mostly fully linearly independent and separated (not entangled) parameter space (i.e., independent control of specific parameters). The surrogate generative model is trained directly with examples in a supervised way.

2. Pure random generative models that cannot be controlled by a parameter vector (e.g., damage location). These models are trained indirectly by another assessment instance and never see the original examples.
3. A hybrid between class 1 and 2, controlling the signal generation by parameter vector, which is commonly entangled not allowing fully independent control of the signal generation.

Figure 2 shows three different model architectures that are investigated and evaluated in this work. More advanced architectures are shown in Figure 3.

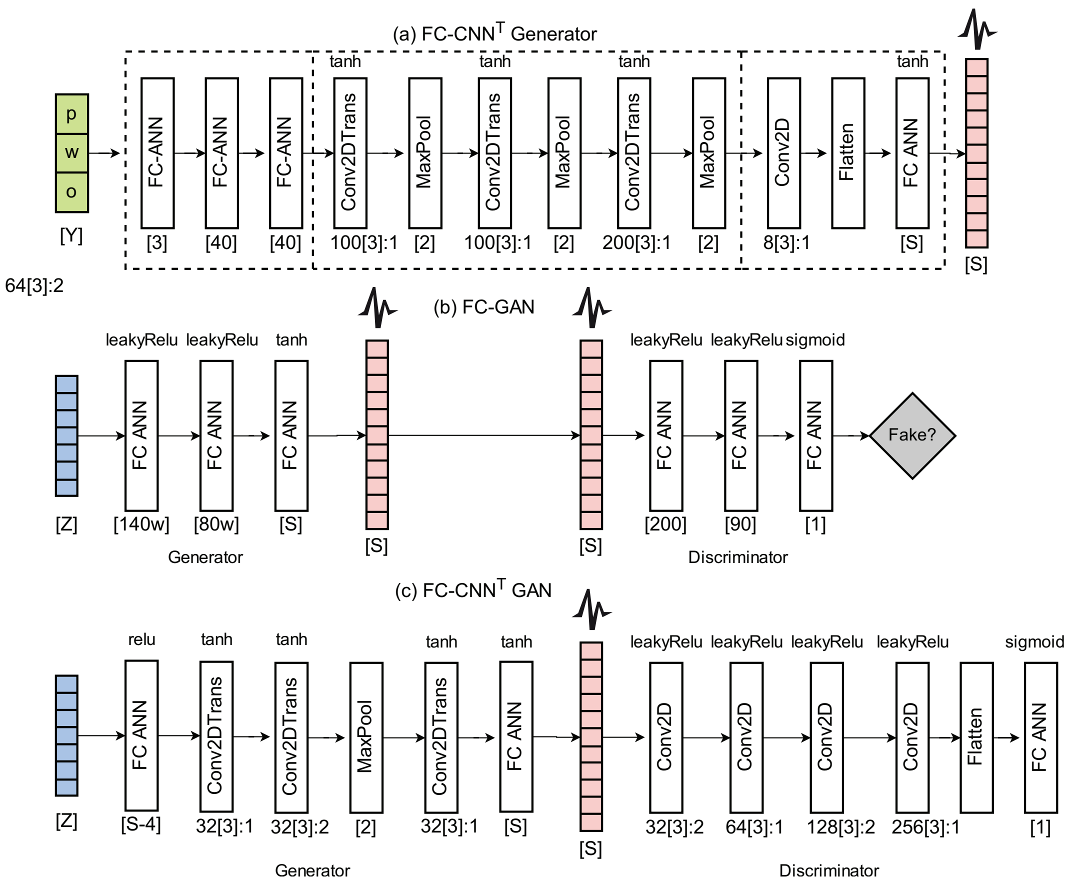


Figure 2. Part one of the model zoo (a) Parameterizable surrogate generator model using a CNN (b) Simple FC-ANN-based GAN (c) More complex ANN-CNN-based GAN. Annotations: Fully Connected (FC) layer [Number of neurons], Convolutional layer filters[kernel size]:striding, Top layer annotation: Activation function

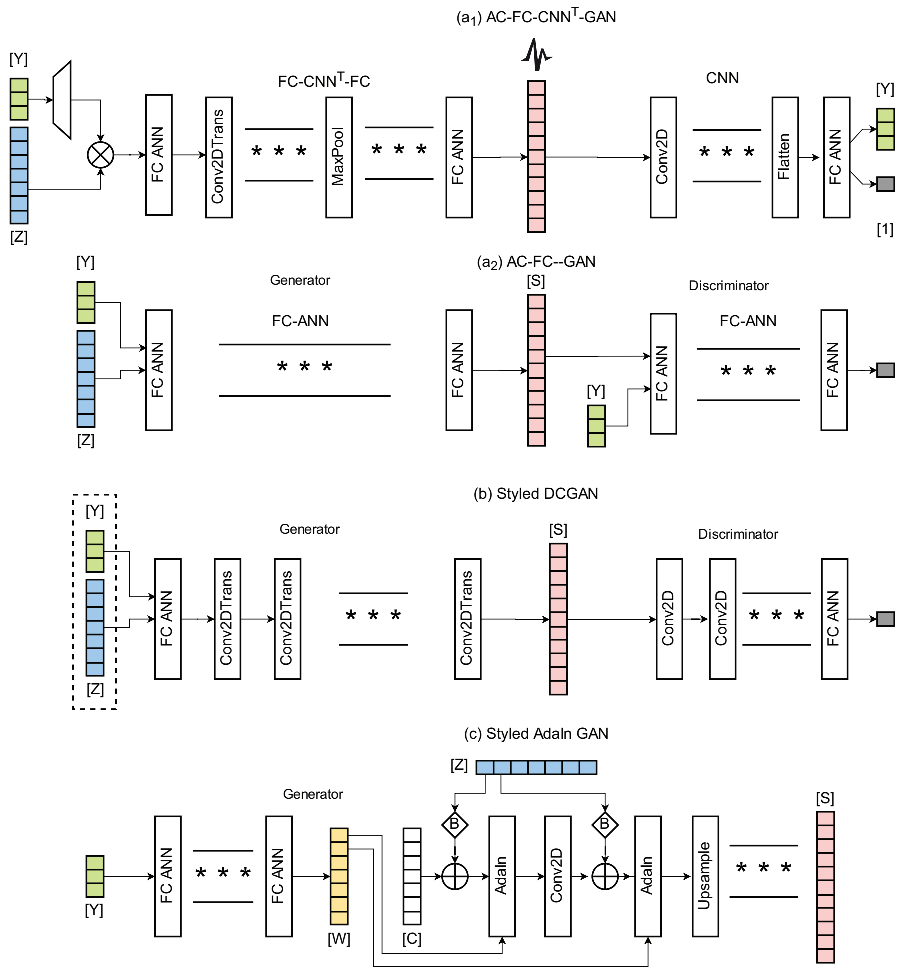


Figure 3. Part two of the model zoo (a) Auxiliary Conditional GAN with an additional parameter vector y which is input of generator and either output (top) or input (bottom) of discriminator (b) Styled Deep Convolutional GAN (c) Styled Adaptive Instance Normalization GAN

3.1 FC-ANN

The simplest generator architecture consists of a set of fully connected ANN layers. The input of the FC-ANN is the parameter vector (e.g., damage position or frequency), and the output is the signal vector (index of the vector represents the time axis). Pure ANN-based generator model are limited, e.g., with respect to a time-dependent feature in the signal or the phase of the entire signal. For time series, state-based recurrent ANN can be used, as discussed in Sec. 3.6. We were not able to generate meaningful pitch or catch signals with a pure ANN model.

3.2 CNN

A Convolutional Neural Network (CNN) consists of basically linear kernel-based matrix transformations (convolution C) in combination with linear or non-linear transfer functions, finally combined with Fully Connected (FC) Artificial Neuronal Networks (ANN). A CNN is a sandwich structure of alternating layers (C, Pooling, FC-NN commonly at the end). A CNN can extract features from any kind of ordered signals (time-resolved signals, frequency spectra, images). It can be used for classification as well as for regression tasks. We use a classical CNN later as a feature predictor model applied to raw GUW sensor signals. A typical convolution operation compress the input vector towards a reduced feature vector (e.g., a damage position), i.e., performing a down-sampling. For generative models we need the opposite, an up-sampling, and we want to expand a feature input vector to a signal vector (or image). For this purpose, transposed convolutional transformations (C^T) can be used [6]. An example of a generative pure FC-CNN^T is shown in Figure 2 (a). It is used to generate GUW signals fully trained by examples of real or simulated signals. The head consist of a FC-ANN block, followed by C^T-Pooling layer pairs, finally terminated by another FC-ANN block. CNNs pose translation, rotation, and scaling invariance (in contrast to pure FC-ANNs).

3.3 GAN

A Generative Adversarial Network (GAN) consists of two models: A generator, commonly driven by random process, and a discriminator, commonly with a single output. In contrast to a surrogate model, the generator never sees original examples. It is controlled during training by the discriminator model. There is commonly a combined model coupling the generator and discriminator in a chain, finally allowing the feedback of the discriminator that classifies a signal in real and fake (synthetic). The discriminator is commonly the only model block that sees real examples. During the training, the discriminator gets a set of real and fake images from the generator classifying real versa fake images. The discrimination error is used to adjust the parameters of both models. The loss of each sub-model is handled differently. Commonly during training, the loss of the discriminator decreases, whereas the loss of the generator increases. There is a permanent competition in the training of both models. Finally, only the generator model is used to train synthetic signals or images (or any kind of data).

The input of the generator is initially a pure random vector, e.g., with 100 elements, each set by a uniform random generator in the range of [-1,1]. There is no control over the output, in contrast to surrogate model with a deterministic control input vector. The generator as well the discriminator models can be implement by FC-ANN or CNN^T/CNN architectures, as shown in Figure 2 (b)-(c).

3.4 AC-GAN

The auxiliary conditional GAN [7] is an attempt to control the generative model by an additional conditional parameter vector. Originally used to distinguish different generation classes (e.g., different image classes) via a one-hot coded control vector, it is assumed that this model can be used with continuous parameter control, too. The basic architectures are shown in Figure 3. The AC-GAN can be considered as a transitional network architecture between a classical GAN and a styled GAN with a much more woven interconnect. There are basically two different architecture variations with respect to the control parameter vector either passed to the input of the discriminator or output by the discriminator model (a-1 and a-2, respectively).

3.5 Styled GAN

There are basically two different controllable GAN model architectures:

1. DC-GAN: Deep Convolutional GAN [8,9] using convolutional layers only (without FC or pooling layers) that map a latent space vector (e.g., N=100) with a uniform random distribution on a feature space. An example application is a glyph generator [9]. The principle architecture is shown in Figure b. The random latent space vector Z is merged with a style vector controlling the generation process by a FC layer (here a one-hot vector of the character code).
2. ADAIN-GAN: Adaptive Instance Normalization GAN [2] splitting the generator into a mapping and synthesis network. The mapping network (multi-layer FC ANN) maps the style vector Z on an intermediate latent vector w, which is the input for multiple adaptive instance normalization layers controlling the generation process. The uniform random distributed vector (Gaussian noise) is added partially to the output of multiple convolutional layers, instead passing the entire random vector to the input of the first convolutional layer, which shown in Figure c.

Only few information is available about the detailed structure of the parameter vector used to control the generator. Only [9] (character generator) gave details and correlated results. The degree of entanglement of the generator control parameters is unclear. [4] used the OpenGuidedWaves data set to train a styled GAN. The data set provides only 12 transducer and 7 (24) pseudo defect positions. Although , they use style control, the authors only state that "the guided wave signal synthesis appears to perform well at every step, producing believable signals via random generation". A data-driven validation was performed using a classifier that is fed with synthetic data, but a rigorous analysis and validation is missing.

3.6 LSTM-based Models

All previous models were pure functional, i.e., the output of the network only depends on the current input. Time series signals pose a history, i.e., a wave propagation at time t_i depends on the wave at past time points t_{i-1}. Therefore, state-based recurrent ANN networks can provide a history memory aiming to generate time series signals. In [10], a bi-directional Long Short-term Memory (LSTM) cell network was used to generate ECG signals. LSTM networks are suitable to generate short duration fundamental waves with strong features (e.g., modifications of signals due to damages or anomaly), but cannot synthesize long signal structures with weak features, which is the case in GUW signals, especially with reflection interference and long oscillations.

3.7 Simulation

To provide ground truth data for our generative model evaluations we compute data by a 2-dim simulation using a visco-elastic wave propagation model and a simulation framework (based on SimNDT [5], extended version 2 [11]) using an elastodynamic finite integration technique. The simulation set-up and some computed sensor signals are shown in Figure 4. The sensor signals are the x-axis component of the *T* field (tension). It is a string simplification of any real sensor model based on the piezo-electric effect. But for the purpose of this work these simplified signals should be sufficient. We used an aluminum plate of size 500 × 500 mm, a sending transducer placed at (250,300) mm (10 mm diameter) and a line array of 20 receiving transducers (point sensor) at the bottom at y=450 mm. The damage was a circular air hole with a diameter of 30 mm, which was placed on position grid (50 mm distance between two grid points). The pitch signal was a sine wave of base frequency 40 (80) kHz and a Gaussian mask window (5 cycles). The simulation was carried out with a time step of 0.06 µs, in total 5000 steps (300 µs), with each tenth step recorded. In total 7 × 6 damage positions were simulated.

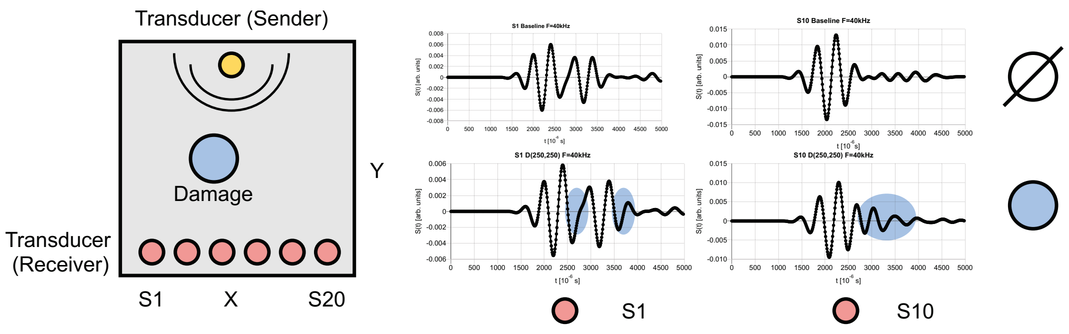


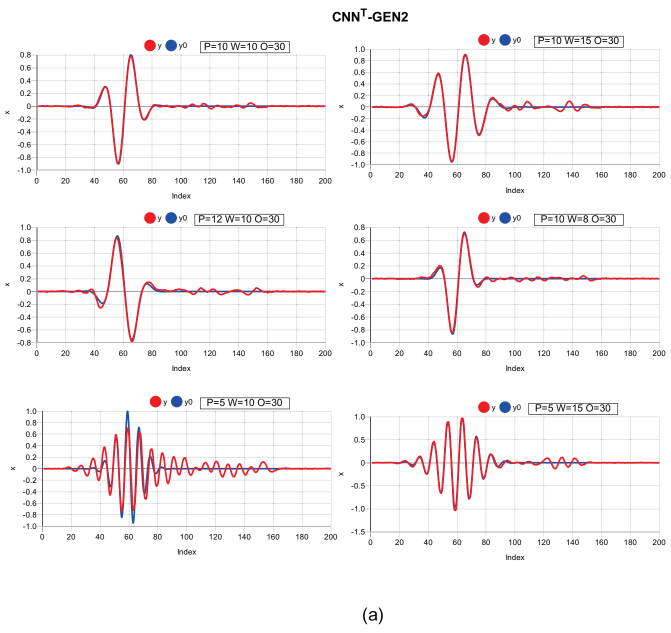
Figure 4. GUW signal simulation using a 2-dim viscoelastic wave propagation model. (Left) Simulation set-up (Right) Some example signals with and without damage (blue areas show damage features)

4. Results

For the sake of simplicity and causality we use a GUW pitch signal as a starting candidate to investigate the qualitative and quantitative precision of different generator architectures. A GUW pitch (generator) signal is commonly a masked sine wave, e.g., using a Gaussian mask window. Such a signal is characterized by three parameters: The period time of the base sine wave (*P*), the width of the mask window *W*, and an optional offset of the burst (window) *O*. The analytical signal generator model is defined in Eq. 1.

$$\begin{gathered}\vec{I}=(1,2,..,N)\\ \vec{S}=\sin\left(\frac{\vec{I}}{P}2\pi\right)\\ \sigma=kW\\ \vec{J}=\frac{\vec{I}-O}{0.5N}2\sigma\\ \vec{M}=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{\vec{J}^2}{2\sigma^2}}\\ \vec{G}=\vec{S}\vec{M}\end{gathered}\tag{1}$$

In the first experiment a simple and well known FC-CNN$^{\mathrm{T}}$ generator model is used, which was trained with a training data set created by the analytical signal function with additional Gaussian noise. Input is the parameter vector $[P, W, O]$, output the signal (N=200 points). The network architecture is given in Appendix A.



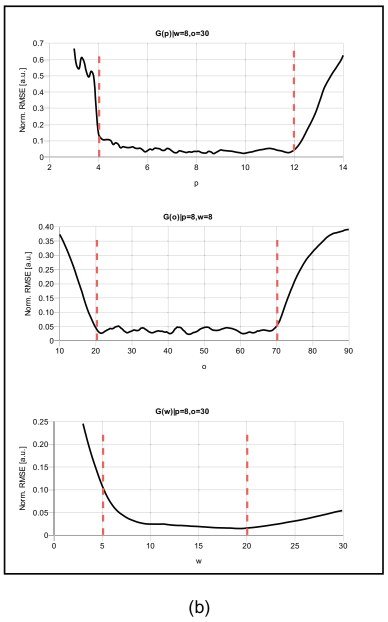


Figure 5. (a) Examples of pitch signals generated by the parametrized FC-CNN^T model. Parameter vector [P,O,W]=[Period, Offset, Width] (b) Normalized RMSE of the generator (compared with analytical model) by varying one parameter. The red lines show the training parameter ranges.

Figure 5 (a) shows examples of signals generated by the FC-CNN^T model. Within the trained parameter range the reproduced signals show a high conformance with the signals generated by the analytical model, with some additional ripple before or after the main signal. Outside the training parameter space the generation increases significantly as shown in Figure 5 (b). In principle, a CNN-based generator model (with non-linear activation functions) is capable to reproduce complex signals based on a sine wave multiplied with a Gaussian function. Some ripple is added before or after the main wave burst. But the signal synthesizer provides no benefits over the analytical function because it utilizes no randomization, which is typical in real measurements.

The next experiment used a FC-CNN^T-GAN model to create randomly generated pitch signals. Again the GAN model was trained with the same training data set created by the analytical signal function with overlayed multiplicative and additive Gaussian noise. Some example signals are shown in Figure 6. Signals with green background are characterized as valid signals that conform with signals from the analytical model. Signals with a yellow background shown artifacts and distortion, and signals with a red background do not conform with the analytical model.

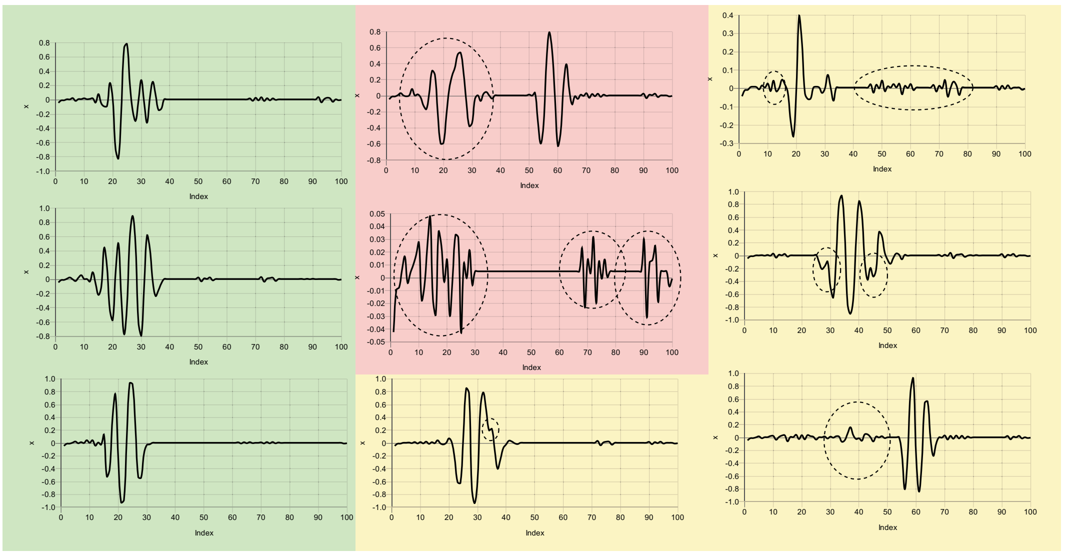


Figure 6. Examples of pitch signals generated by the pure random FC-CNN^T-GAN model.

Figure 7 shows the histogram distribution of the standard deviation of the generated signal for the analytical model, the FC-CNN^T, and FC-CNN^T-GAN models. The standard deviation is a common GUW signal feature used in SHM applications to identify anomalies and damages. The standard deviation of the signals from the GAN model is strongly different from that from the analytical and CNN models. Remarkable is a constant median of about 0.05 for all signals generated by the FC-CNN^T-GAN model. The analytical model produces a slightly varying median around 0 depending on the parameter settings.

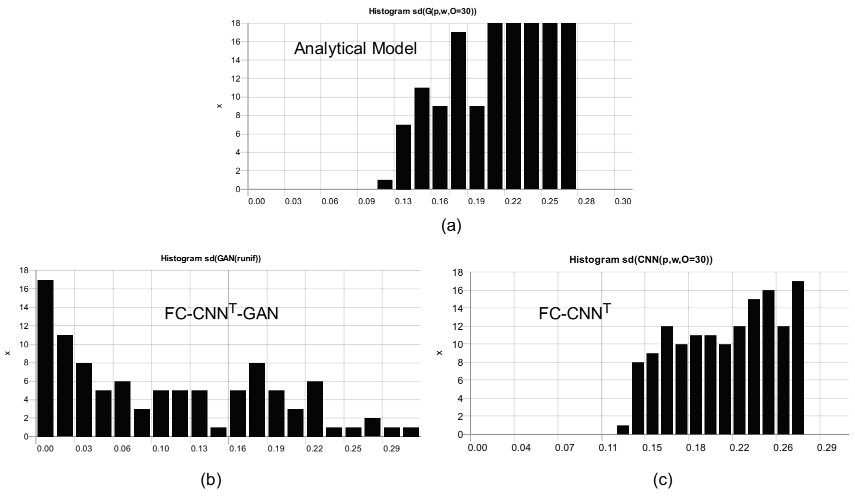


Figure 7. Comparison of the histogram distribution of the standard deviation of pitch signals generated by (a) the analytical, (b) the random controlled FC-CNN^T-GAN model trained with the data from the analytical model, and (c) the surrogate FC-CNN^T model. All histograms are in the range [0,0.3].

In preparation for the experiments with experimental (simulation) catch signal data, a predictor regression model must be created. The previous experiments relied on an already existing ground truth analytical model. But there are no accurate and complete models for real GUW signals. Therefore, to assess the output of generator models, a forward predictor model is required that outputs the respective parameters, e.g., the 2-dim location of a damage or the sensor position. The predictor model uses a classical CNN regression model. Input is one raw reshaped sensor signal (200 data points) at a specific sensor position $S(x,y)$. Output is the estimated parameter set. We used two independent models (with the same architecture) for the prediction of the position $D(x,y)$ of a damage and the sensor position S. The damage coordinates were normalized using the scale 1:500 ([0,1]) and 1:25 ([0,1]) for the sensor index position along the sensor line array as shown in Figure 4. The model was trained with the data set from the simulation (42 damage positions) at a fixed frequency. The entire data set consists of all independent sensor signals (20) and the 42 damage positions, and a base-line damage-free case. All signals were augmented by Gaussian noise (multiplicative and additive), and finally scaled randomly. The training of the models typically converge after 100 epochs with a low learning rate of l=0.002 using an ADAM optimizer. The model architecture is shown in Appendix A.

In Figure 8 prediction results of the damage position using simulated GUW signals are shown. Estimating the damage position using only one sensor signal is challenge and not possible with classical analytical analysis methods. The CNN predictor is able to predict most positions accurately with a location error below 20%. The average position values are accumulated by independent predictions from all sensors.

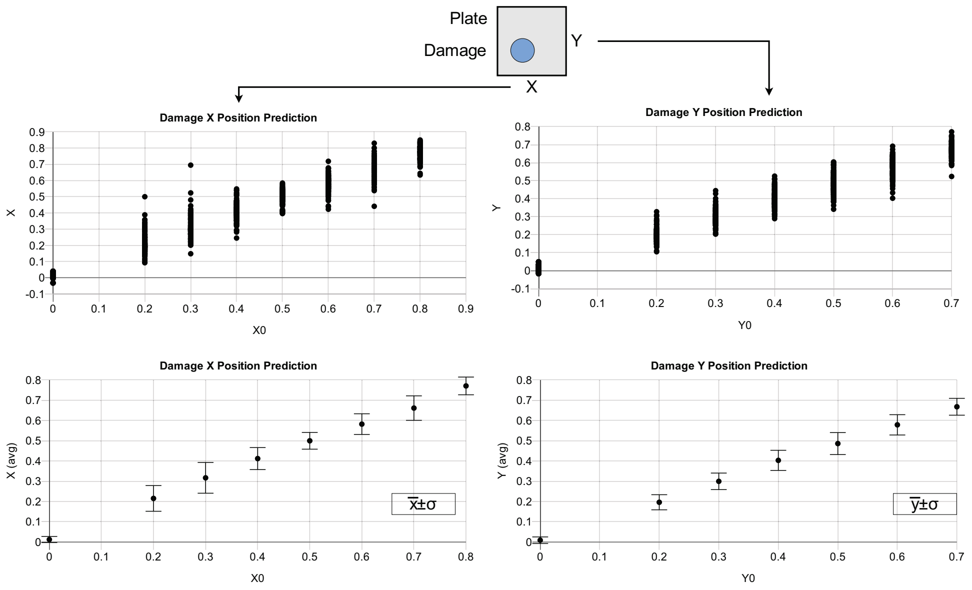


Figure 8. Results of a damage location predictor model. Input is a real (simulated) GUW signal, output are the x and y coordinate of the damage (0: no damage). (Top) Scatter plots (Bottom) average and error plots (σ interval). The 0-labeled coordinates are the ground truth coordinates. All coordinates are normalized with a 1:500 mm scale.

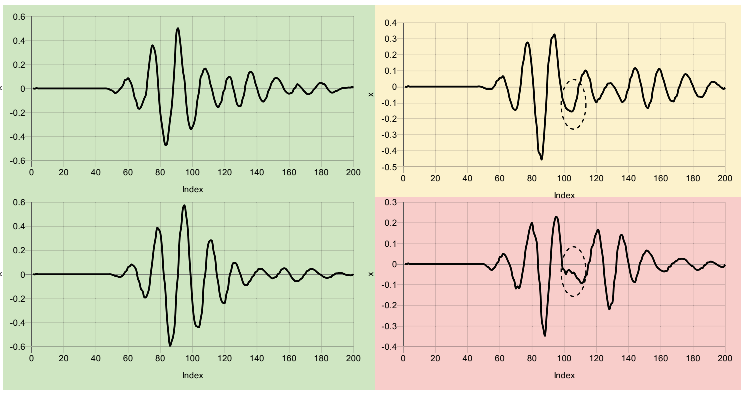


Figure 9. Examples of good and bad GUW signals generated by the pure random process controlled FC-CNN^T GAN model

In Figure 10 analysis results from the CNN predictor models applied to the FC-CNN^T GAN model are shown (assessment of the GAN model). The distribution of the standard deviation of the generated signals is now in accordance to the standard deviation of the original (simulated) signals. The distribution of the predicted X- and Y-coordinates from a random set of generated signals covers roughly the training range (of the simulated signals). The distribution of the sensor position predictions shows a high specialization of the GAN model to the center sensor. The training set were equally distributed with respect to the sensor position ([0.15,0.8], the damage positions [0.2,0.8]. The damage free base-line was augmented to about 1/3 of the entire set of damage examples. The GAN was trained with a binary crossentropy loss function (200 epochs with a learning rate of 0.001). The model architecture is shown in Appendix A.

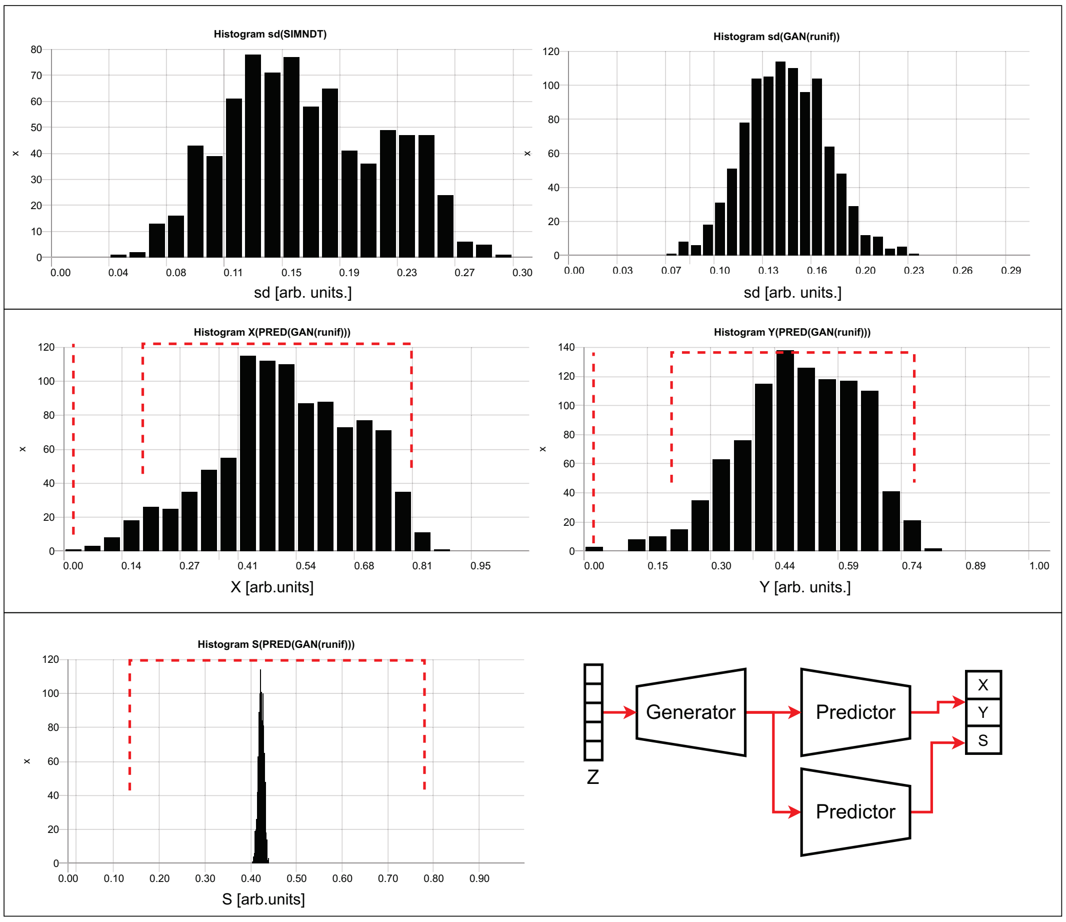


Figure 10. Analysis results from the CNN predictor models applied to the FC-CNN^T GAN model. (Top) Distribution of the standard deviation of the original and the generated signals (Middle) Distribution of the X- and Y-coordinate prediction from a random set of generated signals (Bottom) Distribution of the sensor position prediction. The red lines indicate the training ranges.

5. Discussion and Conclusions

The simple demonstrations showed the principle suitability of surrogate FC-CNN and randomly controlled GAN architecture to generate GUW signals. But they also showed their limitations, e.g., by producing artifacts like ripple or distorted signals, which would not be observed in real data. Generation is only possible within the training parameter space (interpolation), 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). There are doubts that results reported by, e.g., [3,10,4], are valid with respect to the benefit of training data-driven models.

In this work we showed results from fully controllable surrogate generators or not controllable pure randomized GAN models. We will continue to investigate controllable GAN architectures, and the AC-GAN model seems to be the most promising architecture.

6. References

[1] Virupakshappa, K. Oruklu, E., *Using Generative Adversarial Networks to Generate Ultrasonic Signals*, In 2020 IEEE International Ultrasonics Symposium (IUS) (pp. 1-3). IEEE, 2020

[2] Karras, T., Laine, S. and Aila, T., *A style-based generator architecture for generative adversarial networks*, Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019

[3] Hazra, D., Byun, Y.-C., *SynSigGAN: Generative Adversarial Networks for Synthetic Biomedical Signal Generation*, biology, vol. 9, no. 441, 2020

[4] Heesch, M., Mendrok, K., Dworakowski, Z., *Time-Domain Signal Synthesis with Style-Based Generative Adversarial Networks Applied to Guided Waves*, in ICAISC 2021, LNAI 12854, 2021

[5] Molero-Armenta, M., Iturrarán-Viveros, U., Aparicio, S., Hernández, M.G., *Optimized OpenCL implementation of the Elastodynamic Finite Integration Technique for viscoelastic media*, Comput Phys Comm 185 (2014) 2683-2696

[6] Gao, H., Yuan, H., Wang, Z., Ji, S., *Pixel transposed convolutional networks*, IEEE transactions on pattern analysis and machine intelligence, 42(5), 1218-1227, 2019

[7] Mirza, M., Osindero, S., *Conditional Generative Adversarial Nets*, arXiv, vol. 1411.1784v, 2014

[8] Radford, A., Metz, L., Chintala, S., *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*, arXiv:1511.06434 [cs.LG], 2016

[9] Hayashia, H., Abe, K., Uchida, S., *GlyphGAN: Style-Consistent Font Generation Based on Generative Adversarial Networks*, arXiv:, 2019

[10] Zhu, F., Ye, F., Fu, Y , Liu, Q., Shen, B., *Electrocardiogram generation with a bidirectional LSTM-CNN generative adversarial network*, Nature, vol. 9, no. 6734, 2019

[11] Bosse, S., SimNDT2, Revision of the GUW signal simulator SimNDT, http://git.edu-9.de/sbosse/SimNDT2, accessed on-line, 20.4.2024

[12] Tensorflow, https://github.com/tensorflow/tfjs, accessed on-line, 20.4.2024

7. Appendix: Model Architectures

All models were implemented with the JavaScript Tensorflow software framework, either fully implemented in JavaScript (`tensorflow.js`), or wrapped around the native Tensorflow API (`@tensorflow/tfjs-node` or `@tensorflow/tfjs-node-gpu`) [12].

Nomenclature:

- Dense Layer: `dense units (activation function)`
- Convolution: `conv2d filters [kernel size]:striding`
- Convolution transposed: `conv2dT filters [kernel size]:striding`
- If the input shape of a convolutional layer is a line tensor such as $[n,1,f]$, the kernel has the shape $[k,1]$.

7.1 FC-CNN^{T} Generator Model Architecture

- Input: Signal parameter vector $[P, W, O]$ (period, width, offset of signal).
- Output: Time-resolved Signal

```
Layer (type)                        Output shape       Param #
==============================================================
dense 3   (linear)                 [3]                12
______________________________________________________________
dense 40  (linear)                 [40]               160
______________________________________________________________
dense 40  (linear)                 [40]               1640
```

```
reshape                             [40,1,1]           0

conv2dT  100[3]:1 (tanh)            [42,1,100]         400

max_pooling2d                       [21,1,100]         0

conv2dT  100[3]:1 (tanh)            [23,1,100]         30100

max_pooling2d                       [11,1,100]         0

conv2dT 200[3]:1 (tanh)             [13,1,200]         60200

max_pooling2d                       [6,1,200]          0

conv2d   8[3]                       [4,1,8]            4808

flatten                             [32]               0

dense 200 (linear)                  [200]              6600
=============================================================
Total params: 103920
Trainable params: 103920
Non-trainable params: 0
```

7.2 FC-CNN Predictor Model Architecture

- Input: Time-resolved sensor signal
- Output: Parameter set $[X, Y]$ or $[S]$ (position of sensor)

```
Layer (type)             Input Shape      Output shape      Param #
====================================================================
reshape                  [200]             [200,1,1]         0

conv2d 64[5]:1           [200,1,1]        [196,1,64]        384

max_pooling2d            [196,1,64]       [98,1,64]         0

conv2d  32[5]:1          [98,1,64]        [94,1,32]         10272

max_pooling2d            [94,1,32]        [47,1,32]         0

conv2d  32[3]:1          [47,1,32]        [45,1,32]         3104

max_pooling2d            [45,1,32]        [22,1,32]         0

flatten                  [22,1,32]        [704]             0

dense 40 (tanh)          [704]            [40]              28200

dense 40 (tanh)          [40]             [40]              1640

dense 40 (tanh)          [40]             [40]              1640

dense Y (tanh)           [40]             [2]               82
====================================================================
Total params: 45322
Trainable params: 45322
```

```
Non-trainable params: 0
```

7.3 FC-CNN^T GAN Model Architecture

Generator

- Input: Random vector *Z* (uniform value distribution [-1,1], |Z|=100).
- Output: Time-resolved Signal (*N*=200)

Layer (type)	Input Shape	Output shape	Param #
dense [Z]-4 (relu)	[100]	[196]	19796
reshape	[196]	[196,1,1]	0
conv2dT 32[3]:1 (tanh)	[196,1,1]	[198,1,32]	128
conv2dT 32[3]:2 (tanh)	[198,1,32]	[397,1,32]	3104
max_pooling2d	[397,1,32]	[198,1,32]	0
conv2dT 1[3]:1 (tanh)	[198,1,32]	[200,1,1]	97
flatten	[200,1,1]	[200]	0

```
Total params: 23125
Trainable params: 23125
Non-trainable params: 0
```

Discriminator

- Input: Time-resolved sensor signal (*N*=200)
- Output: Classification Real/Fake ([0,1])

Layer (type)	Input Shape	Output shape	Param #
reshape	[200]	[200,1,1]	0
conv2d 32[3]:2	[200,1,1]	[100,1,32]	128
leaky_re_lu	[100,1,32]	[100,1,32]	0
conv2d 64[3]:1	[100,1,32]	[100,1,64]	6208
leaky_re_lu	[100,1,64]	[100,1,64]	0
conv2d 128[3]:2	[100,1,64]	[50,1,128]	73856
leaky_re_lu	[50,1,128]	[50,1,128]	0
conv2d 256[3]:1	[50,1,128]	[50,1,256]	295168
leaky_re_lu	[50,1,256]	[50,1,256]	0
flatten	[50,1,256]	[12800]	0
dense 1 (sigmoid)	[12800]	[1]	12801

```
=================================================================
Total params: 375360
Trainable params: 375360
Non-trainable params: 0
_________________________________________________________________
```