Imperial College London

Machine Learning Applications in Guided Wave Testing

by

Mikolaj Andrzej Mroszczak

A thesis submitted to Imperial College London for the degree of

Doctor of Engineering

Department of Mechanical Engineering

Imperial College London

London SW7 2AZ

May 2024

Declaration of Originality

The content of this thesis is my own research work which was completed with the supervision of Doctor Peter Huthwaite and Doctor Stefano Mariani. Where I have made use of the work of others, I have made this clear and provided appropriate references.

Mikolaj Mroszczak

15/05/24

Copyright Declaration

Abstract

Guided wave testing (GWT) is a non-destructive testing (NDT) technique for in-service testing of pipes allowing the inspection of tens of metres of pipe in either direction from a single position. The aims are to identify and locate physical features along the pipe in the axial direction, particularly the defect indications, such as cracks or corrosion patches. However, the signals output by GWT of pipes are complex to interpret, making the quality of inspection highly dependent on the operator's skills. Due to signal complexities, at present there are no automated procedures to help operators in this task. Some of the recently developed machine learning (ML) algorithms are expected to possess the modelling capabilities required to address this classification task, though they would typically need hundreds if not thousands of labelled input data for their training. This amount of experimental data is seldom available in the NDT field, particularly with regard to the damage cases. This thesis explores the ML for NDT, introducing the data processing pipelines and a comparison of ML approaches. First, it is shown that high ML performance on artificial data does not necessarily translate to a similar performance on real data, motivating the need for robust ML model validation. The following results demonstrate that with scarce experimental data, substantial detection improvements can be achieved by pre-training the chosen ML model with synthetic data, before fine-tuning it on actual inspection data. In particular, the ML algorithm that is found to perform best for this task is a VGG-Net model, which is shown to yield false positive rates in the order of ~1.5 to 4% at the fixed true positive rate of 99.7%. Furthermore, the thesis explores modern generative ML approaches as potential tools to augment the data, showing the capacity to generate realistic data ex nihilo.

Acknowledgements

I would like to express my thanks to my supervisors, Dr Peter Huthwaite, Dr Stefano Mariani, and Dr Robin Jones for their heroic struggle to make my work as good as it could be. Furthermore, I would like to thank all the members of the NDT group, especially Panpan Xu, Filip Szlaszynski, Andy Zimmerman, and Yiannis Simillides for the fruitful discussions. I also thank GUL, who have supported this work financially and technically. Special thanks go to Dr Tom Vogt, Dr Sebastian Heinlein, and Dr Jimmy Fong, who have helped me with some of the toughest breakthroughs in the project.

This thesis would not come to being without my dearest friend, Dr MG, who motivated me to pursue a doctoral degree, understood and shared the experience. I have been tirelessly supported by my friends, SB and MB for much of my time at the college. Finally, I want to thank my partner, OM, who has supported me, motivated me to write and never failed to distract me from the research challenges when it was necessary.

Finally, I would like to thank my grandparents Ela and Wladyslaw, for supporting my education up to this point. My mother Aleksandra, for helping me make difficult decisions, father, Michal, for sparking my interest in engineering and technology, sister Ania, for her unconditional support and showing me a very different approach to life, and brother Stanislaw, who helped me maintain the intellectual curiosity and made my living in London significantly more interesting.

This work was supported by DTP 2018-2019 training grant EP/R513052/1.

Table of Contents

List of Figures

List of Tables

List of Acronyms

AE	*Autoencoder*
ANN	*Artificial Neural Network*
AUPRC	*Area under Precision-Recall Curve*
AUROC	*Area under Receiver Operating Curve*
CNN	*Convolutional Neural Network*
CPU	*Central Processing Unit*
CSM	*Common Source Method*
CT	*Computed Tomography*
DL	*Deep Learning*
DNN	*Deep Neural Network*
EMAT	*Electromagnetic Acoustic Transducer*
FE	*Finite Element*
FFT	*Fast Fourier Transform*
FMC	*Full Matrix Capture*
FN	*False Negative*
FP	*False Positive*
FPR	*False Positive Rate*
GAN	*Generative Adversarial Network*
GPU	*Graphical Processing Unit*
GUL	*Guided Ultrasonics Ltd.*
GWT	*Guided Wave Testing*
MAE	*Mean Absolute Error*
MaxAE	*Maximum Absolute Error*
ML	*Machine Learning*
MLP	*Multi-Layer Perceptron*
MRI	*Magnetic Resonance Imaging*
MSE	*Mean Squared Error*
NDE	*Non-Destructive Evaluation*
NDT	*Non-Destructive Testing*
PRC	*Precision-Recall Curve*

RF	*Radio Frequency*
RMSE	*Root Mean Square Error*
ROC	*Receiver Operating Curve*
SAFT	*Synthetic Aperture Focusing Technique*
SGD	*Stochastic Gradient Descent*
SH	*Shear Horizontal*
SNR	*Signal to Noise Ratio*
SV	*Shear Vertical*
TFM	*Total Focusing Method*
TN	*True Negative*
TP	*True Positive*
TPR	*True Positive Rate*
UT	*Ultrasonic Testing*
VAE	*Variational Autoencoder*
VISCIT	*Virtual Image Space Component Iterative Technique*

1 Introduction

This chapter introduces pipeline testing and explains why it is a crucial technology within the industry. It introduces the competing technologies and shows the rationale for guided wave ultrasonic testing. In the latter portion it both introduces Guided Ultrasonics Limited technology used for the production of the data used in this work and presents the practical issues that warrant the use of Machine Learning. Finally, the structure of the thesis is presented.

1.1 Pipeline Testing

Pipelines are a crucial element of the modern industrial landscape. Gas, oil, and water pipes are used to transport substances over long distances with relatively little active human involvement and downtime. Pipes are also used in industrial plants, keeping substances separate, mixing, heating, or cooling based on the technological need. Of course, all those advantages count for little if their reliability cannot be assured. In the case of chemical and oil & gas pipelines any leaks can lead to an environmental catastrophe. Apart from the obvious losses, this can incur fines and clean-up costs imposed by the regulating authorities. As a result, it is of utmost importance for the owner or an operator of a pipeline to ensure it is free of defects.

The main way to achieve the defect-free operation of pipelines is the employment of a wide range of non-destructive testing/non-destructive evaluation methods. These are broadly classified based on the physical phenomena utilised. The main modalities are visual/enhanced visual inspection, eddy current testing, radiography, magnetic testing, and ultrasonic testing. There are multiple sub-categories in each of the modalities, which can be further explored in [1]. This work focuses on the ultrasonic guided wave testing.

Generally, ultrasonic testing has a range of characteristics that make it specifically suited to the testing of pipelines. Primarily, ultrasonic testing is a full-section testing method, which means it can detect cracks under surface or on the inner surface of the pipe wall. This is especially important in the case of pipelines carrying corrosive or abrasive loads. Secondly, and just as importantly, as opposed to the other full-section method, radiographic imaging, the inspection does not pose health risks and therefore can be performed while the unit under test is normally operating. It does have disadvantages though, the most important of which being the relatively difficult signal interpretation and, in consequence, the long training process of the appropriately qualified inspectors. This quite obviously lends itself to streamlining using machine learning techniques to either automate parts of the inspection process or to work collaboratively with the inspector.

Guided wave testing is a method, in which an ultrasonic wave is propagated along the structure rather than through a cross-section (Figure 1).

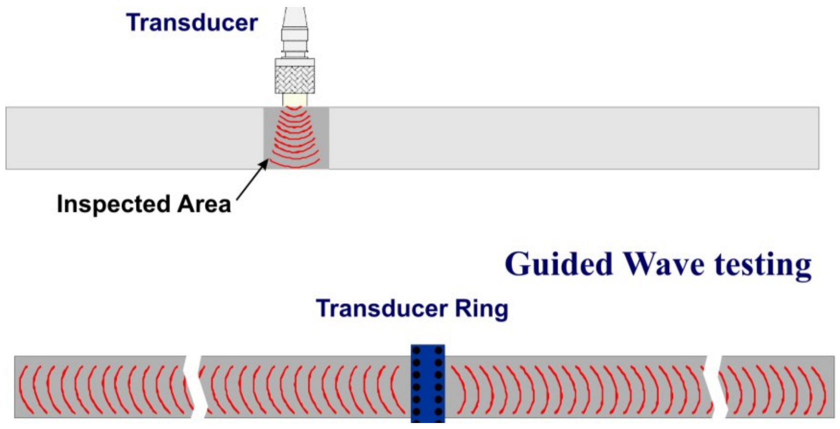


Figure 1. Conventional ultrasound (top) vs. Guided wave testing (bottom). The inspected area is shaded in darker grey, and the propagation of ultrasound is depicted in red. Reproduced from [2].

This has the significant advantage of full volumetric coverage of a large span of the pipeline from a single inspection site. Considering the defects can potentially arise at any point in the pipeline, this is a characteristic that is crucial for the knowledge of the state of the pipe. The only realistic competitor to guided wave technology is the use of pigs – trolleys equipped with an array of ultrasonic, electromagnetic, and potentially other probes which are launched into the pipe, travel through it gathering data and are extracted from an extraction point. The obvious advantage of guided wave testing when compared to pigging is the ability to test the pipeline while it is operating. Additionally, for a pipeline to be inspectable using pigs, it needs to be designed for inspection with suitable pig insertion and extraction sites placed in the pipeline [3]. The main comparative advantage of pigs is their ability to cover much longer distances than guided wave tests and the accuracy provided by the ability to conduct localised tests. A standard guided wave test can cover up to 100 m either side of the sensor location if the pipe is clean (i.e. causes no significant attenuation), straight and does not have many features. This is obviously a highly idealised scenario, but in practice the typical inspection range is in the 20-30 m range [4].

Therefore, there are scenarios in which guided wave inspection is the only acceptable technology that can be used for full-volume testing of pipelines, i.e., if they have not been designed for pigging. It is crucial to note though, that the main disadvantage of ultrasonic testing – the difficulty in the interpretation of the results – is even more pronounced in guided wave testing. This is a result of both a lower number of qualified inspectors and of an increased complexity of the data. This impacts both the technical task of interpretation and the design of inspections, driven by the engineering understanding of the physics.

1.2 Guided Wave Testing and Machine Learning

Guided wave testing has characteristics that make it uniquely suitable to the application of machine learning. The intended use case is a full volumetric coverage of a pipeline network. The total length of oil and gas pipelines in the world exceeds two million kilometres [5]. Clearly the amount of testing necessary to fully cover the network is beyond the capacity of human operators. As the process involves the inspector reaching the site, placing the transducer ring on the pipeline, taking a reading, and analysing the gathered data, much of the process needs to be automated, or at the very least streamlined to make the comprehensive testing a possibility. Guided Ultrasonics Limited has made strides to automate the hardware side of the process, by developing and introducing to the market permanently installed sensors (gPIMS). At the moment, however, one-off inspections are probably still the most prevalent application making the automation of their analysis a priority. This has led to an increased interest in machine learning techniques, which could lower the workload for the operators by automatising the signal analysis phase.

1.3 Guided Ultrasonics Limited

This work is co-sponsored by Guided Ultrasonics Limited (GUL), who are the leading provider of guided wave equipment and training. The company was established in the 1990s as a spin-out from Imperial College London, where the practical capability for guided wave inspection was developed [6]. Guided Ultrasonics equipment uses rings of piezoelectric transducers attached to the pipe to introduce the fundamental torsional T(0,1) guided wave mode into the pipeline. The transducers are operated in pulse-echo configuration, therefore requiring only a single-point access to the pipeline to perform the inspection.

Aside from being a hardware provider, Guided Ultrasonic delivers training and certification to guided wave inspectors. GUL certification follows standard NDT inspector certification routes

(L1, L2, L3 inspector) but there also exist additional certifications for particularly challenging inspection conditions, such as buried or underwater pipes. Clearly, the work of inspectors is highly specialised and very demanding. This leads to a twofold problem – the skill level can vary dramatically between inspectors and even for a given inspector be very dependent on the inspection case. As a result, there is a strong need for a method capable of standardising the results of an inspection as well as providing confidence levels in any defect indication/no defect call. Machine learning is explored as the method of choice, as even if it does not reach human/superhuman performance in the defect indication detection task, it can be used as an additional safeguard against bad calls and as a tool to quantify the complexity of the inspection task – based on the consistency of its predictions, as argued by Pyle [7].

1.4 Inspection Protocol

To understand the guided wave inspection technology, the importance of the physics behind it and the ultimate usefulness and application points of machine learning techniques, it is necessary to appreciate the current state of the art in terms of the technology used and the process followed by the inspectors to make calls.

GUL testing setup consists of a transducer ring (Figure 2) and a wavemaker instrument (Figure 3). The ring is a solid or inflatable body fitted with two rows of piezoelectric transducers which are dry (without the use of medium such as gel) coupled to the surface of the pipe using mechanical clamping or pneumatic force. The inspector places the ring on the pipe and connects the wavemaker instrument, before running a calibration protocol, which compensates for the imbalances in the coupling and sensitivity of transducers. Finally, the actual data collection routine is performed.

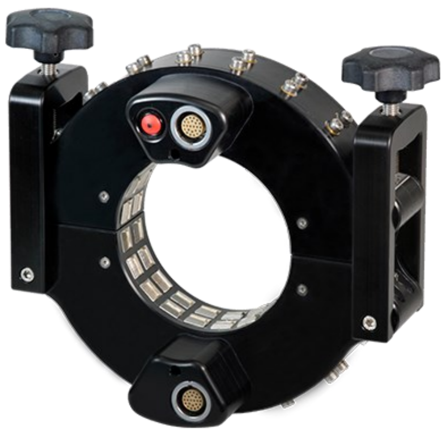


Figure 2. Guided Ultrasonics Ltd solid transducer ring [8].

Figure 3. Guided Ultrasonics Ltd Wavemaker G4 Mini [8].

The transducer rings are comprised of two rows of transducers, which allow for directional control of the inspection, making the data easier to interpret thanks to the separation of signals arriving from the two directions. This is further explored in Section 2.1.3.3. The received data is decomposed into torsional and flexural modes. This is used to distinguish between symmetric reflective features such as welds or flanges and nonsymmetric ones, such as defects and supports. Higher order flexural modes are used in synthetic focusing of the guided wave signal to produce quasi-C-scan unrolled pipe display allowing the inspector to distinguish the features more easily, with an example shown in Figure 4.

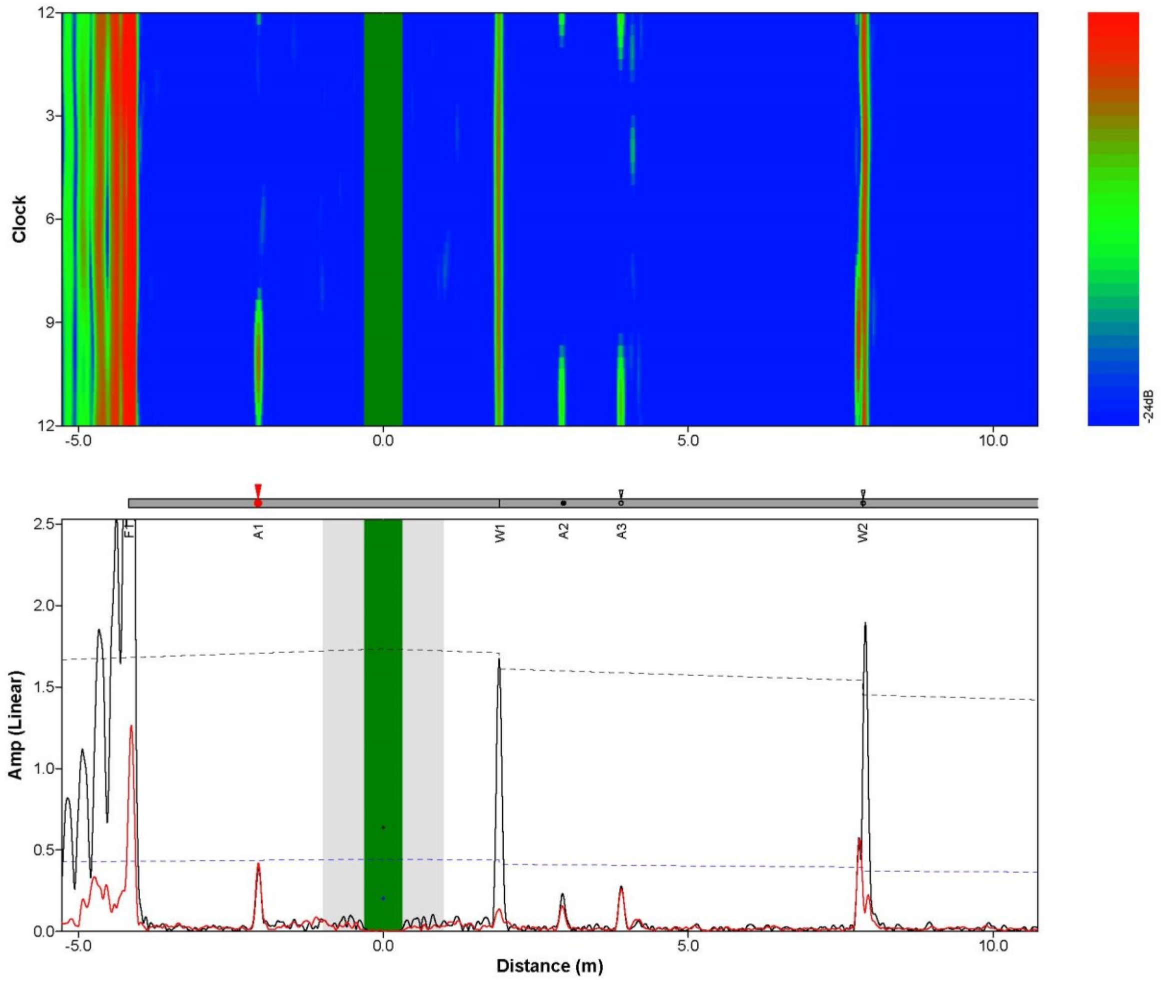


Figure 4. Unrolled pipe display (top) and enveloped symmetric (black) and non-symmetric (red) signals (bottom) [9].

After the data is collected, the inspector needs to manually process and analyse it. This is done by analysing the output similar to Figure 4. The dark green area denotes the dead zone, where the indications are too close to the ring to analyse, this is followed by the near field, marked in grey, where the noise occludes the features. The remaining features of the figure are the traces: black corresponding to the symmetric signal, and the red corresponding to antisymmetric signal. The dashed like is the DAC, marking the drop in the amplitude of the signal based on the distance. The image above the traces is the unrolled pipe display, where the amplitude of the signal is resolved circumferentially. The amplitude is graphed on a 2D display with the amplitude represented by colour. The procedure comprises annotating welds (generally easy to find strong symmetric features, for example marked as W1 and W2 in Figure 4) and ends of inspection range given usually by a flange, a bend or exceedingly low SNR – as in the dark grey vertical area marked 4 in Figure 5. These features are used to assess the attenuation of the guided wave signal in the pipeline. This is dependent on the general condition of the pipe (corrosion, contents etc.) as well as external conditions: buried sections or bitumen coating. The attenuation profile of the pipe is then calculated in terms of a distance-amplitude curve (DAC) – e.g. the trace marked 1 in Figure 5. It is a graph of a function of the amplitude of the signal from a reference reflector against distance along the pipe. The curve can be therefore used to normalise the signal along the pipe. At this stage synthetic focusing can be performed resulting in an unrolled pipe display.

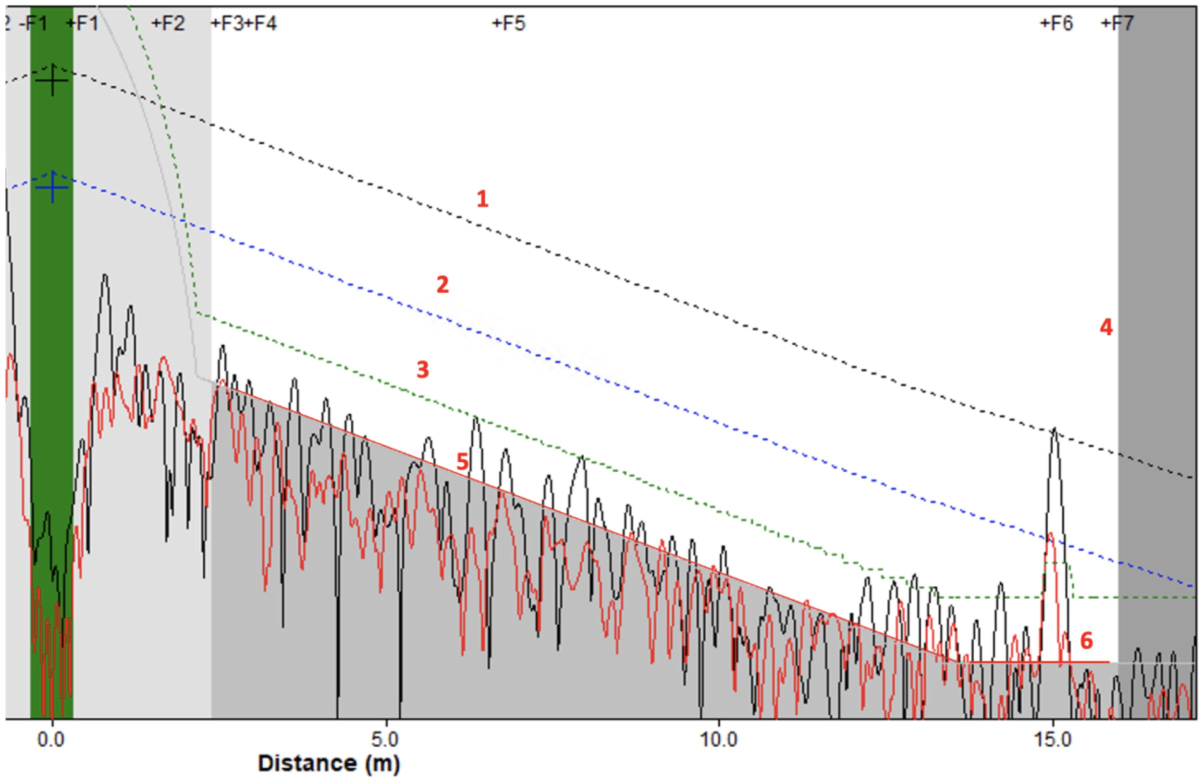


Figure 5. Inspection trace of a generally corroded pipe. Weld DAC is marked in black, call level in blue and detection threshold in green. [9]. The plot uses logarithmic scale on the y-axis.

Following this, the inspector performs a frequency sweep, assessing the signatures across the inspection frequency spectrum (ordinarily between 15 and 50 kHz) to differentiate between frequency dependent and independent features (for example, supports vs. defects). After that, the inspector flags all the signals above a set detection threshold (green dashed line marked 3 In Figure 5) and classifies them as either a benign feature or a defect indication. This decision is usually based mostly on the unrolled pipe display, but often requires inspection at a range of frequencies as well as significant experience. The minimum amplitude of the signature that can be detected is governed by the detection threshold which in turn is dependent on the noise floor level (area shaded light grey in Figure 5). The noise floor level is the higher of two contributors:

1) the coherent noise, usually caused by the general condition of the pipe or imbalance between the transducers. This introduces unwanted guided wave modes, which fall in amplitude according to the DAC and is therefore the detection limiter throughout the inspection length (red line marked 5 in Figure 5). Importantly, the coherent noise cannot be reduced by averaging and is typically the limiting factor in GWT.
2) the random noise, which stays constant, is usually caused by electronic noise, and governs the range limit of the inspection. The limit is given by the point at which the call level (blue line marked 2) is equal to detection threshold (red line marked 6 in Figure 5).

When all the signals have been annotated, the inspection is complete, a document outlining the locations and labels of identified signals is generated, and further decisions are scheduled. As guided wave testing is a screening tool, a follow-up inspection must be performed using conventional ultrasonic or other testing, dependent on the industry standards, is performed at the areas of concern flagged by the initial guided wave test, in order to gain more precise local information.

1.5 Motivation for This Project

Guided Ultrasonics Ltd is interested in the application of machine learning to their guided wave data for a range of reasons. The main rationale is that there is a large variability in the inspection performance between inspectors. As a result, the perceived quality and reliability of guided wave inspection is not based solely on the quality of technology and there is a lack of an objective minimum performance level. This could be remedied by the implementation of an ML defect indication detector; upon whose predictions the inspectors experience could only improve.

The application area is faced with the problem experienced by much of the research concerning the applications of machine learning in NDT– the issue of data scarcity. Most modern machine learning models are trained on datasets of millions of samples. That is clearly not realistically viable with datasets gathered manually by the inspectors. Imperial College NDT group is world leading in numerical simulations of guided waves; therefore, it is hypothesised that finite element modelling could be utilised to circumvent the data scarcity problem following the trend of physics-informed machine learning. This creates a set of questions interesting from the academic viewpoint, from generating simulations representative of the real-world inspection scenario, through processing the raw data into formats implicitly informed by the physical knowledge, to designing machine learning architectures best suited for bridging the gap between the simulated and real data (so called sim2real gap). Thus, this project aims to develop a method to utilise the simulated data in synergy with the experimental data in order to develop a machine learning algorithm to support the work of GWT inspectors.

1.6 Thesis Plan

In the second chapter of this thesis, the theoretical background behind guided wave testing of pipelines, the underlying wave physics, and its practical application in the design of testing procedures and defect indication detection are introduced. Following this, the finite element method – a necessary tool for the generation of large-scale training datasets for machine learning is discussed. The historical background and the theory of machine learning are introduced, with the underlying mathematical principles and the constituent parts of a neural network considered. The chapter concludes with the discussion of the emerging trends and challenges in machine learning.

In the third chapter the guided wave data processing necessary for its use in machine learning applications is explored. The characteristics of guided wave data and the characteristics of its processing from the point of view of machine learning are discussed, with a special focus given to the modal decomposition of the raw data and synthetic focusing methods. The problem of equivalent processing of simulated and real-world data is considered in the context of its impact on the final ML performance. Finally, the production of finite-element data is elucidated and the process for its quick generation while including as many of the characteristics of real signals as possible is described.

In the fourth chapter the design of the machine learning architectures is discussed and the specific design choices are confirmed with experiments. The neural network architectures used are described, including the rationale behind their choice and the process of their adjustment to the specific requirements of GWT. The performance metrics common to ML research are introduced and their applicability to this project described. A novel metric denoted FPR@1TPR is described and the rationale for its usage explained. The metrics are used to present the performance of the architecture on a simulated dataset. A study is conducted to assess the methods of data processing, the architecture design and the selection of the training parameters in the context of their impact on the performance metrics. Further, the problem of sim2real gap is introduced and methods are described that could be used to bridge it and rationalise the choice of transfer learning.

The fifth chapter builds on the contents of the previous chapter by introducing the resultant performance of the ML model on in-service inspection data. In the last part of this chapter, a simplified case of detecting any type of pipe feature in guided wave signals is considered allowing for the use of larger datasets. The results are framed in the context of ML dependence on the number of real and simulated datapoints. Furthermore, the samples misclassified by the best-performing ML model are further investigated and explained.

In the sixth chapter the application of generative learning in guided wave testing are discussed. The implementation of Generative Adversarial Network (GAN) for guided wave problems is introduced. The results of a guided wave generator trained on simulated dataset are presented and the possibility of a more controlled simulation is introduced. The potential future uses of generative learning as well as its drawbacks are discussed. Furthermore, the problems in the development of machine learning in the guided wave or non-destructive testing in the context of the trust and reliability of the ML methods are described.

In the seventh chapter, a self-contained side project is described, in which ML methods are employed to compensate for the artifacts generated by limited view transducer configuration in tomographic thickness reconstruction. The chapter introduces the issues, the method of generating limited view registrations, the ML architecture selection, training and comparative performance against conventional state-of-the art algorithm.

Finally, in the eighth chapter the conclusions and recommendations for further study are outlined.

2 Theoretical Background

This chapter is split into two sections, as the topic of this work has two distinct constituent parts, the guided wave testing and the machine learning. In the first section this chapter introduces the necessary physical background to understand guided wave testing. It discusses wave propagation and interaction with reflectors such as pipe features. Further, it introduces the implementation of differential equations describing the wave propagation in Pogo finite element modelling package. Finally, it briefly discusses ultrasonic transduction. In the second part it introduces machine learning, starting from the historical evolution of the research field and the introduction of the main modalities of Machine Learning. Discussion of neural networks follows with the necessary mathematical background as well as the discussion of why they are the algorithm of choice for much of modern machine learning. Finally, the recent developments in machine learning are discussed, as well as explainability and ethical concerns with its spread.

2.1 Ultrasonics

2.1.1 Mechanical Waves Background

The first part of this chapter is concerned with understanding ultrasonic waves as a means to inspect pipelines. Ultrasonic waves can be generally divided into bulk waves and guided waves. Bulk waves travel in the regions away from surfaces and their longitudinal and shear components are decoupled. Bulk waves interact with interfaces via reflection and refraction. Mode conversion between longitudinal and shear waves can also take place during those interactions. Where the interface in a continuous condition upon which the propagation of the wave depends (i.e., an ultrasonic wave travelling along the surface) the theoretical framework for the analysis of the propagation shifts to the study of a guided wave. In general, bulk wave propagation is dependent only on the material properties. Guided wave propagation, in contrast, is determined both by the material properties and the geometry of the medium (waveguide).

The guided wave solutions are dependent on the geometry of the propagation medium with it impacting the solution to the wave equation. As a result, the initial solutions of the guided wave propagation problem were limited to model environments with the solutions named after the discoverer: Rayleigh waves, Lamb waves and Stoneley waves.

Rayleigh waves [10] exist on a surface of a semi-infinite medium. An example of such are earthquake surface waves. Their propagation is defined by a traction-free surface and a decay away from the surface. Lamb waves [11] are surface waves that exist in thin plates. They are commonly used in material testing of plate and shell-like structures. They are defined by traction-free surfaces on both sides. Stoneley waves [12] are surface waves that exist at the interface of two media able to support the propagation of waves. They exist on a solid-solid or a solid-liquid boundary. In the latter case they are known as Scholte waves. Stoneley waves are defined by the matching solution for particle displacement in both media at the interface and the decay away from the boundary.

To understand the specifics of guided wave inspection and their usefulness for non-destructive testing it is crucial to understand the dispersive behaviour of guided waves. The phenomenon of dispersion is based on the concepts of phase and group velocity. Group velocity (v_g) can be understood as the speed with which the envelope of the signal propagates through space. Phase velocity (v_p), in contrast, is defined as the speed with which a single peak of the wave travels. Mathematically, they are defined as:

$$v_p = \frac{\omega}{k} \tag{1}$$

$$v_g = \frac{\partial \omega}{\partial k} \tag{2}$$

where ω is the angular frequency of the wave and k is the angular wavenumber. The consequences of the formulations become clear with the introduction of the concept of the wavepacket – a group of superposed waves which together form a localised travelling disturbance. A wavepacket is conventionally represented as a product of the carrier wave and the envelope, but frequency domain analysis shows it can also be described as a linear combination of waveforms varying in frequency around the carrier wave frequency. Thus, despite often being described in the terms of carrier (centre) frequency, it is crucial to understand that a wavepacket is in fact a multi-frequency wave combination. The formulations of phase and group velocities thus lead to three possible scenarios for a wavepacket:

- wavenumber is directly proportional to frequency: if that is the case, phase and group velocities are equal, the wavepacket does not exhibit dispersion and the individual peaks and troughs do not move in reference to the envelope.

- wavenumber is a linear function of frequency: phase and group velocity are not equal, but the group velocity is constant regardless of the frequency. The individual peaks and troughs move inside the envelope, but the envelope travels undistorted.
- wavenumber is a nonlinear function of frequency. The group velocity depends on the frequency. In the context of wavepacket the constituent waves travel at various velocities leading to distortion of the overall envelope. This distortion is known as dispersion. The broader the range of frequencies present in the wavepacket (bandwidth) the more significant the dispersion.

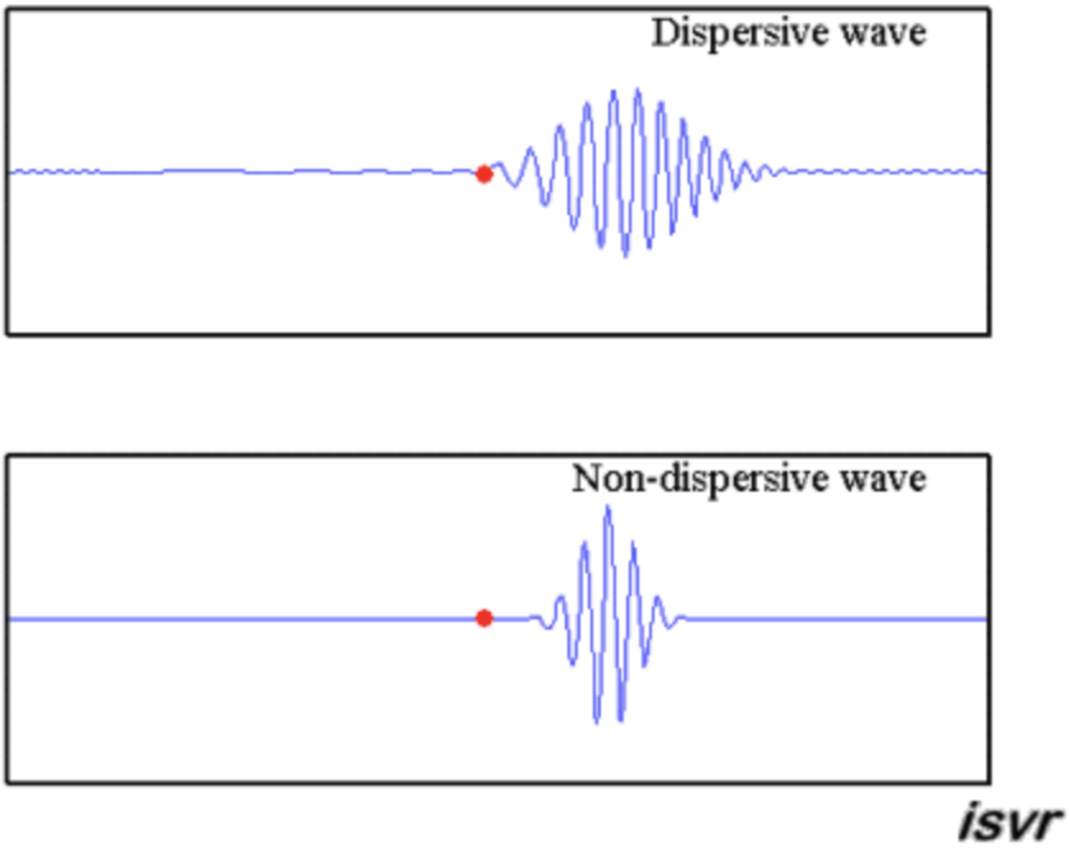


Figure 6. Dispersed (top) and non-dispersed (bottom) wave on a displacement – propagation distance graph. The original shape of the wave is the same as the bottom plot (6-cycle Hann-windowed sinusoid). Reproduced from [13] originally a movie, where red dot is used to visualise local displacement.

Dispersive behaviour has two major consequences in the context of guided wave testing, which are demonstrated on the example of waves in Figure 6. In the example the wave is originally the same shape as the nondispersive snapshot. After propagation, a dispersive wave changes shape while a non-dispersive remains constant. First, with the dispersion the longitudinal extent of the wave increases. As the goal of guided wave testing is the localisation of the interaction of the probing wavepacket with the defect, the longer wavepacket lowers the resolution of the inspection. Secondly, the maximum amplitude of the wavepacket is decreased with dispersion. The amplitude of the signal reflected by the defect is proportional to the amplitude of the probing wave, therefore weakening the probing wave may lead to the reflected defect indication signal falling below the detection threshold, thus leading to missed defect indications.

In the terms of bulk waves, there is typically only one phase and group velocity for a given medium, hence they propagate without dispersion. Guided waves, however, are typically dispersive, since their behaviour will vary as the relationship between wavelength and fixed geometry will vary through the frequency range. There are an infinite number of solutions for a guided wave travelling within a pipe each having a different dispersion relationship. As a result, a solution must be selected for which the dispersion is low (group velocity does not change strongly with frequency) or, ideally, is completely non-dispersive. This makes the calculation of $v_g(\omega)$ function (dispersion curve) necessary for guided wave inspection design. The wave equation for the guided wave needs to be solved to achieve this.

This section introduces a brief solution to bulk wave propagation and later focuses on the solution for Lamb waves. In fact, the standard formulation of the solution of the wave equation for thin-walled pipes proposed by Gazis [14] is based on the Lamb waves in plates, with the thin-walled cylinder of high radius compared to the wavelength considered a special case of a plate.

2.1.1.1 Wave Propagation

Propagation of bulk wave is governed by Navier's equation:

$$(\lambda + \mu)\nabla\nabla \cdot \boldsymbol{u} + \mu\nabla^2\boldsymbol{u} = \rho\left(\frac{\partial^2\boldsymbol{u}}{\partial t^2}\right) \tag{3}$$

where λ and μ are Lame constants reflecting material properties, ρ is the density of the material and $\boldsymbol{u}$ is the displacement vector. Using Helmholtz decomposition, the equation can be split into:

$$\nabla^2\varphi = \frac{1}{c_L^2}\frac{\partial^2\varphi}{\partial t^2} \tag{4}$$

and:

$$\nabla^2\psi = \frac{1}{c_T^2}\frac{\partial^2\psi}{\partial t^2} \tag{5}$$

φ is the scalar potential while ψ is the vector potential, c_T and c_L are the shear and longitudinal wave phase velocities, which can be calculated as:

$$c_L = \sqrt{\frac{\lambda + 2\mu}{\rho}} \tag{6}$$

$$c_T = \sqrt{\frac{\mu}{\rho}} \tag{7}$$

The equations represent shear and longitudinal waves that can propagate through the medium independently in the absence of any inhomogeneities. We can now use this solution to derive the formulae for dispersion relations in guided waves.

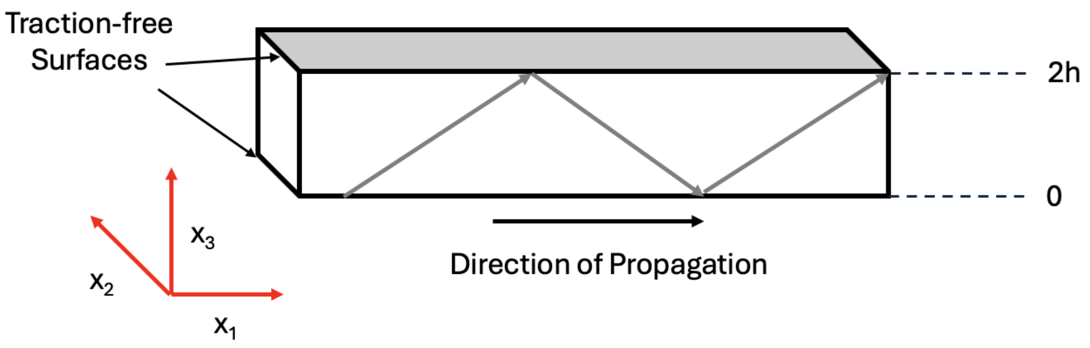


Figure 7. Geometry of guided wave propagation in a free plate.

Considering the case of Lamb waves, we are assuming traction-free surfaces as shown in Figure 7. We can use the Helmholtz-decomposed Navier equation and follow the derivation by the method of potentials [15] to arrive at two equations, known as the Rayleigh-Lamb frequency relations. They tie the frequency and wavenumbers of Lamb waves depending on whether the specific solution is symmetric or non-symmetric. For symmetric modes:

$$\frac{\tan(q2h)}{\tan(p2h)} = \frac{4k^2pq}{(q^2 - k^2)^2} \tag{8}$$

while for non-symmetric:

$$\frac{\tan(q2h)}{\tan(p2h)} = \frac{(q^2 - k^2)^2}{4k^2pq} \tag{9}$$

in the preceding equations p is defined as $p^2 = \frac{\omega^2}{c_L^2} - k^2$ and q as $q^2 = \frac{\omega^2}{c_T^2} - k^2$, while $2h$ is the thickness of the plate. The equations can be solved at any frequency to calculate the phase velocity of the propagating wave. As the equations are hard to solve analytically, specialised tools such as the DISPERSE software package [16] can apply computation methods to find the roots of the equations. The solutions are plotted to assess which modes exist in the plate at a given frequency as well as to compare their dispersiveness for the purpose of non-destructive evaluation.

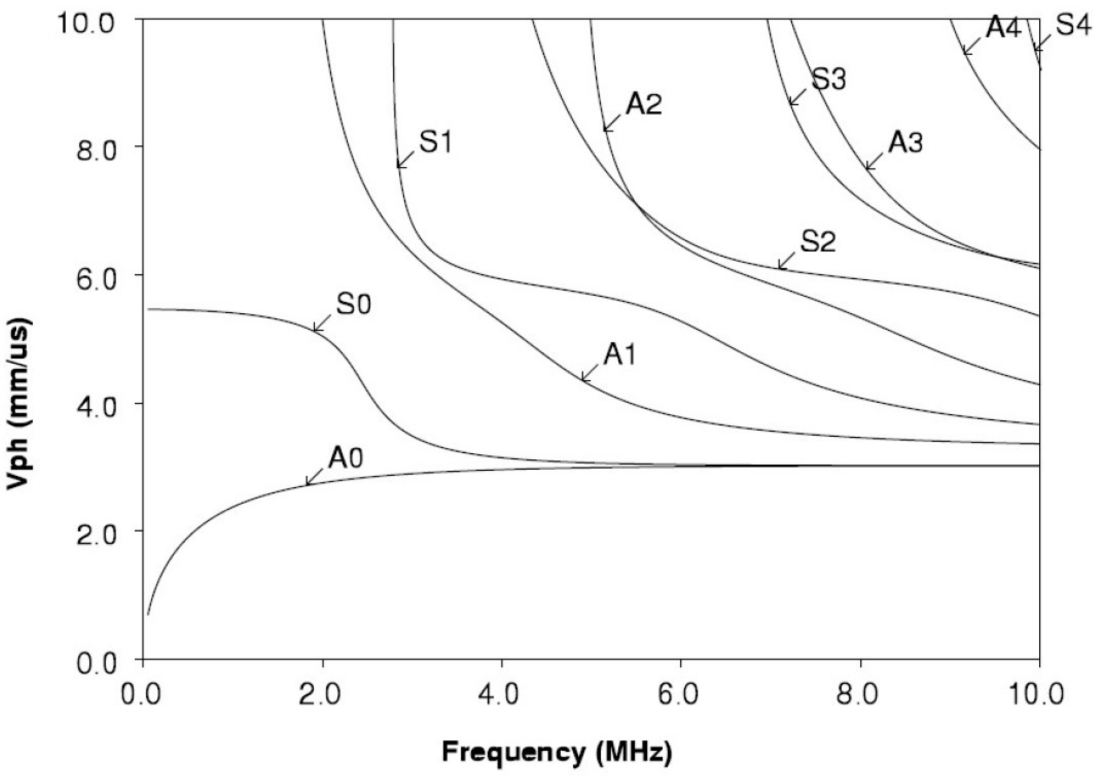


Figure 8. Dispersion curves (phase velocity vs. frequency) of first symmetric and non-symmetric Lamb modes in a 1 mm thick steel plate. The mode names are annotated in the figure [15].

Figure 8 presents dispersion curves for low order symmetric (S) and non-symmetric (A) modes. Graphs like this one can be used to determine which modes exist at a given frequency, as subsequent modes appear with the increase in frequency (i.e. A1 around 2 MHz, S1 about 2.5 MHz). The lowest frequency at which a mode exists is known as that mode's cut-off frequency. The existence of the higher order modes makes the signal more difficult to interpret, as the energy of the probing wave is spread between the intended and unintended modes. The second use of the dispersion curves is the assessment of the dispersiveness of the selected mode at the selected frequency. Group velocity can be found using phase velocity as:

$$c_g = c_p^2 \left(c_p - (f2h) \frac{dc_p}{d(f2h)} \right)^{-1} \tag{10}$$

where $f2h$ is the product of frequency and thickness of the plate. For a given plate, therefore, the group velocity is dependent on the derivative of phase velocity with respect to frequency. Referring to Figure 8, this is simply a gradient of the dispersion curve. To conclude, to select the probing frequency it is best to choose a frequency at which few modes exist, and the gradient of the selected mode is as low as possible.

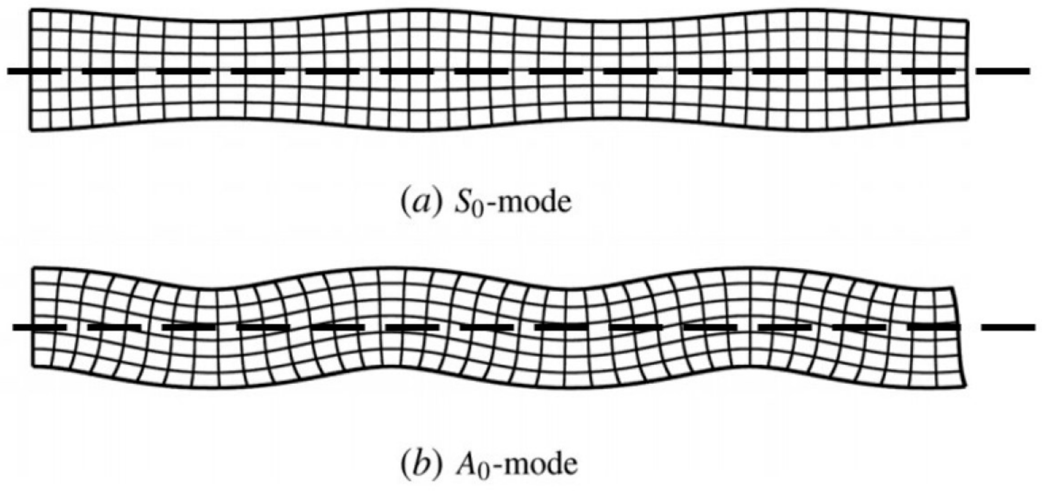


Figure 9. Graphical representation of symmetric (a) and non-symmetric (b) fundamental mode in a plate [16].

Figure 9 shows the symmetric and non-symmetric character of the modes in plates. In low frequency-thickness regimes non-symmetric modes can be understood as bending the plate, while the symmetric as compression-extension.

2.1.1.2 Shear Horizontal Waves

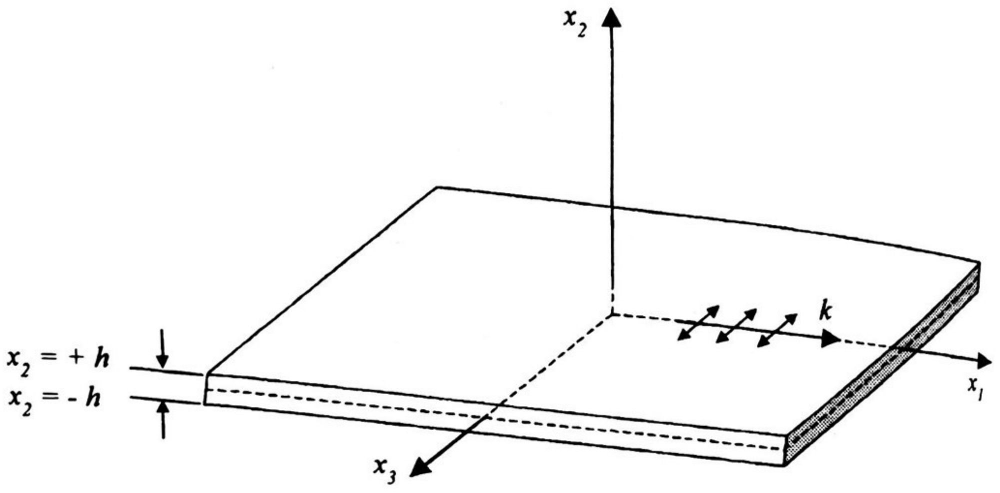


Figure 10. Shear wave propagation along axis x_1 where the particle displacement is along the axis x3. The wavenumber of the propagating wave is k, and the thickness of the plate is 2h. Reproduced from [17].

Other than the symmetry an important characteristic of the wave is the main displacement direction of the particles. For Lamb waves, the displacement direction is in the propagation direction (longitudinal) and out of plane of the plate (shear vertical – SV). The third dimension, in the plane of the plate but perpendicular to the propagation direction, is known as shear horizontal (SH), as graphically represented in Figure 10.

SH waves have a range of advantages when it comes to non-destructive testing. Recall the general consideration for the mode selection. The ideal inspection conditions would support as few modes as possible, and the main probing mode would be nondispersive. Refer to Figure 11, which presents the SH dispersion curves in a steel plate. The fundamental SH mode, n=0 in the figure, is completely nondispersive over the full frequency range. Furthermore, the second symmetric mode (n=2) exists only above the frequency-thickness product of 4 MHz-mm, which, when compared to Figure 8 gives a significantly broader frequency range with one symmetric mode than Lamb waves. Finally, shear waves are not supported by liquids, thus the waves will not transfer from the pipe into the contents or the external medium, lowering the energy of the propagating wave and thus limiting the inspection range. This is especially important if the testing is performed on submerged plates or filled pipes.

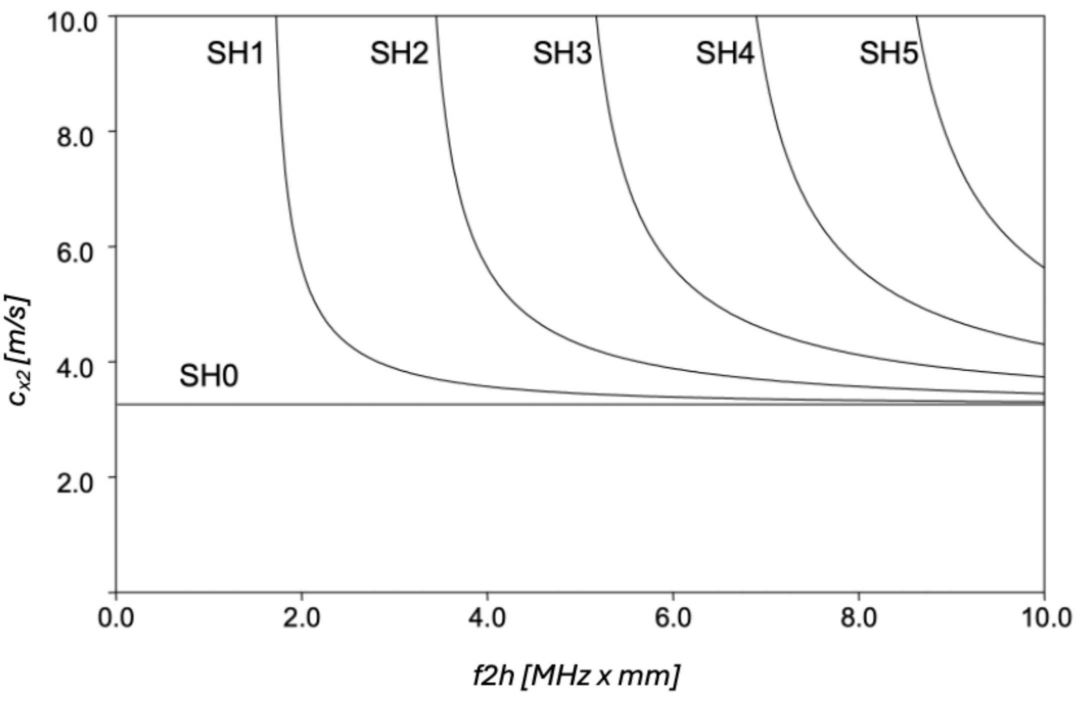


Figure 11. Dispersion curve for SH mode family in a steel layer. Wave speed in x_2 dimension against frequency-thickness product.

2.1.1.3 Guided Waves in Hollow Cylinders

Until this point, we have dealt only with plates as the structure supporting the propagation of guided waves. They can however propagate in other geometries too, most notably rods and hollow cylinders (pipes). While guided wave propagation in rods is beyond the scope of this work, understanding the shapes of the modes in rods is more intuitive than in cylinders, as the latter uses the plate-pipe analogy. Figure 12 demonstrates the torsional mode in rods corresponding to shear horizontal mode in plates in the terms of frequency spectrum. In the terms of particle displacement, the rod is twisting. Figure 13 demonstrates the flexural mode, effectively bending the rod. Considering the axial symmetry of the problem torsional mode is symmetric, while flexural mode is non-symmetric. This characteristic persists in the case of pipes and is extensively used in this work.

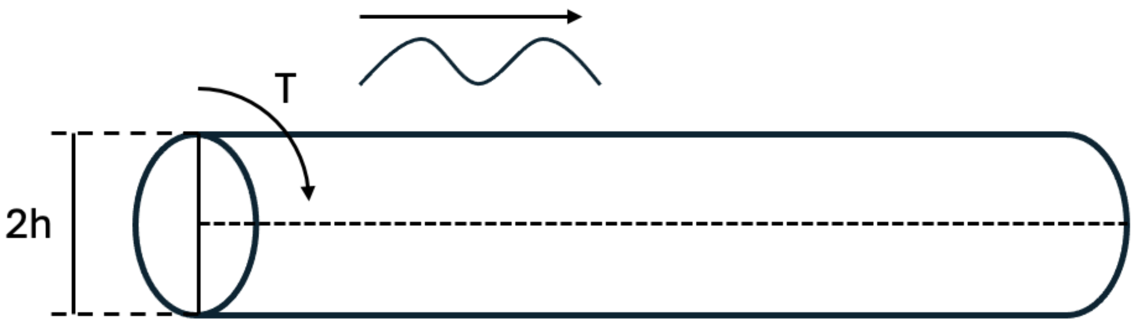


Figure 12. Torsional modes in a solid rod. The lack of deformation of the rod is shown by the straight dashed axis.

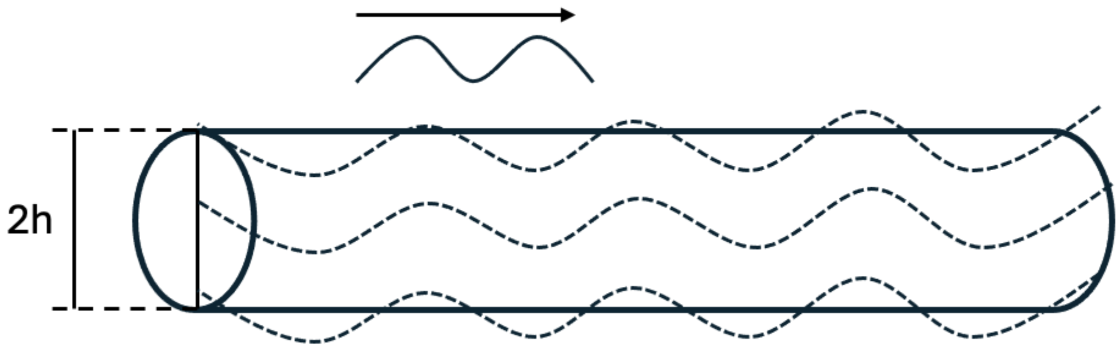


Figure 13. Flexural modes in a solid rod. The deformation of the rod is shown by the axis and top and bottom boundaries.

The waveguide geometry considered in this work corresponds to a pipe, an idealised version of which is a hollow cylinder. The wave equation for this geometry has first been solved by Gazis in 1959 [14]. The resulting formulae for particle displacement are:

$$u_r = U_r(r)\cos(n\theta)\cos(\omega t + kz) \tag{11}$$

$$u_\theta = U_\theta(r)\cos(n\theta)\cos(\omega t + kz) \tag{12}$$

$$u_z = U_z(r)\cos(n\theta)\cos(\omega t + kz) \tag{13}$$

This formulation uses a cylindrical reference system, presented in Figure 14, in which r is the distance from the reference point along the radius of the cylinder, z is the distance along the axis of the cylinder and theta is the angle. In the equations above *n* is the circumferential order of the solution, *u* is the particle displacement while *U* is the displacement potential amplitude composed of Bessel functions or modified Bessel functions. For full derivation, refer to Rose's textbook [17].

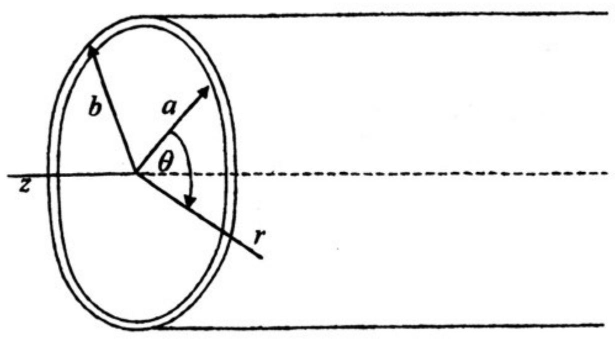


Figure 14. Hollow cylinder of internal radius a, and external radius b. Reproduced from [17].

When analysing the wave propagation in hollow cylinders we can split the modes into three mode types: longitudinal, torsional, and flexural. Torsional and flexural modes have been introduced for the rod case, while longitudinal corresponds to simple compression-rarefaction plate case. In this work the modes are annotated using the notation of Silk and Bainton [18]. The notation is:

- Longitudinal modes: L(0, m)
- Torsional modes: T(0, m)
- Flexural modes: F(n, m)

In this notation the first index (n) corresponds to circumferential order – 0 for longitudinal and torsional signifies their axial symmetry. The second index (m) is a counter of mode number. Figure 15 demonstrates practically the annotation of modes and introduces the modes of interest in this work. In guided wave testing of pipelines, the modes reflected by a feature belong to the family of that probing mode. Those are defined as the flexural modes whose dispersion curves converge to the base mode at infinite frequency. For T(0,1) the family of modes includes F(1,2), F(2,2), F(2,3) etc. The number of modes in the family depends on the frequency of test, due to the cut-off frequencies of higher circumferential order modes.

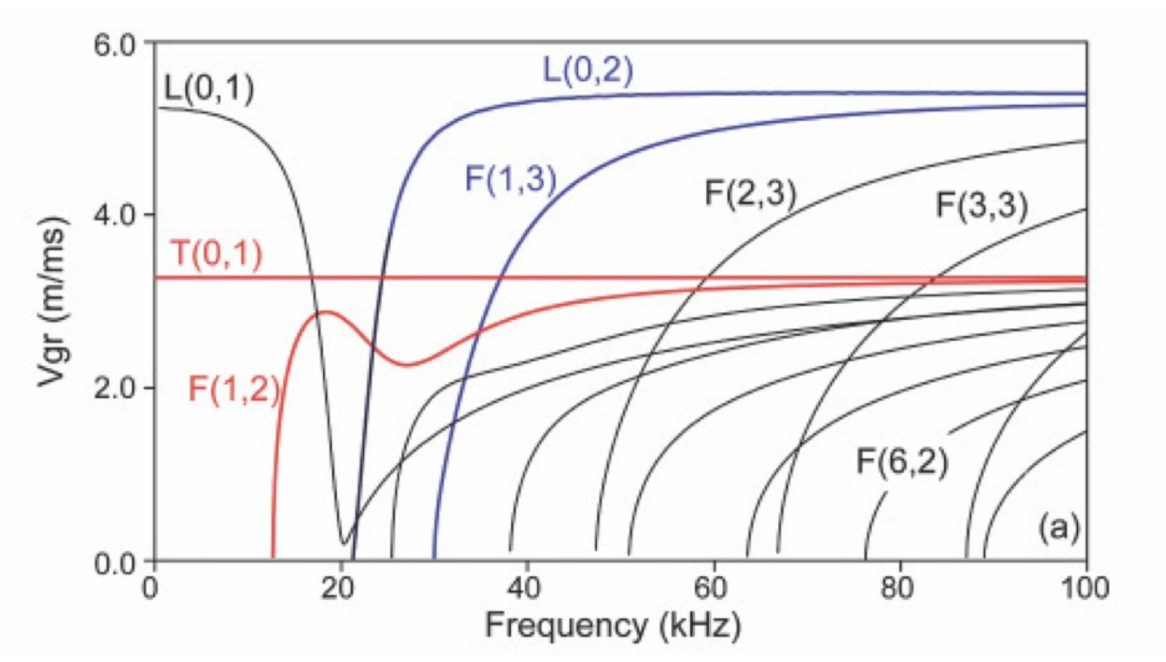


Figure 15. Group velocity against frequency dispersion curves for 3-inch schedule 40 steel pipe. Reproduced from [19].

The modes usually used for inspection are either L(0, 2) mode family (highlighted in blue in Figure 15) or T(0, 1) mode family (highlighted in red). The families are defined by the modes having primarily the displacement in the torsional direction F(1, 2), F(2, 2)… or longitudinal direction for F(1, 3), F(2, 3)… For the full consideration of the advantages and disadvantages of either of those selections, refer to [19]. In the case of this work, the data used is obtained using T(0, 1) as the probing mode and T(0, 1) and F(n, 2) being the reflected modes. In the following sections of this work the notions of *symmetric* or *torsional* mode refer to T(0, 1), *non-symmetric* or *flexural* refer to the whole F(n, 2) group of modes. *Higher order flexural modes* refers to F(n, 2) where n>1.

2.1.2 Wave Propagation Modelling

The propagation of guided waves in pipelines can be solved analytically only if the geometry of the pipe does not diverge from the idealised mathematical model. In reality, the excitation of the wave modes is rarely pure, and the pipeline geometry is affected by corrosion and the presence of features. Furthermore, the wave propagation itself can be affected by factors such as temperature and contents of the pipe. Considering that the goal of modelling the waveform propagation in this work is to use it in a machine learning context, it is necessary to model wave propagation in geometries as far away from the ideal solution as possible and to model a large number of such propagations. The standard method for modelling guided waves in complex geometries is the usage of finite element analysis (FEA), as it allows for the simulation of forces and displacements in arbitrary shapes. This work makes extensive use of Pogo [20]. Pogo is an explicit time domain FE solver, complete with model building toolkit for MATLAB. The capability to programmatically build geometries in MATLAB and export them into Pogo mesher is crucial for this project, as it allows for a quick and hands-off parametric generation of large amount of training data for the machine learning algorithm. In the terms of Pogo operation, this section quickly derives the central difference method used for the wave propagation modelling. By imposing dynamic equilibrium, we can write:

$$[\boldsymbol{M}]\frac{\partial^2 \boldsymbol{u}}{\partial t^2} + [\boldsymbol{C}]\frac{\partial \boldsymbol{u}}{\partial t} + [\boldsymbol{K}]\boldsymbol{u} = [\boldsymbol{F_a}] \tag{14}$$

In this equation ***[M]*** is the mass matrix, ***[C]*** is the damping matrix and ***[K]*** is the stiffness matrix. ***[F$_a$]*** is the external force and ***u*** is displacement. The central difference method of numeric differentiation uses the previous and next values of displacement to calculate the derivatives of displacement with respect to time. Respectively:

$$\frac{\partial \boldsymbol{u}}{\partial t}(\tau) = \frac{\boldsymbol{u}(\tau + \Delta t) - \boldsymbol{u}(\tau - \Delta t)}{2\Delta t} \tag{15}$$

$$\frac{\partial^2 \boldsymbol{u}}{\partial t^2}(\tau) = \frac{\boldsymbol{u}(\tau + \Delta t) - 2\boldsymbol{u}(\tau) + \boldsymbol{u}(\tau - \Delta t)}{\Delta t^2} \tag{16}$$

Pogo solver is explicit, which means $\boldsymbol{u}(\tau + \Delta t)$, i.e., the future displacement, calculation can be based only on the current and previous values of displacement. Therefore, rearranging the equations for $\boldsymbol{u}(\tau + \Delta t)$:

$$\boldsymbol{u}(\tau + \Delta t) = \left(\frac{[\boldsymbol{M}]}{\Delta t^2} + \frac{[\boldsymbol{C}]}{2\Delta t}\right)^{-1}\left[[\boldsymbol{F_a}] + \left(\frac{[\boldsymbol{C}]}{2\Delta t} - \frac{[\boldsymbol{M}]}{\Delta t^2}\right)\boldsymbol{u}(\tau - \Delta t) + \left(\frac{[\boldsymbol{M}]}{\Delta t^2} - [\boldsymbol{K}]\right)\boldsymbol{u}(\tau)\right] \tag{17}$$

The above equation depends on the mass, stiffness, and damping matrices, which depend on the material. They can be simplified if the material is assumed to be homogenous and isotropic, which is the case for this project. The external force matrix is best understood as the excitation; therefore, it is only nonzero when the initial wavepacket is transmitted, and at the source location. Finally, the equation depends on the choice of Δt, known as the time step. It is selected to be low enough to ensure simulation stability, with the exact value determined as a balance between the accuracy of the simulation and the computational cost. As the overall time of the simulation is generally enforced by the goal of the simulation (i.e., a wave takes a certain time to propagate over the geometry), lowering the time step directly increases the number of calculations. The largest time step that still allows for accurate simulation of wave propagation corresponds to the time it takes for the fastest wave to pass through the smallest element in the model:

$$t_c = \frac{\Delta x}{c} \tag{18}$$

where t_c is the largest possible time step that does not cause the instability of the model (critical time step), Δx is the smallest element length and c is the wave speed of the fastest wave modelled. The ratio $\Delta t/t_c$ is known as Courant number, it is necessarily lower than 1 to ensure stability in the explicit time stepping and is one of the parameters that can be changed during the model optimisation.

The elements used in FEM exist in a variety of shapes, in the case of this project, cubic elements are used. As such, only one size of the element, Δx, needed to be selected. The Nyquist criterion states that to express any periodic signal it is necessary to sample at least two points in each period. This translates to at least two elements per wavelength of the shortest (highest frequency) wave simulated. In practice, more elements are used to model the wave behaviour. Marburg reports the minimum of six elements [21], however, best practice is to tailor the size of the elements until the FE results converge to a desired error margin, which usually results in significantly more elements per wavelength [22].

Finally, when designing a finite element simulation, the boundary conditions must be considered. As a standard, Pogo uses sound-soft boundary conditions – corresponding to the object ending abruptly in a vacuum or other non-sound conductive material. This is rarely the case for the boundaries in the longitudinal direction in real-life applications, where the range is more commonly limited by the attenuation or the presence of a pipe feature. As the simulations used by this project were required to be as close to the real world as possible, the goal is to avoid the boundary region reflections, as such signals do not exist in real pipes; the simulation should behave as a finite section of an infinite domain. There are two primary methods of achieving that goal. One option is to extend the model in the direction of propagation and using time windowing to reject the signal reflected off the end of the region. This solution allows all the unwanted signal to be rejected but makes the domain significantly larger than it needs to be, resulting in an increased computational requirement. The second option is the implementation of absorbing boundary conditions. Pogo has built-in functions facilitating the addition of absorbing boundaries using stiffness reduction method (SRM) [23]. This method alters the damping and stiffness matrices of the system. The gradual increase in damping over the length of the absorbing boundary lowers the amplitude of the wave, so that the impulse that reaches the end of the domain is low enough not to alter the results of the simulation. The stiffness matrix is reduced as damping is increased, further minimising the mismatch in acoustic impedance between the elements. While the original paper suggests an absorbing boundary as thin as 1.5 wavelength is sufficient to minimise any reflections to a negligible level, the thickness of SRM absorbing boundary is another parameter that can be tuned until the convergence to simulation requirements occurs.

2.1.3 Imaging

Imaging in the context of guided wave testing is the process of transforming the raw ultrasonic signal received either from the measurement equipment or finite element simulations into a human-interpretable thickness map of the region of interest. It is widely used in human-operated inspections, as the imaged signal is significantly easier for humans to interpret. The development of imaging for guided wave testing has given it the potential to become a viable quantitative NDT method. This project is interested in the imaging techniques, as their usage introduces additional knowledge of physical wave behaviour into the data, thus adding valuable information a machine learning algorithm can utilise. Furthermore, ML on pictorial data is significantly more developed than on waveform-type data, providing further incentive in the ability to use the experience developed in ML applications in different fields.

2.1.3.1 Artificial Focusing Methods

Three commonly used acquisition methods for ultrasonic imaging, be it in guided wave testing or conventional ultrasound are Common Source Method (CSM), Synthetic Aperture Focusing Technique (SAFT) [24] and Full Matrix Capture (FMC) [25]. CSM is a special case of Plane Wave Imaging [26] using a single plane wave generated by simultaneous excitation of all transducers. After the acquisition of data, it is processed (imaged) to reconstruct the physical characteristics of the imaged body. The selection of methods used in this section follows [27].

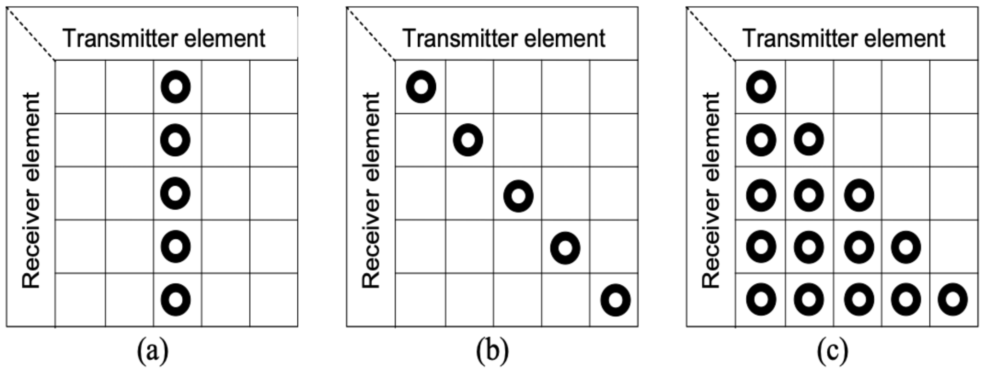


Figure 16.Transmitter-reciever geometries of CSM(a), SAFT(b) and TFM(c) synthetic focusing methods. Reproduced from [27].

The algorithms are differentiated by their source-receiver geometries (see Figure 16). CSM uses all transducers as a single source for excitation and receives on all elements of the array. SAFT fires each transducer in turn and uses the pulse-echo trace. TFM uses every transmitter-receiver combination. This format of data is known as full matrix capture, and it can be used to perform either of the artificial focusing approaches. It is worth noting that while TFM uses FMC, the data can be stored as triangular half matrix (as visible in Figure 16c) due to the reciprocity principle meaning that the measurement by transducer B given an excitation at A is the same as the measurement by transducer A for an excitation at B (neglecting noise/errors). Based solely on the transmitter-receiver geometries, there are historically two considerations in the choice of the imaging technique. The first is the total number of combinations, which corresponds to the amount of data that needs to be stored and the number of calculations that need to be performed to form the image. The second is the number of transmissions (corresponding to non-empty columns of Figure 16 geometries). The more transmissions, the longer the test takes to complete.

Regardless of the transmit-receive geometry, the imaging process is similar. First let us introduce the process in the time domain. The image for any given point *i*, whose coordinates are *x* and *z,* is formed by delaying each of the transmitter-receiver traces by the amount of time it would take for the wave to travel from the transmitter to the point *i* and from the point *i* to the receiver. The image intensity at the point *i* is then the summation of the magnitudes of those traces. Mathematically it can be represented as:

$$I(x,z) = \left\| \sum_{tx} \sum_{rx} s_{tx,rx} \left(\frac{d_{tx}(x,z)}{v_1} + \frac{d_{rx}(x,z)}{v_2} \right) \right\| \tag{19}$$

$$d_{tx}(x,z) = \sqrt{(x_{tx}-x)^2+z^2} \tag{20}$$

$$d_{rx}(x,z) = \sqrt{(x_{rx}-x)^2+z^2} \tag{21}$$

d_{tx} is the distance between the transmitter tx and the point of interest, d_{rx} is analogous for a receiver. v is the group velocity of the travelling wave. The subscripts *1* and *2* signify the potentially different velocity of the wave travelling to the defect and back if mode conversion occurs.

In all the imaging methods the geometric approach is derived directly from the pipe-plane analogy. However, for this assumption to hold it is necessary to account for the ability of a wave to travel around the pipe. To account for it, an extra boundary condition is enforced for that the solutions on the edges of the plate must match.

For SAFT the transmitter and receiver position are the same, therefore the indices of two summations increment simultaneously. For CSM only a single transmission is used, therefore the summation over tx is unnecessary. TFM uses the full formula.

In the case of dispersive waves, the algorithm needs to be extended to compensate for dispersion. While it is possible in the time domain, it is more commonly done in the frequency domain. The equation expressed in the frequency domain is:

$$I(x,z) = \left\| \sum_{tx}\sum_{rx}\sum_{\omega} S_{tx,rx,\omega} e^{ik(\omega)\left(d_{tx}(x,z)+d_{rx}(x,z)\right)} \right\| \tag{22}$$

In this case, the signal is dependent on the transmitter, receiver and the frequency, with the dispersion relationship expressed in the $k(\omega)$ term, corresponding to the wavenumber dependency on frequency. The formulation above assumes the same dispersion relationship on both the path towards the point and back. In the case of mode conversion, the above formula is transformed to:

$$I(x,z) = \left\| \sum_{tx}\sum_{rx}\sum_{\omega} S_{tx,rx,\omega} e^{ik_1(\omega)d_{tx}(x,z)}\, e^{ik_2(\omega)d_{rx}(x,z)} \right\| \tag{23}$$

Both the probing and reflected signals can consist of multiple modes, converting the formula to:

$$I(x,z) = \left\| \sum_{m}\sum_{n}\sum_{tx}\sum_{rx}\sum_{\omega} S_{tx,rx,\omega,n} e^{ik_{\mathrm{m}}(\omega)\left(d_{tx}(x,z)\right)}\, e^{ik_n(\omega)d_{rx}(x,z)} \right\| \tag{24}$$

where n corresponds to the mode the probing signal is converted into and m corresponds to the mode of the probing signal.

As mentioned in the previous section, the dispersion curve can be calculated using numerical software such as DISPERSE, which is however impractical in inspection scenario, as the dispersion curves are specific to the geometry and the material of the medium. The work of Davies [28] uses the unrolled pipe – plate analogy derived in [29] (Figure 17) to calculate the dispersion curves for the T(0,1) mode family based on the wavenumber of the torsional mode. It is important to note that despite unrolling the pipe, the algorithm considers the edges of the plate linked by enforcing a boundary condition on the solution, thus the flexural modes are not lost in this approximation.

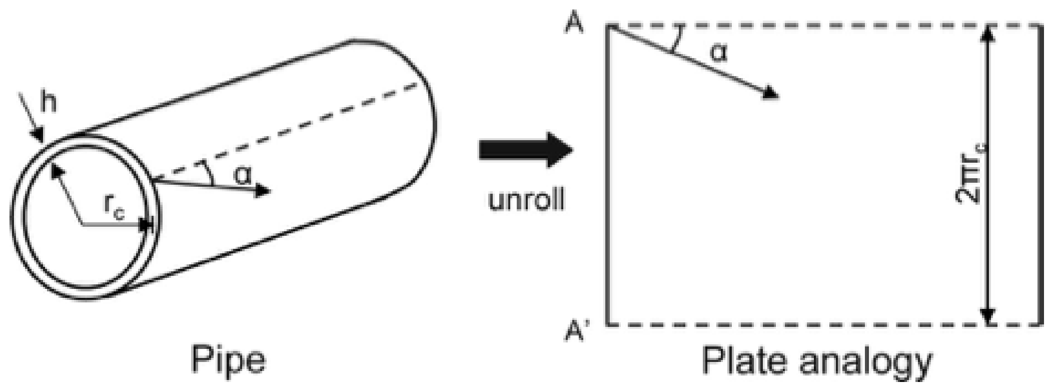


Figure 17. Schematic of the unrolled pipe-plate analogy. Reproduced from [28].

In this reading, the wavenumber can be split into the axial wavenumber (k_z) and circumferential wavenumber (k_{circ}), with the first corresponding to the wave propagation along the pipeline and the second to its circumferential travel. As such, the torsional mode can be understood to travel on a straight path, the first flexural mode (F(1, 2)) is torsional mode travelling on a helical path that has circled the circumference of the pipe once, F(2, 2) has circled it twice etc. As such, the wavenumbers are mathematically linked by the Pythagorean theorem:

$$k_n = \sqrt{k_s^2 - \left(\frac{n}{r_c}\right)^2} \tag{25}$$

Where k_n is the wavenumber of nth flexural mode, k_s is the wavenumber of T(0,1) and r_c is the radius of the pipeline. Davies compares the results of this simplified approach to full DISPERSE calculation, with the results presented in Figure 18. Numerically, the errors are below 2% for schedule 40 3-inch pipes and above, making this a simplified way to calculate the dispersion curves for flexural modes with acceptable accuracy for imaging for most applications. This may be less suitable for very thick-walled pipes where the curvature becomes significant.

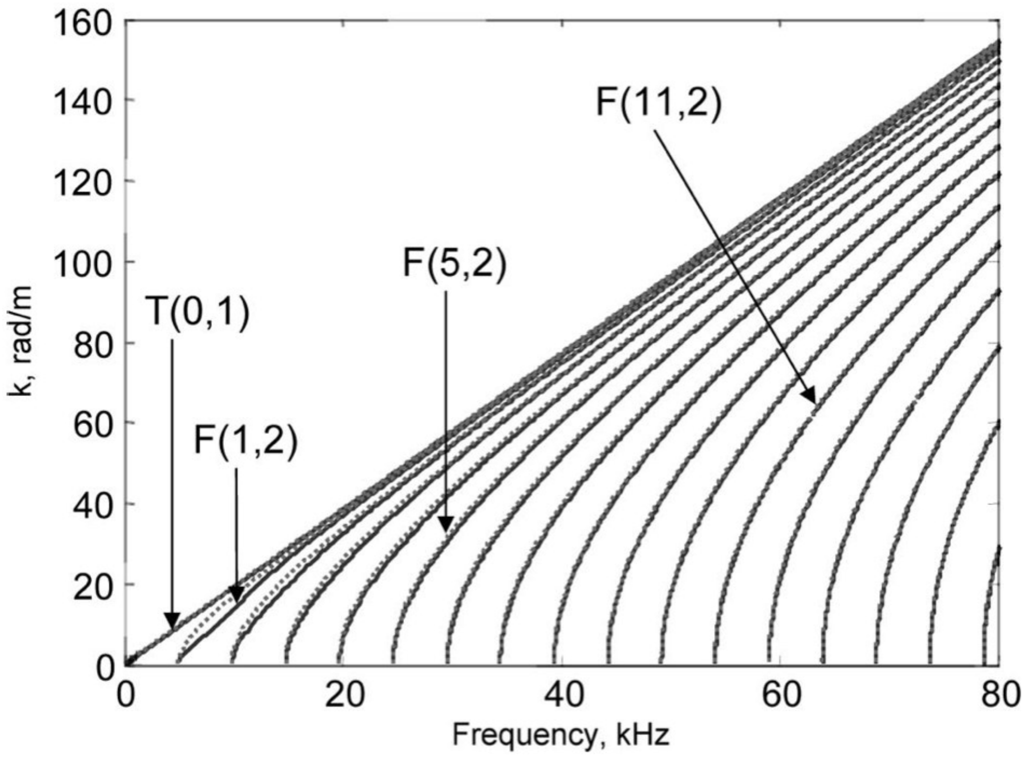


Figure 18. Comparison of dispersion relationships calculated using a pipe-plate analogy (dotted line) and calculated using DISPERSE (solid line). Reproduced from [28].

2.1.3.2 Artificial Focusing Method Selection

The full investigation of the imaging methods in the context of guided wave testing has been conducted by Davies [30]. This section briefly introduces the considerations when selecting the imaging method.

While the goal of any imaging method is its application in an experimental case, it is common to utilise finite element simulations to test the method. The controlled simulation environment allows for a fair comparison of methods by ensuring perfect geometries, no difference in the external conditions or randomness. The work mentioned above has conducted that research for pipe imaging, by first assessing the ability of CSM, SAFT and TFM to image a backwall (a fully symmetrical strong feature). The transducer array is located 1 m away from the end wall and consists of 24 equally spaced transducers. The wave propagated in an 8 inch pipe at the speed of 3260 m/s.

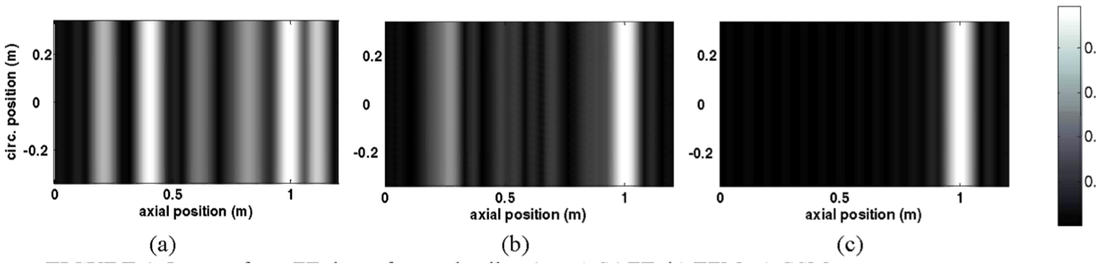


Figure 19. Image from a backwall using SAFT(a), TFM(b) and CSM(c), reproduced from [30].

Figure 19 shows the difference between the imaging methods in the terms of symmetric features. There are clear noise bands present in figures (a) and (b). Those are due to the single transducer transmission in both SAFT and TFM. When a single transducer fires, the wave propagates both in the axial and circumferential direction equivalent to a point excitation of a circular wave in a plate. The circumferential component, travels around the transducer ring and is registered by each receiver in turn. This signal is decoded by the algorithm in the form of the noise bands clearly present in both SAFT and TFM. CSM does not exhibit the same behaviour, because with all transducers firing simultaneously, a plane wave (being the superposition of circular waves) is excited instead of a circular one. The plane wave has no circumferential component, thus removing this source of noise.

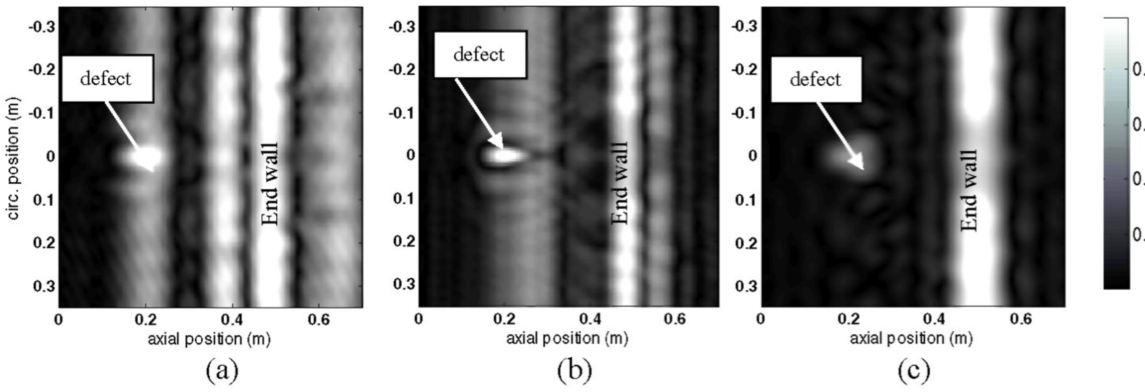


Figure 20. Images from FE data of a 5% reflector at 0.2m and end wall at 0.5m generated using SAFT(a), TFM(b), CSM(c). Reproduced from [30].

Figure 20 presents a follow-up, the performance of the algorithms on a non-axisymmetric reflector – an approximation of a defect indication. In this case, the noise bands present in the symmetric case are still visible and they occlude the image of the reflector in the case of TFM and SAFT. However, CSM focus on the reflector itself is weaker, as it is the only method which transmits a plane wave, which cannot be focused on transmission. Thus, TFM and SAFT have a better focusing performance at the cost of added noise. Davies and Cawley conclude that in the light of the investigation, CSM is the best algorithm for guided wave imaging of pipelines.

Following the study presented, CSM is the algorithm used in this work. In practice, due to the need for generating large numbers of simulations for the purpose of machine learning training, CSM has the added benefit of requiring just one simultaneous excitation of all transducers, reducing the required number of simulations – in SAFT or TFM each transducer excitation necessitates an individual simulation, multiplying the need for computational resources.

2.1.3.3 Direction Control

A row of ultrasonic transducers excites a wave that propagates in both positive and negative directions along the pipe, see Figure 21. For the task of localisation of features, it is crucial to separate the two directions. To that effect two offset rows of transducers are used.

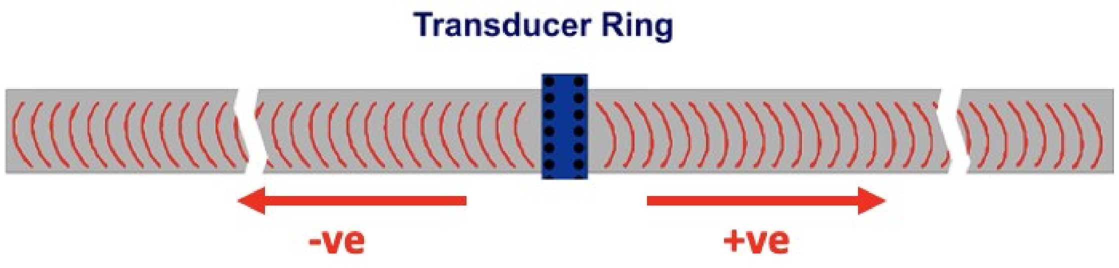


Figure 21. Transmission of guided wave in both directions of the pipeline. The wave propagation directions are shown as red arrows, the area inspected is shaded in grey.

The physical phenomena used are constructive and destructive interference (Figure 22).

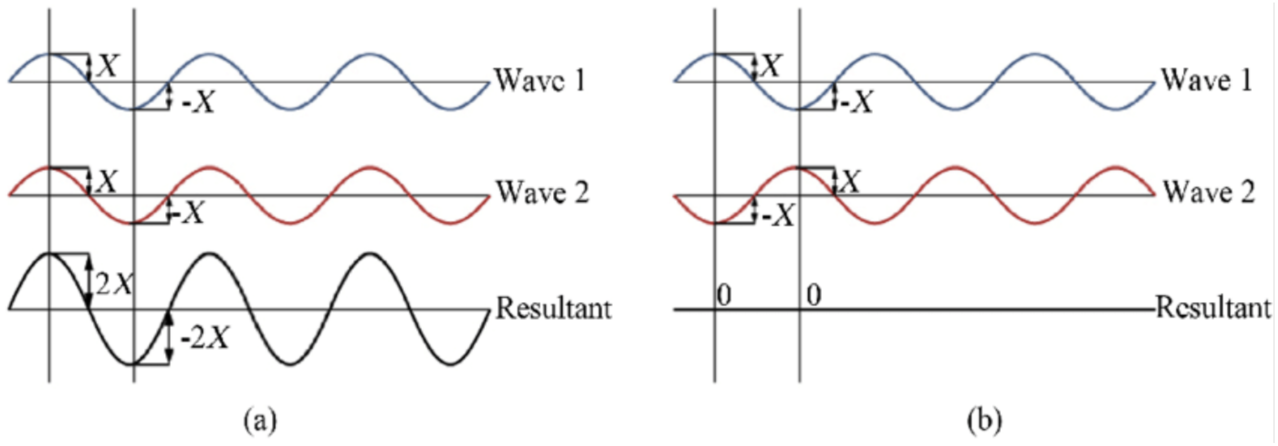


Figure 22. Constructive (a) and destructive (b) interference of waves. The top two waves are the constituent waves, the bottom is the result of superposition of the waves. Reproduced from [31].

Constructive interference, effectively doubling the amplitude, occurs when the waves are in phase, while destructive occurs when the waves are out of phase by half of the period. Full constructive and destructive interference may occur only when the interfering waves are identical.

Considering the guided wave testing, the phase separation of the signals from positive and negative directions can be achieved by utilising two rows of transducers. The transducers are fitted on a single testing ring. The schematic representation of the setup is presented in Figure 23. In a setup like this, the ring A fires separately from ring B. In each case both the ring A and ring B receives the signal. As such, after the test is run, four traces are available: A-A, A-B, B-A and B-B. A-B and B-A traces should be identical due to reciprocity. We can now consider the implementation of the direction control using the signals described.

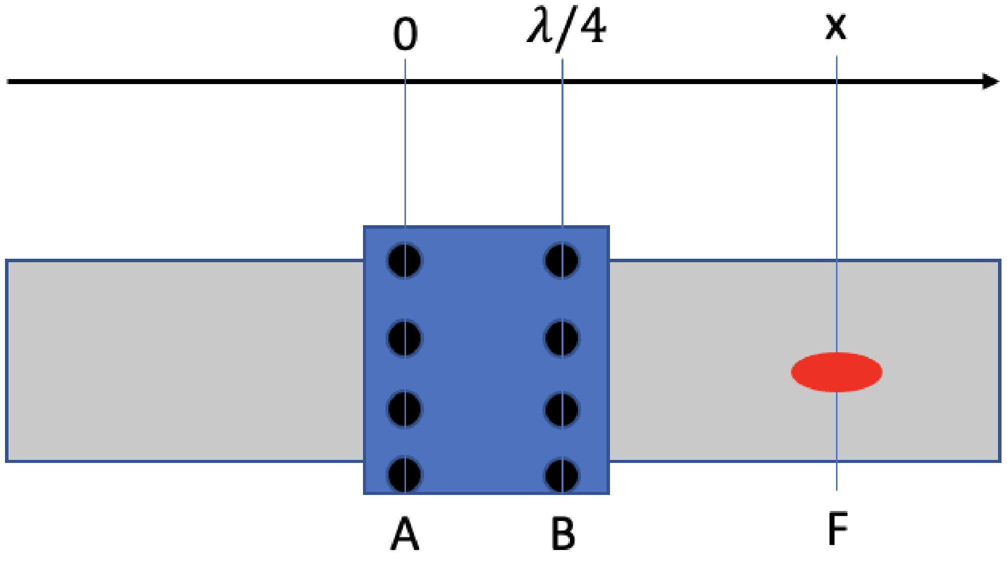


Figure 23. Pipe with a guided wave testing ring fitted with two rows of transducers (A and B) separated by quarter wavelength. Feature F exists at a distance X from row A.

Let us first consider the propagation distance from each of the rings to the feature, accounting for both the forward and backward path.

$$d_{A-A} = d_{AF+ve} + d_{AF-ve} = x + x = 2x \tag{26}$$

$$d_{B-B} = d_{BF+ve} + d_{BF-ve} = x - \frac{\lambda}{4} + x - \frac{\lambda}{4} = 2x - \frac{\lambda}{2} \tag{27}$$

$$d_{A-B} = d_{AF+ve} + d_{BF-ve} = x + x - \frac{\lambda}{4} = 2x - \frac{\lambda}{4} \tag{28}$$

In these formulations λ refers to the wavelength at the centre frequency. The transducer rings are separated by quarter of the wavelength at centre frequency as it causes the A-A distance and B-B distance to be separated by half of the wavelength (this approach means that the non-centre frequencies are not completely compensated, but for narrow-band signals in guided wave pipe imaging this effect is negligible, as other noise sources are significantly stronger). This would correspond to half-period phase shift, thus adding those two traces up results in the destructive interference, thus muting the direction. The traces can be phase-shifted using Fourier analysis, so A-B traces can be used to further amplify the correct direction and mute the opposite. This is especially important in the experimental context, as the signals are not identical and shifted due to the differences in transducer coupling, conditions, and other sources of noise. The phase shift is best performed in the frequency domain. In the guided wave context, the phase shift should be performed separately for each mode, as the propagation speed is different in the way described by the differing wavenumbers. The following presents a full mathematical formulation for the direction separation of positive and negative directions in frequency domain:

$$s_{+ve} = s_{AA} - s_{AB}e^{ik_0 * \frac{\lambda}{4}} - \left(s_{AB} - s_{BB}e^{ik_0 * \frac{\lambda}{4}}\right)e^{ik_n * \frac{\lambda}{4}} \quad (29)$$

$$s_{-ve} = s_{AA} - s_{AB}e^{-ik_0\frac{\lambda}{4}} - \left(s_{AB} - s_{BB}e^{-ik_0\frac{\lambda}{4}}\right)e^{-ik_n\frac{\lambda}{4}} \quad (30)$$

where s_{AA} is the frequency-domain signal, k_0 is the dispersion relationship (wavenumber vs. frequency) of the transmitted mode, k_n is the dispersion relationship of the received mode. The modes can be separated on reception utilising the variation in the signals from different transducers (up to N/2 modes for N transducers). These formulae need to be recalculated for each of the modes to be direction controlled and for each inspection frequency. The result of the calculation is a reinforcement of the mode in the desired direction: positive for $eq. 29$ and negative for $eq. 30$.

2.1.4 Guided Wave Testing Hardware

The utilisation of guided waves for pipeline inspections depends on the capability for ultrasonic transduction. There are several methods of exciting ultrasonic waves in metal, including piezoelectric [32], magnetostrictive [33], and laser-induced ultrasound [34]. This section focuses on the piezoelectric transducer systems as utilised by Guided Ultrasonics, as it is the technology utilised in this work [32].

Alleyne and Cawley have developed a dry coupled system of transducers placed in two rows to facilitate direction control. The dry coupling requires placing the transducers tightly on the surface of the pipe. For small diameter pipes, solid rings are utilised (Figure 24). These are pushed onto the pipe using springs.

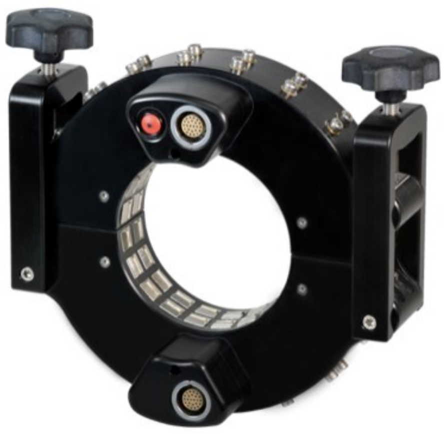


Figure 24. Guided Ultrasonics EFC Solid Ring [8].

For large diameter pipes the solid rings are unwieldy, therefore inflatable rings are used, with the pneumatic force used for coupling (Figure 25).

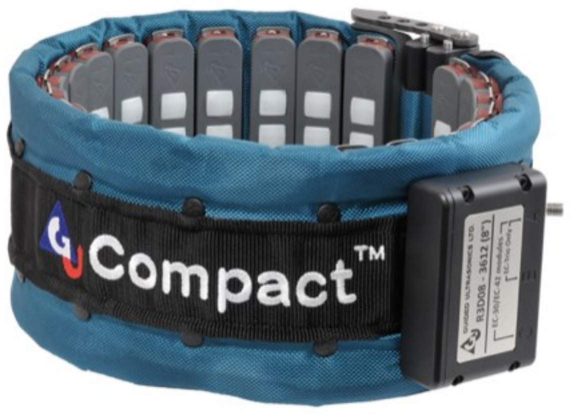


Figure 25. Guided Ultrasonics Compact Ring [8].

In either case, the casing is fitted with a ring of shear-polarised piezoelectric transducers – this ensures they act with tangential force on the surface of the pipe. The transducers are orientated in the circumferential direction, which ensures they excite the T(0, 1) mode family. Were the transducers orientated axially, the L(0, m) mode family would be utilised [4]. The number of elements in the ring n is limited by the highest order of the flexural mode (F(n, 2)) intended to be received. Typically, more than n transducers are used to ensure higher energy of the initial pulse and improve the force distribution around the pipe. The adjoining transducers can be grouped together with their results added up and treated as a single transducer in post-processing to limit the number of channels.

2.2 Machine Learning

2.2.1 History and development of Machine Learning

Machine learning is a set of methods for making decisions based on experience related to computer science and control theory. Historically speaking, machine learning was first introduced in the 1950s and the 1960s. Commonly this advent of machine learning is associated with a Cornell University psychologist, Frank Rosenblatt. His research group developed machines (perceptrons) that could recognise the letters of the alphabet [35]. The concept for those machines is based on the models of human brain and learning developed in the early 1950s by Bush and Mosteller [36]. Rosenblatt's device sparked the first surge in the development of machine learning, mostly centred in the cybernetics community, most interestingly with the scientists at Leningrad State University proposing a prototype of Support Vector Machine (SVM) in 1964 [37]. This first round of the development was halted after the 1969 publication of "Perceptrons: An Introduction to Computational Geometry" by Minsky and Papert. The authors have proven that perceptron networks are unable of representing certain logical functions, such as XOR (logical exclusive or). As digital logic was the focus of research at the time, the book has triggered a decrease in funding and research activity into learning algorithms. The 1970-1980 period is therefore known as the first winter of AI.

The 1980s brought about the second generation of machine learning research, heralded by Kunihiko Fukushima and his proposal of neocognitron [38]. Neocognitron is, in essence, multiple stacked perceptrons, something that we would now recognise as a neural network. At this point the pieces were falling into place for the modern machine learning approach. This was further aided by the invention of backpropagation – a paradigm for efficiently calculating derivatives and training neural networks led by Rumelhart in 1986 [39]. Even though backpropagation was not immediately accepted as the standard, by the late 1980s there was a strong expectation for continued success of ML on the back of the technique and the advances in computational power. Those expectations failed to be realised, leading to the loss of interest in the 1990s, known as the second winter of AI.

In the early 21st century a three-pronged revolution has occurred, which made ML the topic it is today. The first prong was the inception of Big Data as a concept. The globalisation and digitisation of public and private databases brought about the data revolution – the necessity for the development of statistical and operational methods to deal with large datasets. Currently, it is considered self-evident that big data is a necessity for machine learning, making that development a key enabling technology. The second prong was the parallel computing and memory cost revolution. In 2004 Google revealed its MapReduce technology [40], designed to split the computational effort of a complex problem between many simple processors. This has an obvious application to backpropagation problem. From that point the remaining issue was the nature of the small processors between whom the computation could be split. The answer turned out to have come from the gaming industry, where Nvidia has developed CUDA [41] – a computing platform allowing for the use of graphics processing units (GPUs) in general computational tasks. As GPUs are essentially a large number of small processing units designed for parallel operation, they have proven ideal for machine learning training. The third prong of the revolution was the development of new machine learning algorithms. Based on Fukushima's work, multilayer perceptrons became the standard in machine learning research. The research was centred around Geoffrey Hinton, Yann LeCun and Yoshua Bengio, whose activity has culminated in the publication of "Deep Learning" in *Nature* in 2015 [42]. This has symbolically started the modern era in machine learning, in which the innovations no longer follow a conventional academic process but are a combination of research by companies such as Meta, where Yann LeCun is the chief scientist, Google, where Geoffrey Hinton worked until 2023, before quitting over AI ethics concerns, and Baidu, who have taken on Andrew Ng from MIT. The fast paced and commercial nature of the research has led to the modern ML research to be most commonly disseminated at conferences such as NeurIPS and ICML as well as in white paper style publications. Since then, the major developments in the ML research included technologies such as generative learning [43], attention mechanism [44] and its generalisation to a transformer network [45], recently joined in GPT (generative pre-trained transformer) architecture [46] and ChatGPT released by OpenAI [47]. Independently, the reinforcement learning algorithms [48] have been developed by the likes of DeepMind, most famously using them to build AlphaGo model [49], which has managed to beat a professional human Go player. Recently they have published self-training architectures, requiring no input apart from rules of the game to be learned, such as AlphaGo Zero [50] and its generalisation, AlphaZero [51].

2.2.2 Supervised and Unsupervised Learning

There are two main categories of machine learning algorithms, which correspond to the broad way in which they are trained. These are supervised and unsupervised learning algorithms. The distinction dates back to the beginnings of modern machine learning, with many modern applications blurring the line between the two or utilising both approaches to solve segments of a larger task. The interested reader is directed to the recent textbook by Berry, Mohamed and Yap, which comprehensively elucidates the subject [52].

Supervised learning algorithms' name comes from their training method similar to traditional school teaching, where learning is done under supervision. The student answers the question and then is provided the correct answer. Based on whether the student's initial answer is correct or not, they would be more or less likely to use the same method when they are presented with a similar problem in the future. Before graduating, the student must prove they can perform on these preparatory questions, pass an exam and only then they are trusted that they would be able to deal with problems in the real world. The process in graphically represented in Figure 26.

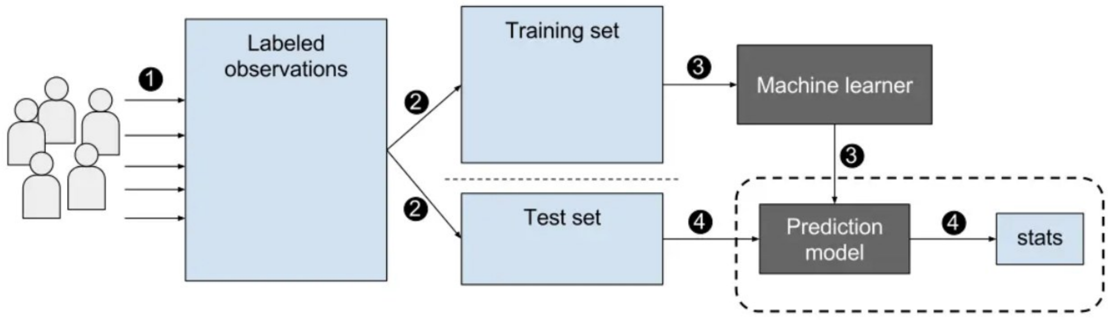


Figure 26. Supervised Learning Schematic Blue boxes represent the sets of data, dark grey boxes represent the processing algorithms. Reproduced from [53].

In the machine learning domain, the 'preparation' questions are referred to as the training dataset, the 'exam' is the test dataset. Those two datasets are selected from the same set of labelled observations. Crucially, for that dataset to exist, the observations must be labelled.

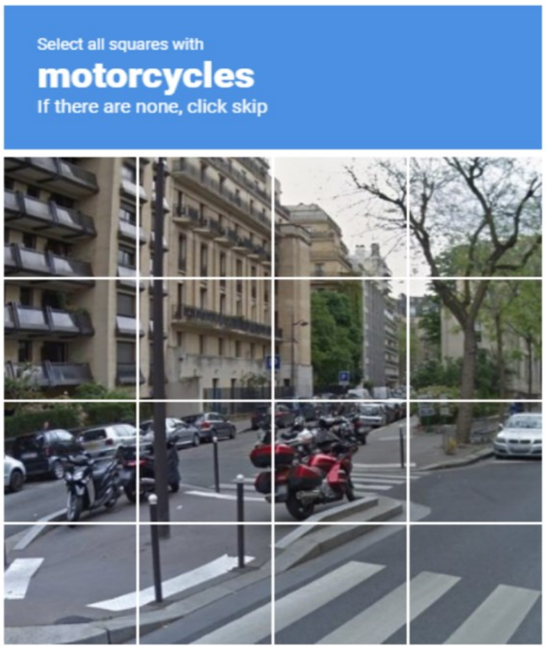


Figure 27. Google image labelling task. The user is asked to select the portions of the image containing the object of interest. Such tasks are commonly encountered as a security measure on websites [54].

While for some questions it is a simple matter, such as the omnipresent image labelling tasks (Figure 27), the highly skilled tasks such as labelling medical images require subject matter experts to provide labels, which is a significant challenge in many machine learning projects.

Supervised learning is used to solve two types of problems – classification and regression. In classification problems, the algorithm's output is an answer to the question of whether the input data belongs to a certain class. The common example is inputting a picture of a cat or a dog and scoring the algorithm on its accuracy in telling the difference between the animals. The regression problems, meanwhile, have the outputs on a continuous scale; they are answering a question of what the value of Y is for a given value of X (or multiple Xs). A typical example is based on the house price dataset, where an ML algorithm is asked to predict the price of a house based on its age, size, number of bedrooms etc [55].

When a labelled dataset is not available or when the question, we need answered is one we do not know an answer to, unsupervised learning is utilised instead. Unsupervised learning is based on the natural process of learning when there are no predefined correct and incorrect solutions. A real-life scenario would be toddler playing with their toys and putting them together based on various characteristics, such as size, colour, material etc. The toddler is not told explicitly that a red car goes with a red brick, but they understand those are in some way similar.

Clearly, unsupervised learning has a different range of use cases, as without knowing the true answer it doesn’t know the question it is being asked. For example, going back to the cat vs. dog classification, it could instead classify the images based on whether they are outdoors or indoors. This leads to the main use cases for unsupervised learning [56]:

- Clustering – the algorithm puts together the samples that are in some way similar, based on the features it has internally extracted.
- Anomaly Detection – the algorithm learns that all the data it is presented in the training process is ‘normal’ and after deployment it is supposed to flag any data that would not fall into that category. This is commonly used by banks to detect fraudulent credit card transactions (i.e., a transaction for a luxury bag on a card that is normally used for groceries would be flagged as anomaly).
- Association – This is most used in recommender systems and is closely related to clustering. The everyday application is online shopping. When the user puts knives, pans and chopping boards in your cart, the algorithm could recommend a garlic press and a cookbook.
- Autoencoders – an unsupervised learning algorithm that is designed to compress the input data, then decode it and still be able to recognise the original data. Conceptually, if an algorithm does that, it removes the unnecessary information in the encoding process. This is most useful in signal and image processing areas, as a good autoencoder can remove the noise from the image while retaining all the pertinent data.

Importantly, when using unsupervised learning algorithms, there are no objectively ‘correct’ or ‘incorrect’ decisions. As a result, much as in the case of human decisions, the external performance metrics must be introduced. As an example, a recommender system may be rated on how many customers have added additional items to their shopping cart, compared to an alternative recommending solution.

2.2.3 Neural Networks

This section presents the fundamentals of the neural network design and its constituent parts, as such, much of the information is considered self-evident in the machine learning community. The interested reader is directed to two online courses taught by Andrew Ng – one of the foremost AI researchers:

- Machine Learning Specialisation – Stanford University and Deeplearning.ai [57]
- Deep Learning Specialisation – Deeplearning.ai [58]

Neural Networks are a type of a machine learning algorithm built of identical basic units calculating simple mathematical functions. Those units are connected to each other with connections of a weight learnt during the training process. The neurons are arranged in layers, with the outputs of neurons of layer number *i-1* acting as inputs to neurons of layer number *i*. The decision on how many neurons to use, in what arrangement and which connections between neurons to allow is the basic question on a neural network design.

2.2.3.1 Neuron Operation

A neuron is a function with any number of input parameters and a single output value, so it can be written as $y = f(x_1, x_2, x_3, \dots x_n)$, where *f* is the neuron function. Typically, the inputs are represented as a vector of values of length n. From now on, that is the convention taken in this chapter.

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_n \end{bmatrix} \tag{31}$$

The neuron performs a simple operation, in which each input is multiplied by a parameter called *weight (w)*, whose product is added to another parameter, called *bias (b)*. Both of these parameters are learnt in the training process of the neural network. Each input has its own weight, signifying the strength of the connection between the input neuron and the operating neuron, while bias is common to all inputs across the neuron. Following the convention for inputs, weights also take the form of a vector:

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \dots \\ w_n \end{bmatrix} \tag{32}$$

Therefore, for a single-valued input $\boldsymbol{x} = x_1$, the function is:

$$y_1 = f(x_1) = w_1 * x_1 + b \tag{33}$$

For a non-one length input, the output is a summation of outputs for each element of input.

$$y = y_1 + y_2 + \cdots + y_n = (w_1 x_1 + b) + (w_2 x_2 + b) + \cdots + (w_n x_n + b) \tag{34}$$

Which can be written in a simplified form as a dot product of w and x:

$$\boldsymbol{y} = \boldsymbol{w} \circ \boldsymbol{x} + b \tag{35}$$

2.2.3.2 Activation Functions

The problem inherent in the simplicity of the mathematics behind the building block of a neural network is the fact that it is a linear function. As a result, even a complex neural network would in essence boil down to a combination of linear functions, which by definition is a linear function. Without introducing nonlinearity at some point in the neural network, the outputs would simply be a linear function of inputs, effectively corresponding to a single layer. The nonlinearity is introduced using activation functions, mapping the output of the neuron operation in a non-linear fashion.

This section provides an overview of the most used activation functions: linear, sigmoid, tanh, ReLU and leaky ReLU.

Linear activation function is simply a linear mapping of the output of the neuron to the output domain. This type of an activation function does not provide nonlinearity; therefore, it cannot be the only type of activation function used. If used, it is most commonly used in the output layers in regression problems, such as the classical house price prediction.

Sigmoid activation function (also known as logistic activation function) is mathematically described as $f(z) = \frac{1}{1+e^{-z}}$. The shape of the function is visible in Figure 28.

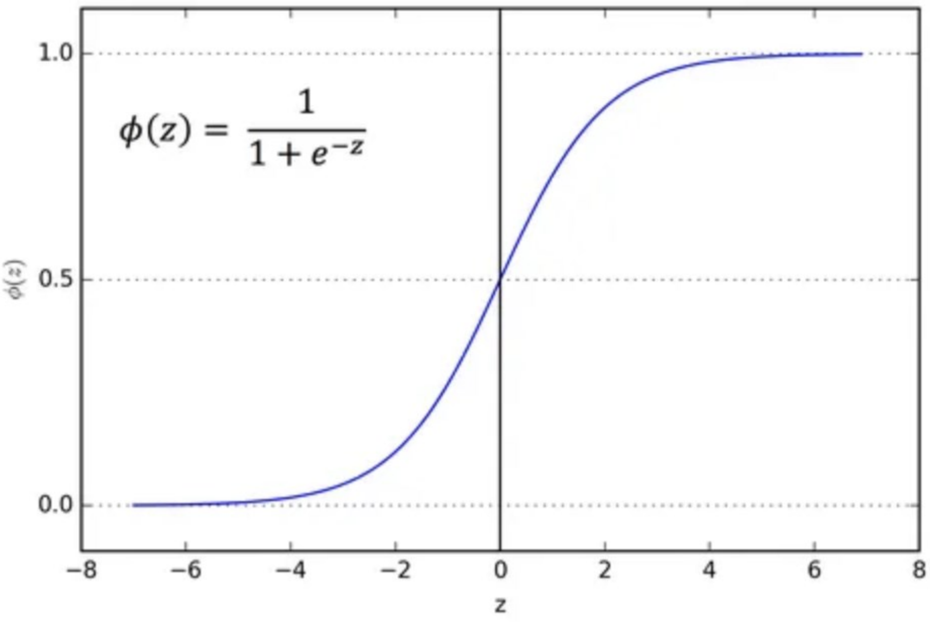


Figure 28. Sigmoid Function Graph.

Visibly, the output of sigmoid is between 0 and 1, is monotonic and differentiable. These characteristics are important, as the monotonicity ensures the higher output of the neuron corresponds to higher output of the activation function, and the function needs to be differentiable for the training process to be possible, as it relies on calculating the derivatives of the outputs. As the output of the sigmoid function is between 0 and 1 it is commonly used as output in the probability prediction tasks, as any probability exists between those values. The version of sigmoid generalised to multi-class classification problem is a softmax function. While sigmoid can be understood to output the probability of a single class based on the input value, softmax outputs the probabilities for a larger number of classes. The main drawback of sigmoid activation function is the flattening of the function in the higher negative and positive ranges. As the updating of neuron parameters is based on the derivatives of the outputs (i.e., the local differentials of the activation function) the training would become very slow if the output of the neuron happens to be high. This is known as the vanishing gradient problem.
Tanh activation function is very similar to sigmoid. Mathematically it is a hyperbolic tangent of the input value, $f(z) = \tanh(z) = \frac{2}{1+e^{-2x}} - 1$ tanh and sigmoid are functions which can be derived from each other. Figure 29 shows the comparison between the tanh and sigmoid functions.

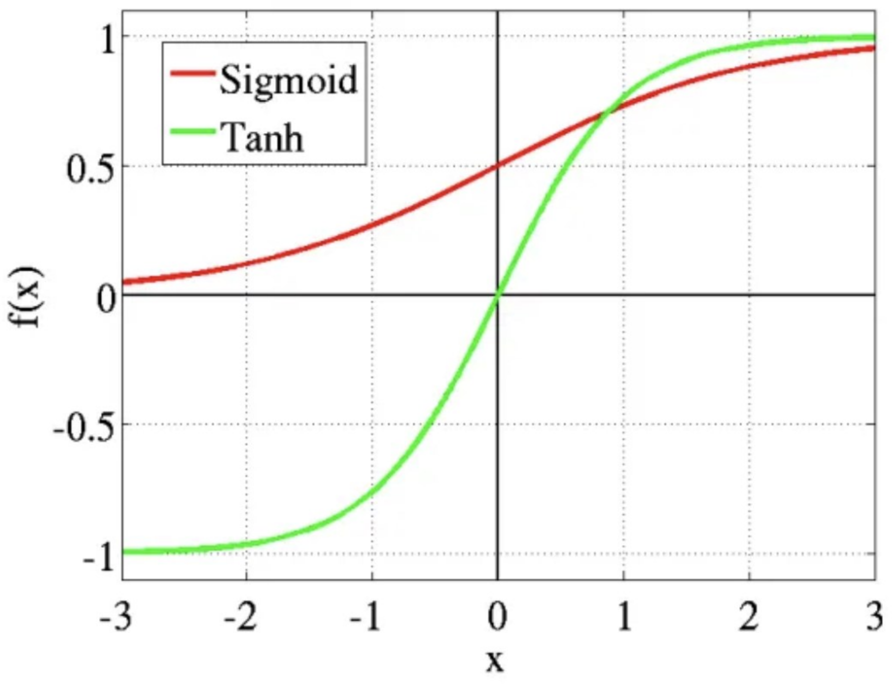


Figure 29. Tanh and Sigmoid graphs comparison.

The main difference in sigmoid and tanh is the output range, as tanh outputs between -1 and 1. This is useful in the generative tasks, in which the output should be symmetrical in the positive and negative domains, such as direct waveform audio generation. Secondly, the fact that tanh has the mean of 0 is useful in the training of the hidden layers of the model. In the past, it has been used as the main activation function for the hidden layers, but it has now been superseded by ReLU.

ReLU, shorthand for Rectified Linear Unit, is mathematically defined as $f(x) = \max(0, x)$

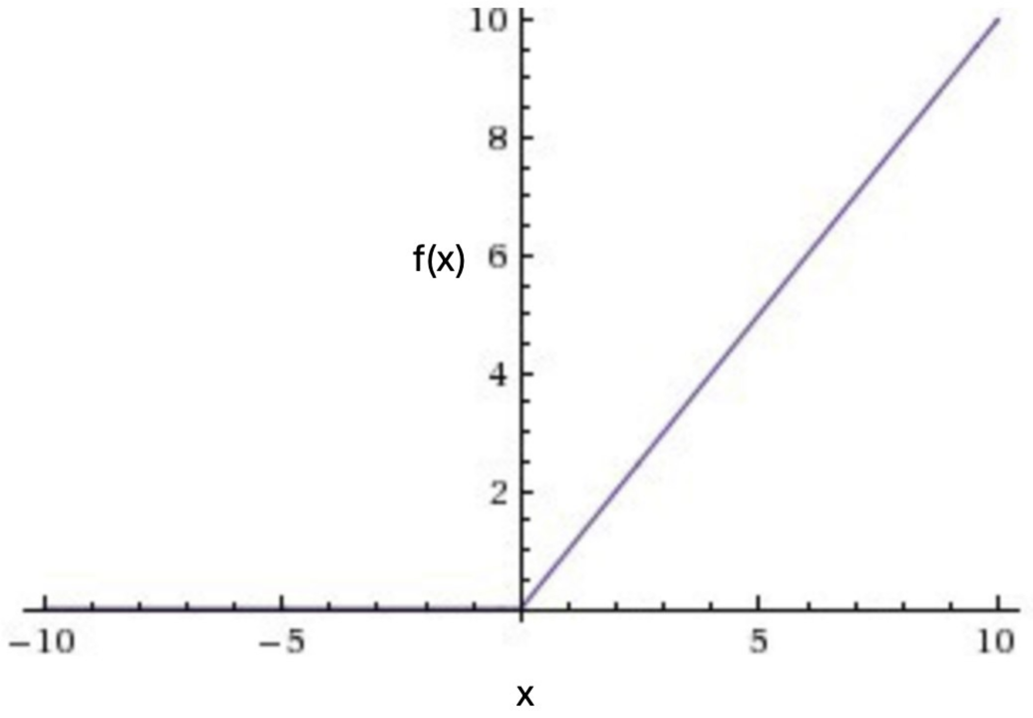


Figure 30. ReLU graph.

Figure 30 represents the graph of RelU function with the $f(x)$ values at 0 when $x < 0$. When the argument becomes greater than zero, the value of the function is equal to the argument. It is a very simple function and is currently the most commonly used activation function for the hidden layers. For most practitioners it is the base case when building an experimental neural network. Compared to sigmoid, its characteristics are the significantly lower computational cost, which was historically important in the research environment, and remains relevant nowadays in the multi-billion parameter models used in production, secondly, it does not suffer from the vanishing gradient problem in the positive range, as the differential is constant. The issue with ReLU is that any negative input maps to 0, which prevents learning from the input data across the full domain (i.e., -1 has the same impact on the network as -1000). While this can be solved by careful pre-processing, the more common approach is to use Leaky ReLU, which solves the so-called dying ReLU problem programmatically.

Leaky ReLU is an activation function very similar to ReLU, but instead of using 0 as the lower boundary it uses a different linear function of input, conventionally $y(z) = \max(az, z)$, where *a* is conventionally set to 0.01.

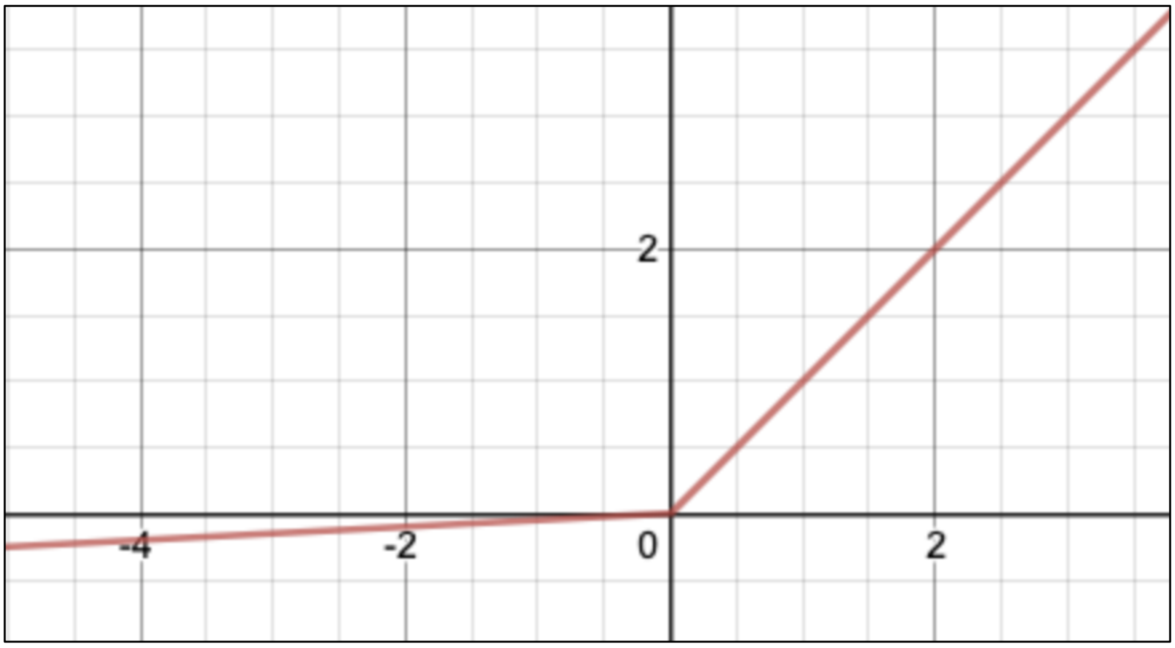


Figure 31. Leaky ReLU graph.

Leaky ReLU shares the characteristics with ReLU, but it can learn from the negative as well as the positive data. It is slightly more computationally expensive, but still less so than the logistic functions. As a result, Leaky ReLU is more reliable than the base version, at the cost of speed, therefore it is commonly used as the hidden layer activation in deep architectures, or when the input data cannot be guaranteed to be clean and well normalised.

2.2.3.3 Layers

A layer in a neural network is a combination of a mathematical operation and a set of values. A layer takes the set of values from the previous layer, performs its operation, takes on the results as its own values and makes them available to the following layers. All the operations inside the layer are independent of each other, therefore they can be parallelised, enabling the use of massive parallel computing processors, such as GPUs. Layers can be understood as the basic building blocks of the neural network, as it is always the inputs to and outputs from layers that the operations are performed on, not the individual neurons. This section introduces some of the common types of layers, understanding of which is crucial to machine learning.

2.2.3.3.1 Fully Connected Layer

As discussed, the full neuron, or the unit of a neural network performs an operation that is mathematically described as $y = \alpha(w \circ x + b)$ where α is the chosen activation function, w is the weights vector, x is the input vector and b is the bias. In the most basic type of a neural network, known as dense or fully connected (FC) neural network the neurons are connected as shown in Figure 32 with each neuron in layer n-1 providing input to all neurons in layer n.

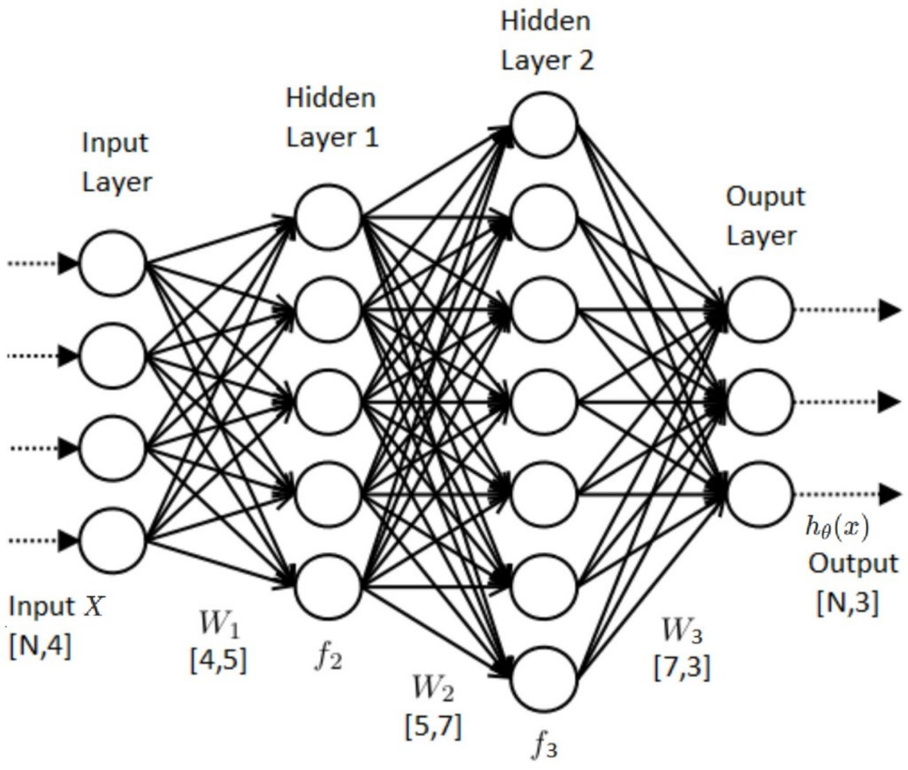


Figure 32. Fully connected Neural Network graph. The values are shown as circles and the operations are shown as arrows. This network does not contain bias. The sizes of weight matrices are annotated under the arrows denoting operations. Reproduced from [59].

The diagram shows why it is more common to talk about neural networks in the units of layers, rather than neurons. Each neuron must produce the number of outputs corresponding to the number of neurons in the following layer. Thus, the previously described weights vector becomes a weights matrix, with the first dimension corresponding to the number of input neurons and the second to the number of outputs. Similarly, the bias becomes a vector of biases with the length corresponding to the number of outputs. Weights (and biases) are therefore best understood not as properties of neurons, but of layers due to their relational characteristic.

2.2.3.3.2 Batch Normalisation Layer

A batch normalisation layer implements its namesake operation. It was first introduced by and Szegedy in 2015 [60]. During training, the distribution of the inputs to the layer shifts, as those inputs are the outputs from the previous layer. When the inputs are far from zero-mean, unit-variance normalised, the training is slower, especially in the case of using activation functions prone to saturation (sigmoid, tanh, to a lesser degree ReLU). Batch normalisation normalises the inputs mid-training process and does so with regard to each mini-batch (the portion of data on which the algorithm is trained in a single step). Batch normalisation layers have been proven to speed up the training and, in some cases, to prevent overfitting, which makes them common in many modern deep learning architectures. Batch normalisation layers are active only during training and deactivated at inference, which is the source of one of the criticisms of batch normalisation – the difference in the behaviour between training and testing.

2.2.3.3.3 Dropout Layer

A dropout layer randomly deactivates some of the neurons in each training step. The purpose of that operation is making sure that the network is not overly reliant on a single neuron, but rather that the predictive power is regularly spread between the neurons. This approach also helps prevent overfitting, thus normalising the network. Similarly to batch normalisation, dropout layers are deactivated at inference.

2.2.3.3.4 Convolutional Layer

A convolutional layer is similar to a fully connected layer, in the fact that it is a layer containing trainable weights. Neural networks based on this type of a layer are collectively known as Convolutional Neural Networks (CNNs), even though most of them contain some fully connected layers.

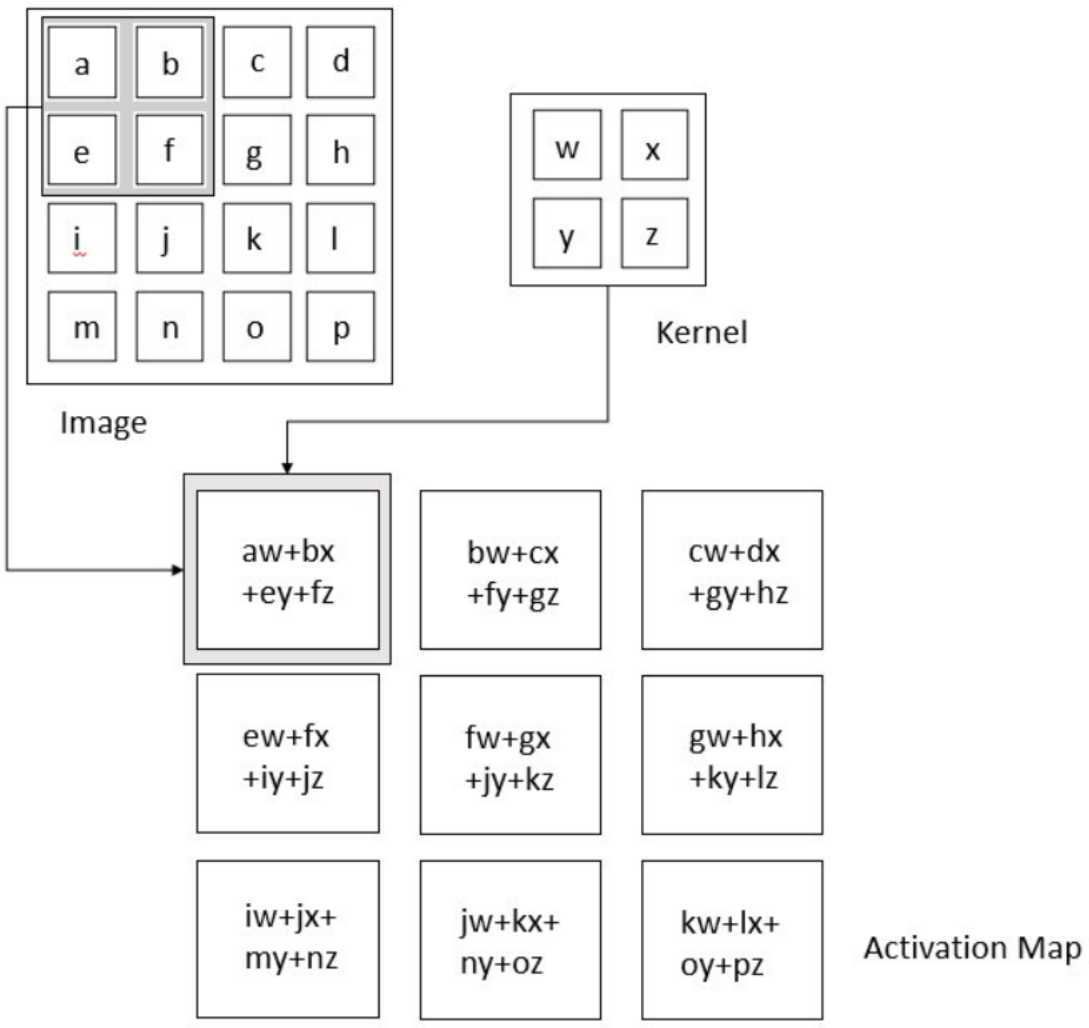
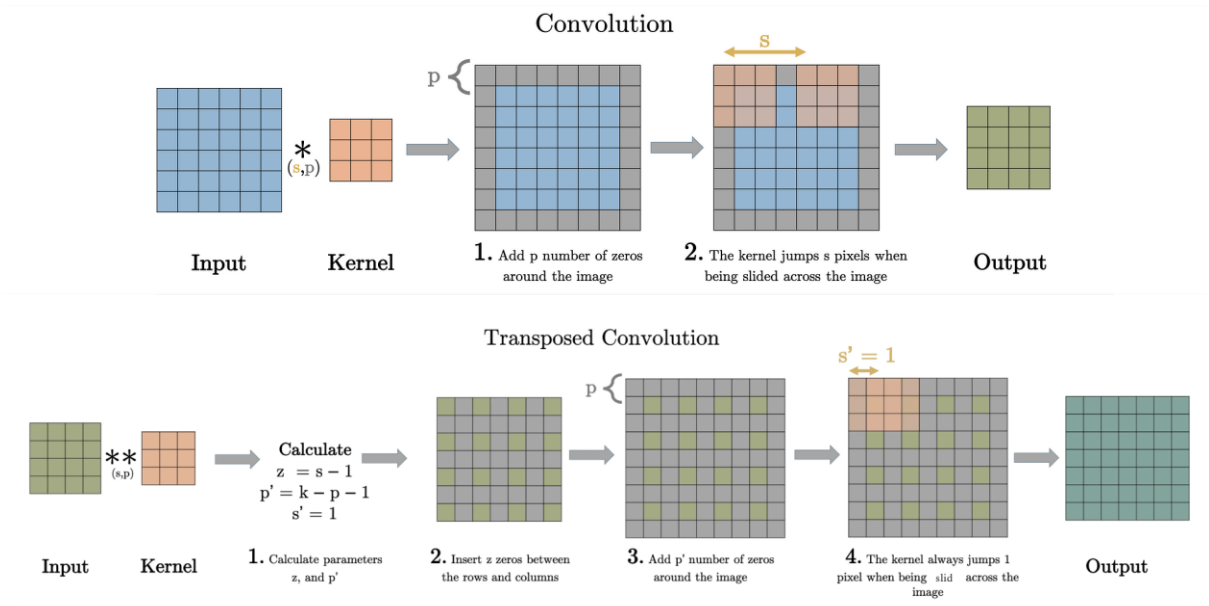


Figure 33. Convolution operation depicting the activation map achieved by applying a kernel to an image. [61].

To understand a convolutional layer, it is first necessary to define its building blocks. First there is the input data, which can have any number of dimensions and channels. In the most common case of analysing RGB images, the data (image) is 2-dimensional and contains three channels – red, blue, and green. The second building block is the kernel, also known as the filter. The filter is smaller than the input data in the non-channel dimensions and has a depth equal to the input data – for the RGB image, the kernel would have the shape of (a, b, 3). A convolutional layer can have any number of kernels, which are equivalent to neuron weights in a fully-connected layer. A convolutional layer performs a dot product between a kernel and a portion of the input data, then similarly to a fully connected layer passes it through an activation function and outputs the result. The kernel is then moved to another position, usually overlapping with its previous one. The parameter that controls the shift in each dimension is known as the stride (s). s = (1, 2) would mean that the kernel moves one sample at a time in the first dimension of the input data and by two samples at a time in the second dimension. Finally, a padding parameter is used to even out the contribution of the data points, adding margins around the borders of the input. In the example of Figure 33, datapoint a is considered only once in the output, while datapoint f is considered four times. To equalise the contributions, padding is used. Moreover, padding allows the size of the output to be controlled, as it is typically desirable for the output from the convolutional layer to have the same non-channel dimensions as the input. Despite a kernel having more parameters than a neuron, the topological limits mean that a convolutional layer has fewer parameters than an equivalent fully connected network.

2.2.3.3.5 Transposed Convolution Layer

A transposed convolution performs an operation directly opposite to a convolution – it multiplies a single number by an (n, m) sized kernel, thus up sampling the input. The values in the kernel are learnt during the training process. Refer to Figure 34 for the visual of a difference between a convolution and a transposed convolution.



Convolution
$*$ (s,p)
p
s
Input
Kernel
1. Add p number of zeros around the image
2. The kernel jumps s pixels when being slided across the image
Output
Transposed Convolution
$**$ (s,p)
Calculate
$z = s - 1$
$p' = k - p - 1$
$s' = 1$
p
s' = 1
Input
Kernel
1. Calculate parameters z, and p'
2. Insert z zeros between the rows and columns
3. Add p' number of zeros around the image
4. The kernel always jumps 1 pixel when being slid across the image
Output

Figure 34. Convolution and transposed convolution. Convolution is shown on top, transposed convolution at the bottom. The sizes of the grids correspond to input and output sizes. Reproduced from [62].

A transposed convolution is commonly used in generative networks, where a small encoding is decoded to a larger data. It is also used in a U-net architecture [63], in which the input data is first compressed to remove the irrelevant information and later decompressed, while reintroducing the original information (see Figure 35).

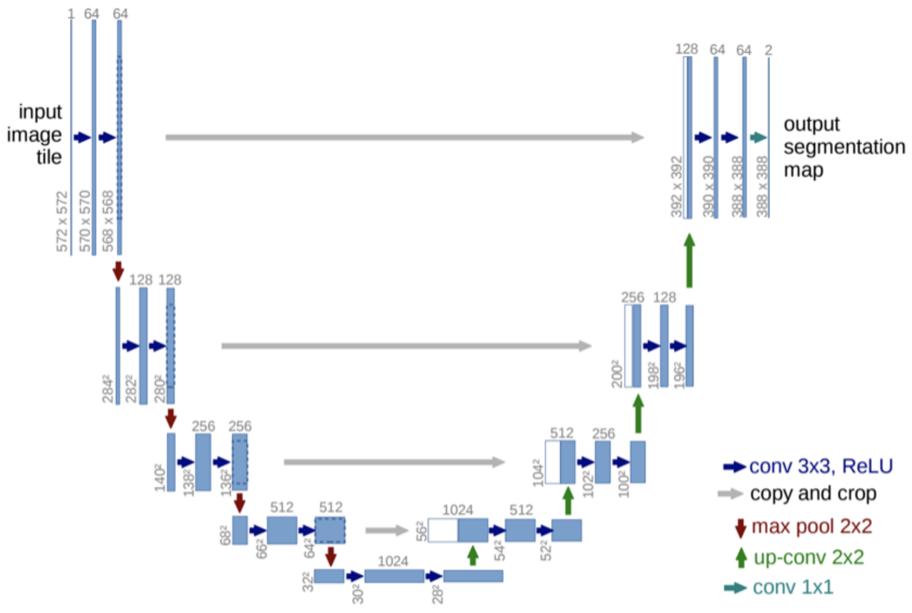


Figure 35. U-net. Output of the first block serves as part of the input to the last, second to block n-1 etc. Reproduced from [63].

2.2.3.3.6 Pooling Layer

A pooling layer is a common operation in a convolutional neural network. This layer uses kernels, similar to a convolutional layer, but instead of a convolution operation, it performs a pooling operation – most commonly a max pooling or an average pooling.

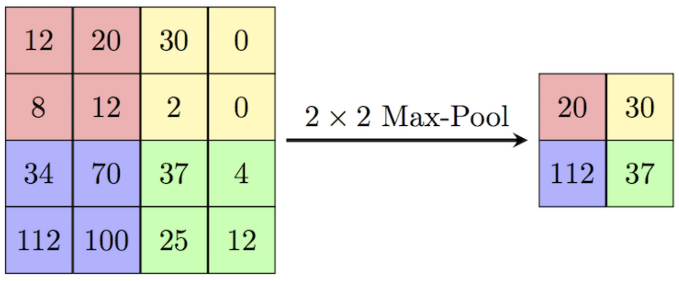


Figure 36. Max pooling. Maximum of the color-coded section of the first block is output to the second block [64].

Max pooling is very well described visually in Figure 36 – the operation takes an n-by-m section of the input and replaces it with the maximum of that section. Average pooling, while less common, follows a similar approach, taking a mean rather than a maximum. The goal of a pooling layer is to make the intermediate data smaller, in the case of max pooling keeping only the biggest (therefore with the highest impact) activations. This on one hand lowers the computational load, allowing for deeper networks and longer training, while on the other allows for feature extraction in architectures such as the mentioned U-net.

2.2.3.4 Forward and Backward propagation

Each neural network training step consists of two stages, forward and backward propagation. As fully connected neural networks are less conceptually complex due to their one-dimensionality, all the derivation in this section is performed on an example of a fully connected network. The same principles apply to convolutional neural networks or other deep learning architectures. The operations described until now, be it for fully connected layers or convolutional layers, are performed during the forward propagation through the neural network. Forward propagation of a fully trained neural network is known as inference – using the finalised neural network for its intended purpose, to predict an output for a given set of input values. Importantly for training, rather than just calculating the final value of the output of the neural network, during forward propagation all the intermediate values are calculated and stored.

Backward propagation is the other crucial element in the modern machine learning – it is essentially the automatic calculation of the derivatives of all the trainable parameters (in fully connected layers, weights and biases) in the network with respect to the given loss function – a measure of how far the output of the network is from the desired output. Before backpropagation became widespread, multilayer perceptrons existed, but the update of their trainable parameters required the manual calculation of the derivatives. This has understandably hampered the experimentation on the architectures, as any change required recalculation of derivatives for every trainable parameter. Forward and backpropagation are linked, as to calculate the partial derivatives for each of the parameters it is necessary to know their values.

In forward propagation, consider the simple fully connected neural network displayed in Figure 37. This neural network consists of two layers, as the input layer does not involve any calculations.

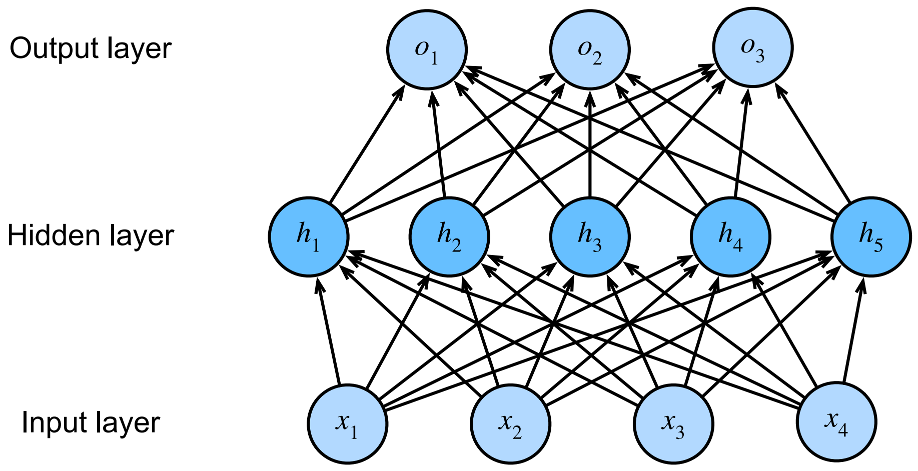


Figure 37. Fully-connected neural network with one hidden layer. Reproduced from [65].

Consider now the batch of data that the layer can calculate at a single time. This is known as the minibatch and is a basic unit of data that forward and backward propagates through the neural network. Referring to the equation of a neuron in a fully connected layer $y = w \circ x + b$, the minibatch is known as $\boldsymbol{X}$, the 2D matrix input of the shape (*number of examples, number of inputs*). Further, let's consider the value of the units in the hidden layer denoted as $\boldsymbol{H}$. This has the shape of (n, h), where n is the invariant number of the examples in the minibatch, and h is the number of units in the hidden layer. Finally, we consider the output layer units $\boldsymbol{O}$, of the shape (n, q). As both the hidden and output layers are fully connected, we consider the weights in the hidden and output layers, which are also 2D matrices, denoted as $\boldsymbol{W}^{(\mathbf{1})}$ of shape (d, h) and $\boldsymbol{W}^{(\mathbf{2})}$ of shape (h, q) as well as biases $\boldsymbol{b}^{(1)}$ of shape (1, h) and $\boldsymbol{b}^{(\mathbf{2})}$ of shape (1, q). As is clear from these shapes, the shape of the weights and biases of the layers is invariant to the size of the minibatch, n. The effect of the training on minibatches is to allow the trainable parameters of the neural network to update in a stochastic manner. $\boldsymbol{X}$ is given as the input to the neural network, and based on that we calculate the values of neurons in the hidden layer as:

$$\boldsymbol{H} = \alpha\left(\boldsymbol{X}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)}\right) \tag{36}$$

$$\boldsymbol{O} = \alpha\left(\boldsymbol{H}\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(\mathbf{2})}\right) \tag{37}$$

α is the activation function introducing nonlinearity. In the case of a more complex fully connected neural network, further hidden layers $\boldsymbol{H}^{(\mathbf{2})}$, $\boldsymbol{H}^{(\mathbf{3})}$etc. can be used to improve the predictive power of the network. During forward propagation all the values of the neurons in the hidden layers are stored to be used in backward propagation.

Backward propagation is effectively a method of automatic differentiation. A neural network outputs a value that is to be as close to the true value as possible. As a result, a derivative of the weights with respect to the function of the distance between the output value and the actual value provides the direction in which the weights have to change for the output to get closer to the desired value. This function used for calculating the distance is known as the loss function. In the simplest form it is the difference between the predicted output and the true value i.e. $Loss = \left|output_{predicted} - output_{true}\right|$.

The loss functions are usually further modified by regularisation terms. Regularisation terms introduce auxiliary objectives to the training process. Commonly they are used to ensure the weight values are as close to each other as possible. This has the advantage of utilising the predictive power of all neurons rather than being overly reliant on a limited number of nodes (effectively limiting the size of the neural network while keeping the computational complexity). This regularisation function is given by the equation:

$$s = \frac{\lambda}{2}\left(\left|\left|\boldsymbol{W}^{(1)}\right|\right|_F^2 + \left|\left|\boldsymbol{W}^{(2)}\right|\right|_F^2\right), \tag{38}$$

Where λ is a scalar known as regularisation rate, controlling how impactful the regularisation term is compared to the loss function.

The sum of the loss function *L* and the regularisation term *s* is known as the objective function J.

Consider the neural network in Figure 37. It has two sets of weights $\boldsymbol{W}^{(1)}$and $\boldsymbol{W}^{(2)}$. For simplicity let us not consider biases, as the process for the biases is analogous. The goal of the backpropagation process is therefore to calculate the derivatives of those sets of weights with regard to the objective function $\frac{\partial \boldsymbol{W}^{(1)}}{\partial J}$ and $\frac{\partial \boldsymbol{W}^{(2)}}{\partial J}$. To calculate the derivatives, neural networks use the chain rule, $\frac{\partial Z}{\partial X} = \frac{\partial Z}{\partial Y}\frac{\partial Y}{\partial X}$. The first step to calculating the derivatives for all the nodes is therefore calculating the derivatives of *J* with regard to *L* and *s*.

$$\frac{\partial J}{\partial L} = 1 \tag{39}$$

$$\frac{\partial J}{\partial s} = 1 \tag{40}$$

Next, the partial derivatives of the objective function with respect to the output layer are calculated – this results in a vector of derivatives corresponding to outer layer neurons:

$$\frac{\partial J}{\partial \boldsymbol{o}} = \frac{\partial J}{\partial L}\frac{\partial L}{\partial \boldsymbol{o}} = \frac{\partial L}{\partial \boldsymbol{o}} \tag{41}$$

Next, calculate the partial derivatives of the regularisation term with respect to both sets of weights, which we can do directly:

$$\frac{\partial s}{\partial \boldsymbol{W}^{(1)}} = \lambda \boldsymbol{W}^{(1)} \tag{42}$$

$$\frac{\partial s}{\partial \boldsymbol{W}^{(2)}} = \lambda \boldsymbol{W}^{(2)} \tag{43}$$

Next, we use the chain rule to calculate the gradients of the output with respect to the second set of weights – linking the hidden layer with the output layer:

$$\frac{\partial J}{\partial \boldsymbol{W}^{(2)}} = \frac{\partial J}{\partial \boldsymbol{o}}\frac{\partial \boldsymbol{o}}{\partial \boldsymbol{W}^{(2)}} + \frac{\partial J}{\partial s}\frac{\partial s}{\partial \boldsymbol{W}^{(2)}} = \frac{\partial J}{\partial \boldsymbol{o}}\boldsymbol{h}^{T} + \lambda \boldsymbol{W}^{(2)} \tag{44}$$

This is one of the desired values. Now we can backpropagate further. The gradient of the objective function with respect to the hidden layer output is:

$$\frac{\partial J}{\partial \boldsymbol{h}} = \frac{\partial J}{\partial \boldsymbol{o}}\frac{\partial \boldsymbol{o}}{\partial \boldsymbol{h}} = \boldsymbol{W}^{(2)^{\mathrm{T}}}\frac{\partial J}{\partial \boldsymbol{o}} \tag{45}$$

The next intermediate value is known as *z*, which is the input to the activation function:

$$h = \alpha(z) \tag{46}$$

The derivatives of the common activation functions are precalculated and used directly. As the activation functions are used elementwise, the derivative needs to be applied the same way:

$$\frac{\partial J}{\partial \boldsymbol{z}} = \frac{\partial J}{\partial \boldsymbol{h}}\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} = \frac{\partial J}{\partial \boldsymbol{h}} \odot \alpha'(\boldsymbol{z}) \tag{47}$$

Where $\odot$ denotes element-wise multiplication. Finally, this derivative can be used to calculate the desired:

$$\frac{\partial J}{\partial \boldsymbol{W}^{(1)}} = \frac{\partial J}{\partial \boldsymbol{z}}\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{W}^{(1)}} + \frac{\partial J}{\partial s}\frac{\partial s}{\partial \boldsymbol{W}^{(1)}} = \frac{\partial J}{\partial \boldsymbol{z}}\boldsymbol{x}^{T} + \lambda \boldsymbol{W}^{(1)} \tag{48}$$

In this way, through the forward and backward propagation all the intermediate values are first calculated and later used to calculate the gradients. In this section we have derived the gradients for the update of the weights for all the units in the neural network. The same process can be used for calculating the updates to the biases. This process is done simultaneously for all the examples in the minibatch. As all the intermediate values need to be stored throughout the forward and backward propagation, the memory requirements for the training are quite high and significantly higher than for inference. To limit those requirements, the minibatch size can be reduced.

2.2.3.5 Loss functions

The previous section briefly discussed how the trainable parameters can be updated in the neural network. The objective function itself is comprised of two constituents – the loss function and the regularisation term. The regularisation term has been discussed in some depth in the previous section, but generically, this term is used to improve the training behaviour of the network. The second component, the loss function, is more directly linked to the specific problem the neural network is designed to solve. This section introduces the background theory of loss functions, as well as exemplifies some of the more commonly used.
In the backward propagation process, it would be ideal to calculate the derivatives of all the weights directly based on how distant the predictions are from the desired values. Let us consider a thought experiment in which two students are asked to solve a quadratic equation. There exists a well-defined process for solving quadratic equations, calculating the determinant of the equation and later the solutions. One of the students does not attempt to answer the equation, but just guesses the answer. The other is able to follow the procedure correctly until the solution calculation stage, at which point they make a mistake, and their results are off by a factor of 2. If the marking scheme provides only options of 0 or 1 point, the feedback given to both students is exactly the same, even though their directions of study should be quite different. Conversely, if the random-guessing student gets the answer right, their approach would be validated just as much as the correct way of solving the problem. A better designed marking scheme would provide the students with the information at which stage a mistake was made, which information could be used to direct the learning and test all the abilities required to perform the task. Unlike the quadratic equation example, the questions asked of neural networks rarely have a well-defined algorithmic solution – in such cases, conventional programming is a preferred approach. Due to that problem, it is not possible to check the neural network solution step by step. This puts the onus on the design of the training examples, as they need to cover the possibilities of mistakes made at each stage.
A loss function deals with the other side of the problem – it assigns the specific numeric value to how bad the output of the neural network is. A well-designed loss function results in a

larger value when the neural network needs to adjust its output more, and a smaller one when it is close to correct. The problem that the designer of the ML algorithm needs to face is what is the best way to quantify that error. For most typical tasks, the loss function selection is narrow and the specific loss functions are implemented in the machine learning frameworks, such as TensorFlow. For complex or novel applications, though, they also provide a way to define a bespoke loss function. As an example, the simplest possible loss function is the absolute difference between the outputs ($\hat{y}$) and the true values (y). This is known as the Mean Absolute Error loss. Mean absolute loss works well if the output is designed to adjust the volume of an alarm tone based on the acoustics of the room. A bespoke loss function based on both the volume and the frequency content could be used, on the other hand, when adjusting the volume of music. The next part of this section introduces the most commonly used loss functions and their use cases.

2.2.3.5.1 Mean Squared Error

Mean Squared Error Loss is a minor refinement of the Mean Absolute Error. It is calculated as:

$$L = \left(y_{true} - y_{pred}\right)^2 \tag{49}$$

The advantage of the Mean Squared Error loss function over the Mean Absolute Error is encapsulated in the difference between the linear and the quadratic function. The quadratic loss rises in the value faster than the linear. This leads to quicker learning when the output of the neural network is far from the desired output. It is typically advantageous; therefore, Mean Squared Error loss has mostly supplanted the Mean Absolute Error loss. Mean Squared Error is used as a loss function in the applications where the output of the network is continuous and not bound, as Mean Squared Error can be calculated for any set of two values. It is therefore the most commonly used loss function for regression tasks.

2.2.3.5.2 Cross Entropy – Binary and Categorical

Cross Entropy is a statistical measure of the dissimilarity between two probability distributions for a given random variable. In information theory the information of an event, a measure of how surprising an event is, is calculated as:

$$h(x) = -\log\left(P(x)\right), \tag{50}$$

where h is the information and $P(x)$is the probability of the event x. The concept stemming from information is the entropy – a measure of the total amount of surprise in a given distribution. If the probability distribution is very skewed, it contains less surprise, as most events are likely. If the distribution is balanced, all the events are equally likely, thus making every single one relatively surprising. The entropy $H(X)$ is:

$$H(X) = -\sum_{x} P(x) \log\big(P(x)\big) \tag{51}$$

where X is a discrete random variable, and the likelihood of event x is distributed according to the probability distribution P. Entropy is thus the average level of information across the possible outcomes. Finally, the crossentropy between two distributions P and Q is the calculation of the number of bits necessary to represent the event using distribution Q instead of P. It is calculated as:

$$H(P,Q) = -\sum_{x} P(x) \log\big(Q(x)\big) \tag{52}$$

When used as the loss function, P is the approximation of the target probability distribution (it is 1 for the correct class and 0 for all other classes) while Q is the prediction of the machine learning algorithm. For a two-class problem, the formula can be unrolled as:

$$H(P,Q) = -P(class0) \log\big(Q(class0) + P(class1)\, log\big(Q(class1)\big)\big) \tag{53}$$

When modelling the continuous probability distributions, it is typical to use the natural logarithm instead of binary, and this is how crossentropy is implemented in ML frameworks. This scales the output by a constant value so makes no difference when minimising the loss function.

Cross entropy is also known as logistic loss (conveniently abbreviated to log loss) and negative logarithmic loss. The logistic loss formula is derived from a different starting point, simply as the logarithm of the error for each class, but the final formula aligns exactly. The value derived is known as the binary crossentropy, given that two probability distributions exist. It can be generalised to a multi-class problem when it becomes known as the categorical crossentropy. This version is typically used in combination with the softmax activation function. Cross entropy is by far the most commonly used loss function for classification tasks.

2.2.3.5.3 Focal Cross Entropy (Focal Loss)

Focal loss is a more advanced loss function, introduced here as a method of describing how loss functions can be used to address some of the pertinent issues in machine learning. While crossentropy loss performs very well in a wide range of classification scenarios, there are two significant failure modes. The first one is the function of the stochasticity of the crossentropy – to approximate the probability distribution, the crossentropy is calculated as a mean across a minibatch of training examples. If one of the classes is overrepresented in the training dataset, the loss function incentivises getting the dominant class right at the expense of the underrepresented class. In the extreme, this can lead to the algorithm reverting to categorising every example as the member of the dominant class. This issue can be addressed easily by multiplying the cross entropies on the per-example basis by the weighting factor dependent on the balance of the classes. Such a modified loss is known as balanced crossentropy. The second issue with crossentropy is the failure to focus on hard examples (i.e., the ones with probabilities close to 50/50) rather than on the easy ones. In many machines learning applications, it is beneficial to focus on the difficult examples. This problem is addressed by the variation of the crossentropy known as the focal loss [66]. The focal loss is given as:

$$L = -\sum_{i=1}^{n} (i - p_i)^{\gamma} \ln(p_i) \tag{54}$$

For misclassified examples the value of p_i is small, making the focal loss behaviour similar to that of standard crossentropy. When the confidence of the algorithm increases, the weighting factor γ pulls the value of the loss function down, ensuring that more attention is paid to the difficult examples. The value of the weighting factor is normally determined experimentally, in the original paper the value performing best for the authors is 2 [66]. Coming back to the issues with the crossentropy, it would seem that focal loss would solve both at the same time, as quite naturally the examples from the underrepresented class are more difficult to classify, but practically the most commonly used implementation of the focal loss takes the form of:

$$L = -\sum_{i=1}^{n} \alpha_I (i - p_i)^{\gamma} \ln(p_i) \tag{55}$$

where α_I is introduced as an additional weighting factor used to place more reward on the correct classification of the underrepresented class.

2.2.3.6 Optimisers

Once the loss function and the gradients of all the trainable parameters have been calculated, the next step is to perform the update of the values of the trainable parameters. The amount by which the values are updated is based on two constituents – the gradient itself and the optimiser. The optimiser is a function calculating the update based on the input gradient. The simplest possible optimiser is known as the gradient descent – it simply updates the parameters by subtracting the gradient multiplied by a selected constant.

$$w_{new} = w - a(grad_w) \tag{56}$$

As discussed previously, the training of the neural network is done in batches of n examples. The version of gradient descent adapted to multiple inputs is known as stochastic gradient descent (SGD) and calculated (for n examples in the batch) as:

$$w_{new} = \ w - a\frac{\sum_{i=1}^{n} grad_w}{n} \tag{57}$$

Effectively updating the weights using an average of the gradients across the minibatch. This simple approach was commonly used in the early days of machine learning, but with the evolution of the field an understanding has arisen that the loss landscape can be complex. In such a landscape keeping the update coefficient (learning rate) constant makes it prone to two issues – too small would be unable to escape the local minima, while too large would risk overshooting the global minimum. While these issues were known, it is only when complex problems became the object of research that it became obvious that a constant learning rate would typically be too small at one point in the training process (when trying to find a rough solution) and too large when fine-tuning the answers. It became abundantly clear that it is not simply a problem of selecting the correct value for the learning rate, but that the optimizer needs redesigning to enable the adjustment of the learning rate over the course of the training. This section introduces the early improvements on the SGD optimiser to provide the overview of the adaptation algorithms and Adam optimiser [67]– the choice of most modern ML algorithms including the ones developed in this work.

2.2.3.6.1 Stochastic Gradient Descent with Momentum

The main issue faced by the stochastic gradient descent is understood easily when considering the gradient descent on a single example. Consider the one-dimensional loss landscape in Figure 38, and take a starting point of x=4.5.

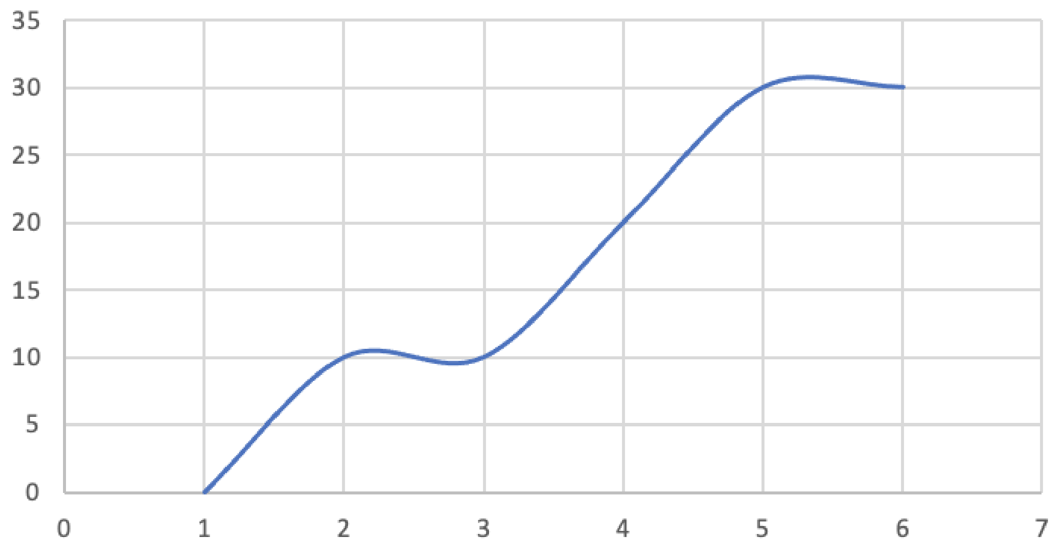


Figure 38. Example loss landscape with a local minimum around x = 2.8.

Then the weights are updated so the loss function is lower, which results in the next output being at x=2.8 (for example). The gradient of the loss function at that point is zero; this is a local minimum. As a result, the update of the gradient descent would be calculated as 0 and the training would be stuck. One of the methods of addressing this shortcoming is the introduction of momentum, first described in the context of neural network optimisation in 1999 by Ning Qian [68]. Momentum assumes that the updates of the network tend toward the global minimum, and this uses the previous updates as an additional input to the current update. The following derivation assumes V is the update after applying the momentum, while S is the update without the momentum:

$$V_t = \beta V_{t-1} + (1-\beta)S_t \tag{58}$$

$$V_{t-1} = \beta V_{t-2} + (1-\beta)S_{t-1} \tag{59}$$

$$V_{t-2} = \beta V_{t-3} + (1-\beta)S_{t-2} \tag{60}$$

Which can be combined to calculate V_t as:

$$V_t = \beta^2(1-\beta)S_{t-2} + \beta(1-\beta)S_{t-1} + (1-\beta)S_t + \beta^3 V_{t-3} \tag{61}$$

β is the momentum term, describing what is the impact of the current gradient in comparison with the previous updates; a value of zero means that $V_t = S_t$ and hence there is no momentum, and increasing values mean greater momentum. It is less than one, so as is clear from the derivation, the impact of the past on the following update deteriorates by the factor of β with each update.

Using stochastic gradient descent with momentum, the update to the weights is calculated as:

$$V_t = \beta V_{t-1} + (1-\beta)\nabla_w J(W, X, y) \tag{62}$$

$$W_{n+1} = W_n - V_t \tag{63}$$

where ∇_w is gradient with regard to weight, and J is the objective function. Compared to SGD, the momentum term adds the additional calculation to what was a constant update rate, making the training process more robust, especially when optimising near so-called ravines – regions on the loss landscape, where the gradient in one direction is significantly higher than in the other. In such cases, momentum allows the optimiser to preserve the ability to move in the correct direction over the update iterations. The difference is shown in Figure 39.

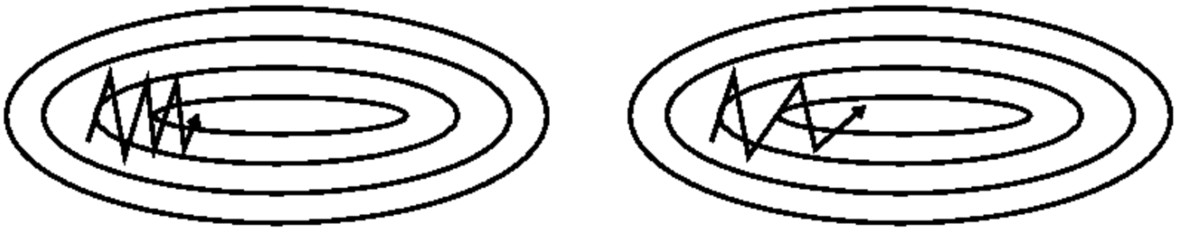


Figure 39. Comparison of model update without momentum (left) and with momentum (right) each straight stretch of the line corresponds to a single update of the model. The concentric lines represent the loss landscape with the lowest loss at the centre. Reproduced from [69].

2.2.3.6.2 Adam Optimiser

Adam is a modern optimiser first introduced at by Kingma and Ba [70]. It takes advantage of the first order moment (gradient) and second order moment (squared gradient). This effectively allows the optimiser to calculate an individualised learning rate for each of the trainable parameters. Adam has immediately gathered a large following, as proven by nearly 150,000 citations of the ArXiv preprint of the paper. In 2016 Sebastian Ruder has conducted a review of optimization algorithms [71], which has established Adam as the first-choice algorithm in machine learning applications. The Adam algorithm update procedure, in the words of its creators is:

1. Compute the gradient and its element-wise square.
2. Update the exponential moving average of the 1st order moment and the 2nd order moment.
3. Compute an unbiased average of the 1st order moment and 2nd order moment.

4. Compute weight update: 1st order moment unbiased average divided by the square root of 2nd order moment unbiased average (and scale by learning rate).
5. Apply update to weights.

As is clear from the procedure, Adam has more parameters than SGD (just learning rate) or SGD with momentum (learning rate and momentum term β). Adam requires a learning rate, a decay term for the first order gradient β_1, decay term for the second order gradient β_2, and ϵ, a term used in step 4 of the update to avoid the division by 0. Although the number of parameters in Adam optimiser is large, potentially proving problematic at the stage of the hyperparameter tuning, the values for the hyperparameters suggested by the original paper are remarkably well suited to most machine learning problems. They are, as follows: learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08. Of the parameters of the optimiser, only the learning rate is commonly adjusted, either to a different constant, or to an exponentially decaying value, getting smaller as the training progresses to better optimise near the global minimum.

2.2.4 Modern Topics in Machine Learning Research

Machine Learning and Artificial Intelligence more broadly are some of the most dynamic areas in research. This section aims to introduce the recent impactful developments both in the context of the techniques used and the concerns regarding the explainability and fairness of ML models. This project has utilised Generative Adversarial Networks, but the majority of this section is provided for completeness and to inform future research.

2.2.4.1 Generative AI

Generative Artificial Intelligence is a subset of machine learning models that aim to produce outputs closely resembling human creations. They do that by learning the patterns and distributions in the training dataset and use this knowledge to generate new instances of data adhering to the patterns. The concept of Generative AI is very broad and many approaches are utilised, depending on the domain-specific requirements, but some of the most commonly utilised types of models are Generative Adversarial Networks (GANs) [72], Variational Autoencoders (VAEs) [73] and autoregressive models.

2.2.4.1.1 Generative Adversarial Networks

GANs, in principle, utilise a framework where two neural networks, the generator and the discriminator, are in competition to either deceive the other one or not to get deceived. The generator aims to produce data that closely mimics real examples, while the discriminator is presented with a data point that may either be generated by the generator or an actual real example and is then tasked with telling one from the other. This eponymous adversarial process drives both networks to improve their performance, forcing the generator to create ever more realistic data to deceive the improving discriminator. The training process of a GAN is characterized by a balance between the generator and discriminator. If one of the networks improved significantly faster than the other, the training process would end quickly, without much improvement to the generator. Initially, the generator produces random and low-quality samples that the discriminator can easily distinguish from real data. As training progresses, the generator gradually improves its performance and generates more realistic samples, making it increasingly difficult for the discriminator to differentiate between real and generated data. The training process involves iterative updates of the generator and discriminator networks. The generator's loss function is based on the discriminator's misclassification of the generated data as real, while the discriminator's loss function is based on its ability to correctly classify real and generated data. As the weight updates for the networks are separate, an opportunity exists for the designer to fine tune the training process so that the balance is retained.

GANs have been used mostly in the visual domain. They can be used to generate novel life-like images of faces or scenery. They are also commonly used in neural style transfer [74], where the artistic style of one image can be transformed to another and image-to-image translation, for example turning a sketch into a realistic graphic. Finally, GANs are used in the medical domain, where they are employed to generate novel images that can be used to train image recognition algorithms, thus helping to alleviate the data scarcity problem.

The drawbacks of GANs are mostly centred around their vulnerability to the data distribution and the selection of the hyperparameters. It is common that either a discriminator or a generator network outperforms the other by a margin significant enough that a training process stops. Secondly, GANs have an issue with mode dropping and mode collapse. Mode dropping, exemplified by a generator of dog images never producing photos of Poodles, leads to incomplete coverage of the target distribution, while mode collapse – the same generator producing only Labradors, results in the generator producing limited and repetitive samples. Addressing these issues is crucial for reliable and diverse data generation.

2.2.4.1.2 Variational Autoencoders

Variational Autoencoders (VAEs) are based on two underlying technologies, autoencoders and variational inference. Autoencoders are neural networks consisting of an encoder and a decoder. The encoder maps the input data to a low dimensional latent space, which is essentially a compressed representation of the input. The decoder is the decompressor part – it uses the latent space representation to produce the original input. As the latent space contains less information than the input, autoencoders can be used for dimensionality reduction, data compression, or denoising. The second technology, variational inference is a method of approximating complex probability distributions by representing them as a set of continuous variables. In the context of VAEs it is assumed that the underlying probability distribution belongs to a library of standard distributions that can be described using continuous parameters. For a Gaussian distribution the parameters would be the mean and the variance. The architecture of a variational autoencoders is therefore comprised of three elements, each of which requires designing – the encoder, the decoder, and the latent space. Both encoder and decoder are typically deep neural networks. Depending on the type of the input data they can follow any architecture. The limitation is that, as the encoder is meant to compress the data, the output of the encoder should be lower-dimensional than the input. The output of the decoder, in turn, is compared to the input to the encoder, therefore it must have the same dimensionality. Classically, the design of the latent space requires domain knowledge, as choosing the correct initial distribution can significantly improve the trained VAE performance.

The variational autoencoder training process involves the optimisation against two parts of the loss function – the reconstruction loss, typically MSE or crossentropy, measuring the dissimilarity between the input and the output of the VAE and the regularisation term – KL Divergence. Kullback-Leibler (KL) divergence is a term that quantifies the dissimilarity between the inferred latent distribution and the prior distribution.

VAEs have, similarly to GANs, been used for novel data generation, but their main impact lies in its unique applications. VAEs have shown promise in anomaly detection. In that application, the network is trained on normal data samples and the latent space representation (mean and standard deviation) is used to determine whether a new data sample belongs to the normal data category. Secondly, VAEs are useful for data imputation and denoising tasks, where missing or corrupted data points need to be recovered. By reconstructing the missing or noisy data from the latent representation, VAEs can effectively impute missing values and remove noise from the data.

The main drawback of VAEs is the difficulty in the design of the latent space. Traditionally, it involved a standard Gaussian distribution with two parameters (mean and variance), which is easy to optimise. If the prior data distribution is far from Gaussian or multimodal, that approach would fail to provide a good representation. In recent years this approach has been improved upon by using a multimodal Gaussian as a standard as well as combining VAEs with other generative approaches to model the distribution.

2.2.4.1.3 Autoregressive Models

Autoregressive models, in opposition to the previously discussed architectures, utilise the concept of sequential generation i.e., the point generated is dependent on those generated previously. Autoregressive AI models the probability distribution of the next data point based on the preceding ones, the so-called conditional probability distribution. The order-dependence of the output of autoregressive models necessitates the utilisation of specialised architectures, designed to take advantage of that context, such as a Recurrent Neural Network (RNN) [75] or a Transformer [76]. RNNs are a class of neural networks designed to process sequential data by retaining an internal variable which is a representation of the past of the sequence. The common implementations are long short-term memory (LSTM) units or gated recurrent units (GRUs). In autoregressive models, the hidden state is updated at each time step based on the current input data and the previous hidden state. The final hidden state is used to compute the conditional probability distribution for generating the next data point. Transformers are a more recent architecture using self-attention mechanisms to weigh the importance of inputs based on their values and positions in the sequence. In autoregressive models, contrary to RNNs, transformers can process the entire input sequence in parallel, allowing them to take advantage of parallel computing facilities such as GPUs and speeding up the training process. During the training of an autoregressive model, it computes the conditional probability distribution for each data point based on the context of the preceding data points. This training process can be computationally demanding, as generating a single point requires rerunning the model for each point in the sequence. In order to speed it up, a process called teacher forcing is often used, providing the ground truth values (i.e., the sequences that exist in the real world). This drastically improves the speed of the training but may lead to the autoregressive model performing significantly worse when the ground truth is not available.

The most impactful application of autoregressive models is language modelling. Large Language Models (LLMs) are autoregressive models trained on large bodies of natural text. They can then generate realistic and context-specific text. This is nowadays used in AI-powered chats such as OpenAI ChatGPT or Google Bard. Furthermore, language models are used in machine translation and autocompletion. The rise of LLMs has been enabled by the existence of large databases of natural text, therefore the application of the same principle to other domain areas is dependent upon the existence of similar data databases or successful transfer learning.

Autoregressive models have an issue with the generation of long sequences of samples due to the initial algorithm's inability to parallelise the generation. The speeding up of both the training and the generation is currently one of the largest technical hurdles before the models. Furthermore, all generative AI models give raise to a range of ethical and security concerns. Currently, AI-generated text and images are difficult or impossible to differentiate from the human-generated ones, highlighting the need for robust fairness and security regulations.

2.2.4.2 Reinforcement Learning

Reinforcement Learning (RL) is a training paradigm different to conventional supervised and unsupervised learning. Initially inspired by behaviourist psychology, it takes advantage of natural human learning based on trial and error and the reward mechanisms. The central concepts in reinforcement learning are the agent, the environment, and the reward. The agent is the computational entity that has the ability to select a course of action based on the policy it has learned. The environment is the set of states that the agent can perceive and interact with. Finally, the reward is granted to the agent given the environment reaches a state determined by the designer. The training process involves the optimisation of policy function to maximise the cumulative reward at the end of training. Fundamentally, the agent should be encouraged to explore the environment, trying out unknown strategies, but the exploration direction should be biased towards exploiting the more effective strategies. The right balance of the reward landscape is known as the exploration-exploitation trade-off.

Reinforcement learning algorithms are used in two main areas. The first is games, or problems that can be posed as games. The reason is that games usually have a very well-defined ruleset and the interactions the player can perform as well as an inherent reward landscape in the score. Many games can be automated by allowing the RL algorithm to train semi-autonomously and allowing it to reach its full potential. The second category is generalised control problems. For this type of problem, RL is well suited because while the final goal is known (the setpoint), the path to reach it is unknown even to the designers, especially in highly dynamic conditions. The same applies to problems such as autonomous driving or energy grid balancing.

The main concern with reinforcement learning is ethical and safety-based, as the agent is ultimately rewarded for reaching the desired state, which can potentially lead to its incurring costs or damage that the designers would deem unacceptable. This can be alleviated by carefully designing the restrictions and rewards, but the trial-and error nature of RL algorithms means that the designer can never entirely exclude safety concerns.

2.2.4.3 Transfer Learning

It is common when applying machine learning to real-life problems that the training data is limited. As a result, methods have been developed to adjust the training so that the algorithm trained on one set of data can perform well on another. The common term for such methods is transfer learning. Weiss, Koshgoftaar and Wang, the authors of a recent review of transfer learning techniques give a real-life example of two people trying to learn piano [77]. One has no musical experience, while the other is an expert guitar player. It is reasonable that the musically experienced trainee would learn significantly easier, i.e., require fewer training examples. The goal of transfer learning is to apply this principle to ML algorithms. In practical terms, it is based on training the algorithm on abundant out-of-distribution data and following it with a shift to in-distribution data. The task is easier if the target distribution and the original training distributions are similar. Object recognition algorithms can be trained on general photo datasets, such as ImageNet, for example. The problem arises when distributions are disparate, i.e., trying to apply a cat-detecting algorithm to the detection of defect indications in guided wave testing.

The common process for the practical implementation of transfer learning can be summarised as follows:

1. Take a previously trained deep learning model.
2. Freeze the trainable parameters, so the knowledge learnt is not lost in the subsequent training.
3. Add new layers on top of the existing model, which are meant to learn the target dataset distribution.
4. Train the new model on the target dataset.
5. Unfreeze all of the layers, change the learning rate so it is very small.
6. Train the whole model on the target dataset – this is known as fine tuning.

Note that points 5 and 6 are not always implemented, and fine tuning needs to be applied very carefully, as it can lead to the model "forgetting" the baseline information gained in pre-training.

2.2.4.4 Explainability and Fairness

The final modern direction of research in machine learning are the topics on the intersection of the technical challenge and an ethical concern, explainability and fairness. The explainability of a machine learning algorithm refers to the user's ability to determine why the output of the model is what it is, while the fairness refers to the equity of the outcomes of the model regardless of the characteristics of an individual or a group.

2.2.4.4.1 Explainability

AI models, especially deep neural architecture suffer from being black boxes. While in theory it is possible to trace how the input data impacts the activations of neurons step by step, in practice modern ML architectures have the numbers of parameters making such an approach impractical. Simultaneously many domains, especially the safety critical ones, like medicine, require the access to the decision-making process. An explainable machine learning model builds trust in the users, and it also allows for a more transparent distribution of responsibility. Finally, the regulations on the usage of automated decision systems often require the ability to explain the decisions.

The explainability can be introduced into the machine learning model by a range of methods. First, the model can be designed as explainable. This is usually done by the selection of a human-readable architecture, such as decision trees. The downside of this approach is the fact that such architectures are typically less powerful than unexplainable deep learning architectures. The second approach involves the utilisation of so-called post-hoc explainability methods. Those include the training of surrogate models – models with explainable architectures trained to mimic the predictions of unexplainable models, and using techniques such as LIME (Local Interpretable Model-agnostic Explanations) [78] or SHAP (Shapley Additive exPlanations) [79]. Finally, the third approach is using the AI-human collaboration, where an AI algorithm makes a recommendation or guides the human decisionmaker, who has the ability to override and provide feedback to the algorithm.

2.2.4.4.2 Fairness

While the author does not expect AI fairness concerns to have any bearing on the research presented in this work, it should be considered by every AI practitioner, thus this brief discussion is included for the benefit of the reader.

Fairness in AI is generally the pursuit of equitable results for the user no matter their personal characteristics. The lack of fairness in AI has two main reasons. First, the developers of AI hail mostly from developed countries and affluent backgrounds, second the existing datasets are mostly gathered in so called WEIRD (Western, Educated, Industrialised, Rich, Democratic) countries, making any model trained on such datasets naturally inclined to follow cultural biases of the data source. The dangers of unfair AI range from the ethical, that one should not strive to amplify inequalities to tangible legal and economic risks, as making decisions biased against protected characteristics can lead to legal challenges, fines by the regulators or reputational damage and public backlash.

AI fairness is a complex issue to tackle, as it is less of a technical issue and more of a systemic problem. However, recently there have been efforts to remove those biases. First, the developers must pay attention to the datasets they are using and be aware of the potential bias issues in the resultant models. Secondly, if the datasets available are biased and their utilisation is necessary, techniques such as resampling and adversarial training [80] can be utilised to remove some of the bias from the datasets. Furthermore, there exists a significant research push towards the creation of algorithms aware of bias and designed to reduce it [81]. Finally, any ML algorithm that has been identified to possibly cause fairness concerns should be periodically audited and remedial action must be taken if it is discovered to exhibit bias.

Taken together, AI explainability and fairness are two of the most important pillars upon which ML community must build their algorithms if they are to be considered transparent and trustworthy. These values must be a part of ML development especially in fields such as NDT, where the safety concerns and the regulatory overhang demand the confidence in the algorithms and responsibility for their decisions.

3 Data Processing and Considerations

This chapter introduces the data used throughout the rest of the thesis. This is crucial to consider, as it is widely accepted that the quality and the processing of the data is a prerequisite for training a well-performing ML model. The chapter is split into two sections covering the sources of the data – the experimental data received from Guided Ultrasonics Ltd., and the simulated data generated using finite element modelling. The machine learning models under research use two input modalities, the raw A-scans and images processed similarly to Guided Ultrasonics' processing method. An A-scan is a recording of the amplitude of the ultrasonic signal received plotted against the time of reception. This chapter explores each input-output data pathway. Furthermore, it explores the problems with the data acquisition and processing procedures which need to be overcome for the further development of machine learning approaches to guided wave testing. Finally, the resultant dataset compositions are presented.

3.1 Size of Datasets

This section provides the reference for the size of the datasets, whose characteristics are introduced later in the chapter and is meant as a reference for understanding the ML development further in the thesis.

Table 1. Dataset sizes and compositions.

Dataset Name	Sample No	Comments
Simulated Pristine	6000	N/A
Simulated Defect	6000	N/A
Simulated Weld	6000	N/A
Experimental Image Defect	32	Augmented to 160
Experimental Image Pristine	150	Augmented to 750
Experimental A-Scan Defect	58	Augmented to 406
Experimental A-Scan Feature	634	293 welds, 164 supports, 58 defects, 41 bends, 36 flanges, 28 reverberations, 4 entrances into earth, 10 unknown.
Experimental A-Scan Pristine	1800	N/A

Table 1 presents the various datasets generated for the purposes of this project. The data presented makes clear the difference between the sizes of simulated and real datasets. The difference is especially pronounced when considering the defect datasets. That difference is the motivating force behind much of the training and machine learning process design described in this work. Most of the datasets are independent of each other, but the experimental A-scan defect dataset is a subset of experimental A-scan feature dataset. The differentiation between the two is introduced, as they are used in two independent ML approaches.

3.2 Data Characteristics

This section deals with the input side of the data used in this project. On the side of the real data, it introduces the format of the raw datasets, the numbers available and the algorithms used for extracting the useful parts of the file. On the simulated side it introduces the settings and parameters used for running the simulations. Finally, it deals with the first stage output – raw numerical data, which can be used as an input in further, specialised data processing.

3.2.1 Real Data

The real data has been provided by Guided Ultrasonics Ltd. Every file is available in two formats – the first is the standard export from WavePro software, coming as an XLSX spreadsheet containing some of the metadata and the inspection results, while the second is a full representation of the inspection file in a JSON file. It contains significantly more information pertaining to the direct outputs of transducers and the richest metadata available. Due to the significant difference between the two formats, they are treated separately.

3.2.1.1 Amount of Data

The most important quality when considering any data source as training data for machine learning is its abundance. This is especially pertinent in the context of non-destructive testing, as the defective real data is notoriously scarce. The data gathered consisted of files collected over 55 inspections performed between the years of 2005 and 2012 on pipe diameters ranging from 4 to 36 inches. The inspection range is varied between 2 and 60 metres and most inspections were performed in the two-directional mode, effectively resulting in two data samples for each inspection file. The inspections were performed by a qualified inspector who also flagged the features of interest and set the DAC curve. The frequencies used varied between the inspections, but all are in the 14-60 kHz range, with each inspection conducted at 5 different frequencies.

It is crucial that the data is proven representative of the general inspection problems. To that end the range of features present in the dataset was investigated, based on the reports prepared by the inspectors on site. The set was found to contain 292 welds, 164 supports, 58 defects, 41 bends, 36 flanges, 28 reverberations (false echoes) and 4 entrances into earth. It also contains 10 signal signatures that the inspector was unable to identify.

3.2.1.2 Processed Export

The first format utilised in this work is the processed export format. It is generated by using the export function in WavePro software. This work uses a 'superuser' version of the export, thanks to the collaboration from GUL. The standard export provides only the metadata and the graphs generated in the software. Depending on the settings in the software at the time of export, it is either traces of the envelopes of T(0, 1) and F(1, 2), the synthetically focused unrolled pipe display or both. Importantly, the standard-user export format does not provide the numeric data. The export used in this work contains significantly richer data in the format introduced further.

3.2.1.2.1 Anatomy

The processed export file is an XLSX spreadsheet. It is split into two sections – the metadata and information in the first rows followed by the numerical data corresponding to the data registered by the inspections.

The metadata section is split into three subsections. The first is the header containing the information about the inspection and the file: the type of export, version of the software used for exporting, and Row Offset – the size of the metadata section, allowing for the relatively easy access to the numerical data. The second contains the information about the inspection and the file: file name, notes added by the inspector, coded type of the ring used, location of the inspection, inspection ID, date and time of the inspection and time of the export. The third section of the metadata and the one extensively used for the research is the list of features marked by the inspector, containing the type of feature, location, length, size (in mV transducer output amplitude) and any comments the inspector has made. Two separate lists exist, one for the forward direction of the inspection and one for the backward direction.

The data section is split into two sections, the forward inspection, and the backward inspection. Each of the sections contains columns with the following data: Distance, T(0, 1) amplitude of the RF signal, T(0, 1) amplitude of the envelope, F(1, 2) amplitude of the RF signal, F(1, 2) amplitude of the envelope. In some of the files there are additional columns containing the horizontal and vertical polarisations of the F(1, 2) mode. After the data there are columns containing a set of operator-set curves: Noise level, Call level, Weld DAC, Flange DAC, and Decay. The latter corresponds to the change in the DAC level between the points. This information can be used to distinguish between uniform condition pipes and locally corroded or buried pipes. Following the full set of columns for the forward inspection an identical set of columns for backward inspection is present.

3.2.1.2.2 Export Routine

As the file is a mixture between metadata used to inform the extraction process and the data to be extracted, the extraction process is fairly involved. It is therefore split into two extraction scripts, one which extracts the raw numerical data for forward and backward directions separately and one which processes the parts of data to be used for ML. This separation allows for a more flexible approach (i.e., extracting only defect indications, only welds, all features, changing the length of the data sample etc.). This text aims to describe the process step by step.

Extraction Routine:

1. Load the spreadsheet, separating it into text and numeric data using built-in MATLAB functions.
2. Search the text variable for column headers corresponding to relevant data: T(0, 1) and F(1, 2) envelope and RF, DAC curve and distance column, relating the exported data to real world. Adjust the column numbers by -1, as the data import is narrower than text by one column.
3. As the export file is not standardised and sometimes the forward and backward direction are reversed, ensure the order is correct.
4. As DAC curve exists only for the length of the inspection, reject the data points for which there is no corresponding value of DAC curve.
5. Extract the remaining data into 8 variables T(0, 1)/F(1, 2), envelope/RF, forward/backward.

6. Normalise the data between -1 and 1.
7. Save the backward and forward data in separate raw numeric files, with the corresponding distance vector.

Point 6 in the extraction routine is controversial, as it removes the information stored in the DAC curve, i.e. the normalised relative strength of the signal in the trace. This is done to produce the data in line with what would be available to the automated system at inspection time. As the DAC curves are currently set by the operators and the ambition of the system is to operate independently, any information injected by human is rejected. This makes the problem more difficult, but ensures better generalisation, as DAC information can be re-injected at any stage.

Furthermore, this approach departs quite far from what a human inspector is doing – the ML algorithm is forced to look at sections of the trace in separation from each other, while a human operator would follow a strict protocol, taking into consideration the context, setting the DAC curves and using the relative indications of features to classify them. Following such approach and designing a multi-agent ML pipeline is a promising research direction, which should be explored as part of future work.

Export routine, based on the extracted file:

1. Load the metadata from the corresponding original file.
2. Find the cell ranges containing lists of features of interest – separately for forward and backward.
3. Filter the cell ranges for the relevant data: feature type, location, optionally severity (for defect indications).
4. Filter the cell ranges for the type(s) of features of interest.
5. Do the following for both forward and backward filtered feature lists:
    a. Save the name of feature and the file name as a unique identifier.
    b. Use the location of the feature to extract part of the data traces containing the feature.
    c. Save the extracted window under the label clearly identifying the feature.
    d. Repeat for all remaining features on the list.
    e. Save the ID - data trace pairs as a dataset for the ease of traceability.

3.2.1.3 Raw Export

The second format of the file export is a raw file containing all the available data about the exporting process, the inspection, and the data registered by the individual channels. The downside of this format is that there is no processing performed on the data prior to the export. As guided wave raw inspection data typically contains significant noise, much research has been conducted to transform the raw data into useful knowledge. Furthermore, as the data in this format is not intended for dissemination, the format has not been designed for the ease of use, adding to the complexity of the extraction process.

3.2.1.3.1 Anatomy

This form of export is a JSON file. As such, it is a hierarchical structure, which contains metadata at each level as well containers of lower-level data. Due to the richness of data available in the file, this section focuses on the sections of the file relevant to the export routine, affording a cursory overview of the rest of the file.

The first level of the file contains the information about the exporting time and procedure as well as Header and SampleSet containers. The former contains the inspection metadata while the latter contains the readings of each channel.

The header contains information on the inspection ranging from the time and the location to the owner of the asset under inspection and inspector notes to calibration results and DACs. From the perspective of this work, there are two important sections of the header. The first is the Amplitude Balance array variable, corresponding to the relative sensitivities of each channel. This value can be used to normalise the channel results to a common baseline. Additionally, the number of elements in this array can be used to find the number of channels in the ring used during the inspection. The second important section is the data relating to the DAC available in the DACSurface container. The DAC is set manually by the inspector, with the curve fitted to the points selected. As such, it does not necessarily follow any analytical curve. For the purposes of the export, though, it is approximated by an exponentially decay dependent on the distance from the position of the transducers and the frequency of the inspection. The formula used for the DAC approximation is:

$$DAC = ae^{bx+cf+dfx} \tag{64}$$

where a, b, c, and d are constants set in the file, f is the frequency of the inspection in Hz and x is the distance from the transducer in meters. Clearly for the curve to be decaying with distance, the $b + df$ component needs to be negative. Furthermore, attenuation increases with the frequency, making d negative. Typically, distance is the overwhelmingly most impactful variable when calculating the decay, therefore it is expected that b is negative. Due to the experimental nature of the exporting process, the sign is sometimes flipped, which needs to be remedied in processing to recreate the intended DAC.

SampleSet contains all the data registered by the transducers. Structurally it is an n-element cell array, where n is the number of times any transducer receives and writes data. The cell can further be split into two functional sections – the calibration traces and the inspection traces. Only the inspection traces are considered in this work. Each of the traces within SampleSet is coupled with its own metadata referring to the specific trace. Most importantly, it includes the size of the dead zone (number of points removed from the beginning of the trace, corresponding to the area so close to the transducers that the excitation interferes with the reception), the frequency at which the inspection took place and the codes for the transmitting and the receiving transducers. The latter information is especially crucial, as the traces are not recorded in order, thus requiring processing to transform them into standard FMC format. Apart from the information used in this work the metadata also contains the information about the excitation type, the balancing values on transmission and reception, filtering and sampling rate. Finally, it contains Data variable, holding the raw numeric values for the inspection.

3.2.1.3.2 Export Routine

The main issue when extracting the data from the raw export files is the sheer quantity present in each of the files. At maximum, a single file contains the inspection data collected at 8 channels in each of two rows and five frequencies (considering the reciprocity) coming to a total of $5 * (16^2 + 16) * \frac{1}{2} = 680$ traces, with the added calibration traces. Secondly, the format is not standardised, therefore the export routine must be robust to single vs. double directional inspection, variations in the number of channels and calibration traces or even the presence or lack of presence of a frequency information sample between inspections at different frequencies. As opposed to the processed export format, this export is done in a single step, simply saving forward and backward directions separately.

1. Load the JSON file.
2. Find the number of channels based on the number of amplitude balance values.

3. Check if the file corresponds to a two-directional inspection or a one directional inspection:

To determine this, compare the amplitude balance values number to highest channel number in the dataset. If the number is equal, the inspection is one sided, if it is half, it is two sided. That is caused by the channel naming convention, for one sided it is A*0*:A*n*, for two sided it is A*0:*A*n/2,* B*0:*B*n/2.*

4. If the inspection is two-sided, adjust the channel number to the channel number in one ring.
5. Check the number of unique frequency values in the SampleSet.
6. Calculate the number of data traces recorded for each centre frequency:

The data is recorded for each channel pair, plus the traces corresponding to a single channel acting both as a transmitter and as a receiver. As such, the number of data traces per frequency is $(n^2 + n)/2$ where *n* is the number of the channels.

7. Most of the files contain samples providing no inspection data but serving simply as frequency information. Find those samples and save their numbers. Revert to an alternate script compensating for the different data structure if these samples are not present. Divide the samples into separate frequencies and do the following for each frequency.
8. Create a time vector using the metadata.
9. Create a set of indexes within a single frequency test corresponding to the direction of the inspection under consideration.

If the data is single directional the indexes are simply 1:nTraces. The situation is more complex for the two-directional inspections, as it is necessary to separate the directions from each other. This is achieved using the algorithm gradually building the index vector based on the number of channels.

10. Use the indexes generated to separate the data traces generated on ring A and received on ring A, generated on ring B and received on ring B and generated on ring B and received on ring A.

The data export format results in a peculiar ordering scheme. For an exemplary 8-channel, single direction ring for each channel the order of traces is as follows:

A1 -> A1, A3, A5, A7, A2, A4, A6, A8

This is further complicated in the two-directional scenario, but the index generation algorithm separates these data sources. The general ordering scheme is: first all the odd-numbered channels, then all the even-numbered channels. Within each category, sort in ascending order, as the typical FMC-style data uses channels ordered sequentially.

11. Remap the traces to FMC convention.
12. Implement any filtering on the FMC traces, typically a bandpass filter with a passband between 10 kHz and 100 kHz.
13. Sum the matrices up over the transmitters, thus receiving a CSM-type data, which can be used in the CSM imaging process described in the previous chapter.
14. Save each of the matrices, A→A, B→B, and B→A separately including the DAC information, pipe schedule and identifier linking it to the original file. The metadata saved is necessary for the purposes of the imaging algorithm implementation.

3.2.2 Simulated Data

The real data supplied by GUL constitutes an invaluable trove of information, allowing for the testing of a machine learning algorithm. Depending on the choice of the data sample length, the number of pristine examples available to gather from the data is very large; the issue is, however, the availability of data containing features (only 634 of them in the dataset). The problem is even more acute if the algorithm is to focus on the specific type of a feature, such as a defect indication, of which there are 58. While the dataset of 634 features, made more valuable by their acquisition in a service environment, may seem large in the context of NDT, it pales in comparison to even most rudimentary datasets used for general-purpose machine learning. In 2013 Kaggle, a data science community, ran a competition in designing a machine learning algorithm that would tell cats from dogs [82]. The organisers provided the participants with a training dataset containing 25000 images, balanced between cats and dogs, i.e., 12500 of each species. The difference of two orders of magnitude between the number of features in the experimental dataset and the number of cats, or three orders of magnitude, if just defect indications are to be considered points to the need of acquiring significantly more training data. One method is the augmentation of the data by flipping, shifting, or cropping the training data, but there are limits to the method and the resulting algorithm must be carefully tested for overfitting. To use the dataset size in the order of magnitude of more common ML approaches it has been decided that finite element simulations would be used to augment the real dataset.

The further advantage of simulations is their perfect controllability, thus allowing for quickly generating a simple dataset that can be used to test a proof-of-concept machine learning algorithm.

3.2.2.1 Simulation Parameters and Process

The object to be simulated is a pipe – a hollow cylinder. This basic geometry is to be modified by features of interest, selected from the range present in the real data. Once the geometry exists, a torsional displacement wave is to be generated from a set of elements on the surface of the pipe at a single axial location and displacements are to be recorded at the same elements.

The real inspections have the rough range between 10 m (for extremely attenuative conditions) and 50 m (for a pipe in good conditions). The inspection is also typically performed in two directions. As is clear from the analysis of the raw export data, the inspection is performed by firing each channel in turn and recording the resultant echoes.

These conditions can be simulated faithfully, but they result in a simulation of a significant size, and require multiple simulations to mimic a single inspection, thus limiting the number of simulations that can be completed in each timeframe using limited computational resources. As a result, a set of simplifications is implemented:

1. Limiting the size of the simulated pipe: the computational complexity, and therefore the time of the execution of the simulation, is proportional to the size of the pipeline, as each element state needs to be calculated at each time step. Furthermore, the time of the simulation needs to be increased too as it would take longer for the probing wave to travel to the end of the inspection range and back. In the real inspection, the distance of propagation is directly linked to the attenuation of the wave and thus the signal to noise ratio. The simulations do not show the attenuative behaviour and the noise is injected artificially, thus the inclusion of the distance does not impact the accuracy of simulations. On the other hand, the simulation of a 5 m long pipe instead of a 50 m long one allows for around a hundredfold time saving based on the 10-fold increase in the number of elements and same increase in the number of time steps.

2. Simulating a unidirectional inspection instead of bidirectional. Similarly, to the previous point, it is more computationally efficient to simulate two half-length pipes than a full-length pipe. As the direction separation is implemented in the real data and it is known to perform well enough that reverberations are an issue only for the strongest features, running a unidirectional simulation is a simple way to achieve around a two-times time saving (assuming doubling the length of the pipe).
3. Using a plane probing wave instead of individual excitation of the channels. In the real-world inspection every channel is excited separately, allowing for the collection of full matrix data and potential complex processing. This is enabled by the relatively short time of a single inspection, defined by the wave travel time. Assuming a rough shear wave speed of 3200 m/s and a two-way propagation path of 100 m, the single channel excitation and reception takes about 0.03 seconds. For 16 channels the raw transmission-reception time would come to 0.48 s. Even accounting for the time needed for the reverberations to settle, the difference between a single channel and 16 is not a crucial consideration. Conversely, for simulated inspections every channel firing requires a full run of the simulation, thus limiting the run to a plane wave shortens the simulation eightfold. The downside is the loss of the potential to perform full-matrix techniques, such as total focusing method. However, as has been mentioned in the previous chapter, CSM is a preferred method of synthetic focusing method in guided wave testing.

Considering all the efficiency savings, the simulated pipe is a 5 m long pipe with a transducer ring located 1 m away from one of the ends. The end reflections are minimised by utilising absorbing boundaries with the length of 0.5 m on either side. The pipe wall thickness is 8 mm, and the pipe external diameter is 8 inches. The material of the pipe is steel, with Young's modulus of 210 GPa, shear modulus of 80 GPa and density of 8000 kg/m^3. Given those material properties the probing shear wave speed in the pipe is 3162.3 m/s, while the maximum wave speed, corresponding to the longitudinal wave, is 6055.3 m/s. To ensure full coverage of the simulated portion of pipe, the time of the simulated inspection is set to 0.0285 s, corresponding to the propagation distance of 45 m, providing more than 4-fold margin.

These physical considerations are implemented into the FEM solver using 8-noded brick elements with a single integration point. The dimension of the element (identical in all directions) is 2.67 mm. The number has been selected so the thickness of the pipe wall is exactly 3 elements, limiting the noise caused by rounding errors. This element size results in 240 elements around the circumference of the pipe. The total number of elements in a model is 8089920. The time step of the model is a direct consequence of the selected element size, as it must be smaller than the time taken for the fastest wave (bulk longitudinal wave) to propagate across the smallest element in the system. The Courant number selected for the set of simulations is 0.3, resulting in the time step of 0.132 μs.

This model undergoes a force excitation at the distance of 1 m from the end of the model. The shear circumferential force is excited at 24 nodes (10% of circumference), joined in the groups of 3 to mimic GUL's rings. The direction of the excitation is calculated based on the angular position of the node. The excitation directions are saved using the Pogo dofGroup functionality, allowing for easy reimplementation of the same direction on reception, resulting in the direct output of torsional displacement.

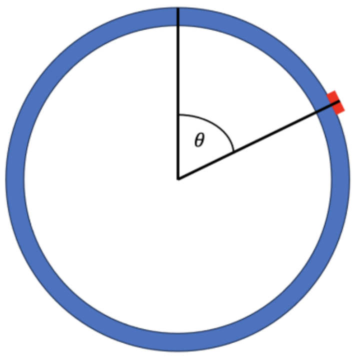


Figure 40. Location of the transducer on the surface of the pipe at the angular position of θ.

Pogo requires the excitation (or reception) directions to be expressed in the cartesian coordinates. As the intended direction is circumferential, the axial component is always zero. For an example transducer presented in Figure 40 the horizontal and vertical excitation directions are calculated as follows:

$$f(x) = -\sin(\theta)F, \;\; f(y) = \cos(\theta)F \tag{65}$$

where $f(x)$ is the horizontal displacement, $f(y)$ is the vertical displacement and F is the intended tangential force.

The excitation shape is a 5-cycled Hann-windowed wavepacket. Every simulation is run at 5 different frequencies of 17, 21, 25.5, 31 and 35 kHz. These are selected to mimic the frequency range and excitation type most commonly used in the real-life inspections.

Pristine pipe simulations can be used to enhance the negative dataset, but the more pertinent issue is the augmentation of the positive dataset. As such, it is necessary to add features into the simulation. The simulations use two types of features – an axisymmetric, large weld-type feature and a non-axisymmetric small defect-type feature. A weld-type feature is simulated using element stretching approach – locally changing the size of the elements to create the target geometry. The simulated weld has the parameters of location, axial extent, and height of the cap. Mathematically, it is simulated as a quadratic function with the maximum of the height of the cap at the axial location and roots at positive and negative half-extent. The graphical representation of the parameters of the simulated weld is presented in Figure 41. Both the axial extent and the height of the cap are randomised, the former in the range of 20-40 mm and the latter in the range of 1.6-4.8 mm. The resulting model is axially symmetric.

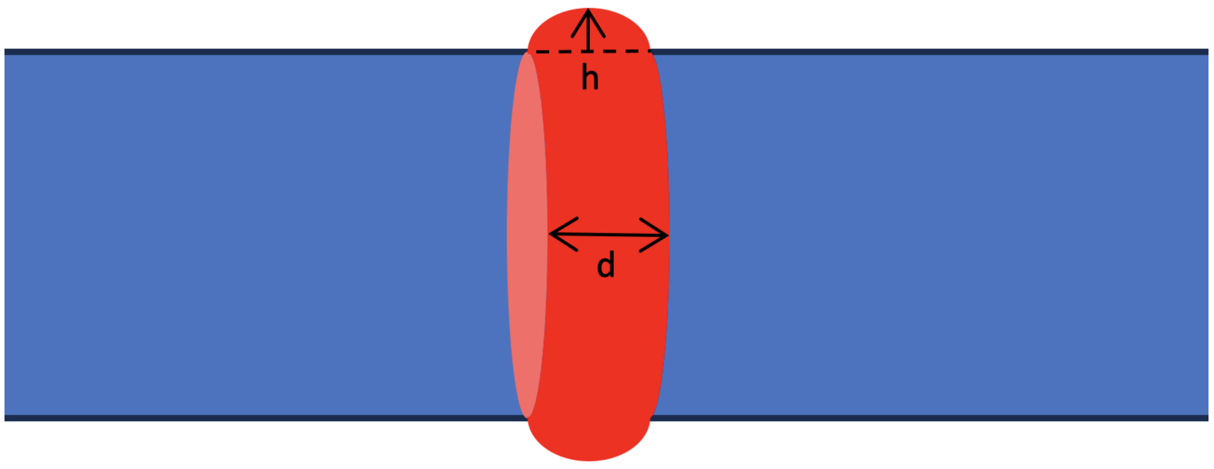


Figure 41. Geometry of a simulated weld (red) on a pipe (blue). The height of the cap is marked as h, and the axial span is marked as d.

The defect is simulated as a small through thickness circumferential crack. In the simulation it is achieved by disjoining adjacent elements, effectively creating a zero-length crack. While this type of a defect is not realistic, the size of the reflection is influenced solely by the change in cross-sectional area percentage. With the defect generated as a through-thickness crack this variable is easily controlled, as the change in cross sectional area percentage is the same as the span of the crack expressed as the percentage of the circumference. During the simulation defects of CSC between 3 and 18% were simulated. Real defects are typically corrosion-related; therefore, they have spans in all three dimensions. In the course of the project the simulated defects are simplified to zero-length change in CSC as the detection resolution of the wave is limited by the wavelength. For T(0, 1) wave in steel at the frequency of 31 kHz, the wavelength is around 10 cm. While the dimensions of the feature impact the reflected wave, the effect is to be very small in the context of this project, motivating the move to a simpler and easier to control model.

3.2.2.2 Anatomy

The output of the simulations is easily controlled and depending on the intended use case two versions are used. Generally, the output format is a POGO-HIST file, a structured format containing, at the first level, the data about the simulation: time step, length of simulation and the data trace. Furthermore, the user can include self-defined metadata which is defined in the input file and maintained in all outputs once the model has been run. In this work it is used to store information such as the locations and sizes of defects and welds, the length of the pipe, the location of the ring, length, and presence of absorbing boundaries as well as any information that could be useful at the stage of data processing or analysis. This richness of the data contained in POGO-HIST provides full traceability of simulations. Both the simulation data and the metadata are used in the output data format regardless of the simulation output data.

The first output form, exporting essentially raw data uses the dofGroup described in the previous section to read the tangential displacement of the nodes corresponding to each of the transducers individually. This output format corresponds to the unprocessed output of piezoelectric transducers used in torsional shear configuration. The output format is therefore a matrix of (nTransducers, nSamples) size.

The second output format implements the basic data processing within Pogo. dofGroups functionality can be used to directly calculate the mode amplitudes. This is since mode amplitudes are simply transducer readings weighted by the mode shape. For T(0, 1) the mode shape is a constant, thus the mode amplitude is simply a summation of all transducer readings. For flexural modes, the weights correspond to the mode shape, which is a sinusoid with the number of periods around the circumference equal to mode circumferential order (F(1, 2) is one, F(2, 2), two etc.). Due to the Nyquist’s criterion, the maximum order of the mode that can be extracted is half of the number of transducers. In practical terms, if higher order flexural modes are required, this work uses the spatial Fourier transform method to apply mode extraction in post-processing. Considering that, this output format has the shape of (nModes, nSamples), where nModes is typically 1 or 2.

3.2.3 Common Output Formats

As a final note on data characteristics, to homogenise the processing between the real and simulated data, it is necessary to introduce the common data formats. It is possible to draw the parallels at this point. The Processed Export GUL format provides the T(0, 1) and F(1, 2), which corresponds directly to the simulation output, where dofGroups are used to extract the first two modes. Similarly, the Raw JSON export output format is effectively a CSM output – a record of the tangential displacement of each transducer. This format is the same as the first, unprocessed version of the simulations.

The two data formats can be separated at this point, as they are used for separate approaches – the T(0, 1) and F(1, 2) can be used directly as inputs to the machine learning algorithm. The higher order modes, which can be extracted from CSM data can be used for imaging the pipe. For each of the approaches, the processing is vastly different, thus much of the subsequent work is split between the two.

3.3 Data Processing

The raw data extracted from either the simulations or the inspection records requires processing. The exact type and amount of processing required is dependent on the source and the intended use of the data. Clearly, the processed inspection data requires less input than the raw data. Furthermore, the simulated and real data require different processing due to the differing desired results; generally speaking, the inspection data is noisy and difficult to interpret. As a result, it is standard to filter it, so the features are easier for the inspector to identify. Conversely, if a simulation is well set up, there are no sources of noise present. As a result, any non-zero signal is certain to come from a feature of interest. A detection task on such data is trivial, thus making it an extremely low-value training set for ML. Hence, the goal of processing the simulated data is to make it less clean, ideally injecting the noise to the level of a processed real data.

3.3.1 Lower-Order Modes Data

The lower order modes data is significantly simpler to process than the full data. That is mostly caused by the pre-processing of the inspection export which uses a finely-honed GUL process. As such, the main processing methods are matching the amplitudes and injecting the noise into the simulations.

3.3.1.1 Attenuation Compensation

The primary difference between the real and simulated datasets is the lack of attenuation in the simulations. Fortunately, this issue is easily remedied by the application of DACs to the real data. As DAC is a curve that tracks the received amplitude of a reference reflector as a function of distance, it is enough to pointwise divide the mode amplitudes by DAC to correct the signals.

$$T(0,1)_{norm}(x) = \frac{T(0{,}1)(x)}{DAC(x)}, \qquad F(1,2)_{norm}(x) = \frac{F(1,2)(x)}{DAC(x)} \tag{66}$$

3.3.1.2 Noise Injection

The second main difference between the real and the simulated data is the noiselessness of the simulations. There are many strategies for injecting the noise into the data either at the stage of the simulation or in postprocessing. Crucially, it is important for the injected noise to have the same characteristics as the noise in the real data, meaning a mixture of coherent and incoherent noise. Coherent noise is the noise inherent in the inspection process. Its amplitude is connected to the signal amplitude and falls with attenuation. It is typically caused by the physical conditions of the pipe, and the lack of balance in the transducers which could, for example, cause additional modes to be generated and measured in the signals. Importantly, it is impossible to remove it by averaging. Incoherent noise is typically caused by the electronic interference. It is generally easy to add, as it can simply be considered a band-limited white noise. The coherent noise though requires a more involved approach.

This work uses the procedure described by Mariani, Heinlein and Cawley [83]. The approach is based on unbalancing of the transducers. The amplitude of each transducer is scaled by a value between 0.5 and 1.5. The same values are used on reception. This approach adds coherent noise, mimicking the issue in the inspection record.

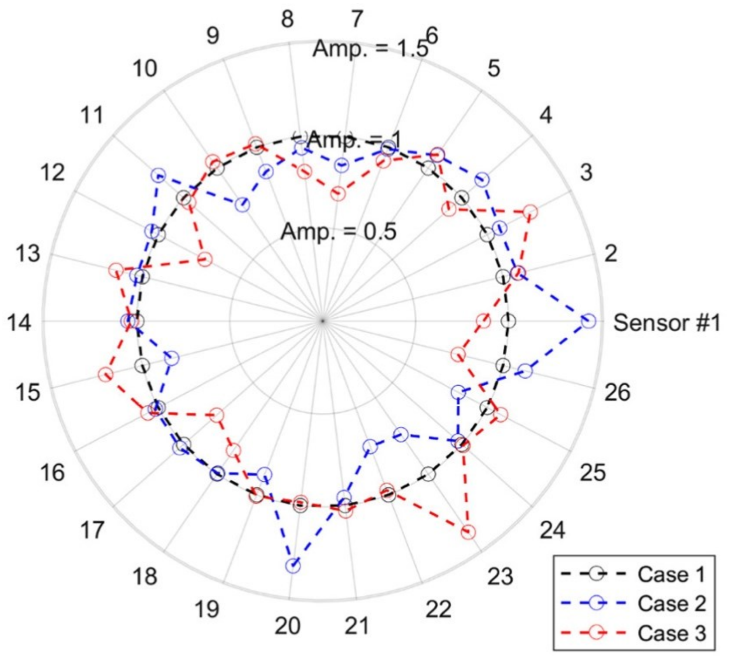


Figure 42. Balanced transducer amplitude multiplication factors (Case 1) and two possible unbalancing factors which can result from the method used in this work (Cases 2 and 3). Reproduced from [83].

3.3.2 Imaging

The data processing for imaging is primarily focused on the experimental data and designed to remove the noise from the signal. Compared to lower-order data, synthetically focused data is significantly more noise-prone, due to both the higher complexity of processing involved, calling for many assumptions (such as constant wave speed used for backpropagation, which in fact can change with factors such as temperature) and the fact that the experimental data used for imaging is only processed with a bandpass filter. Consequently, efforts are made to match the experimental data points' characteristics to those of the processed data. Most importantly, the lower-order data export format is split into forward and backward directions. For raw export this processing needs to be implemented.

3.3.2.1 Direction Control

Direction control is a crucial processing method. While the previous chapter introduced the mathematical and physical basis for the direction control, this section provides the practical rationale for its utilisation.

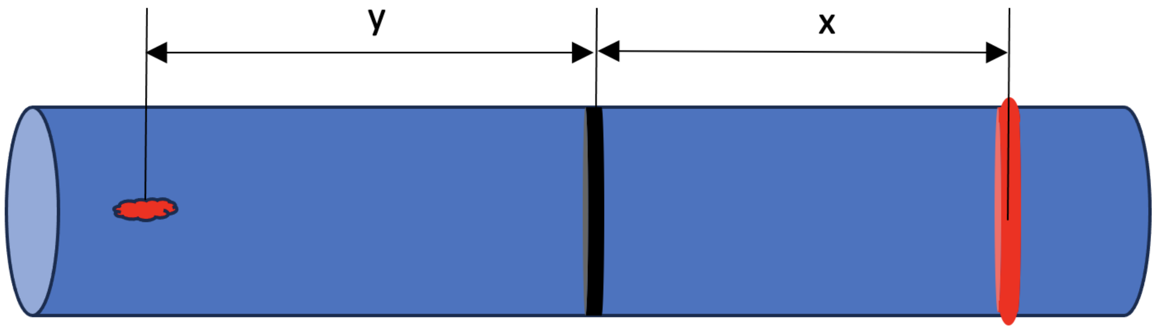


Figure 43. Pipe under bidirectional guided wave test. Two features (in red) are present, a weld separated from the inspection location (in black) by distance x and a defect separated by the distance y.

Consider a pipe under inspection as depicted in Figure 43. The features present on two sides of the transducer ring are separated from the ring by very similar distances. As such, with no direction control applied, the image resulting from the inspection is presented on the left in Figure 44. With the features located so close to each other, their separation and identification is made significantly harder.

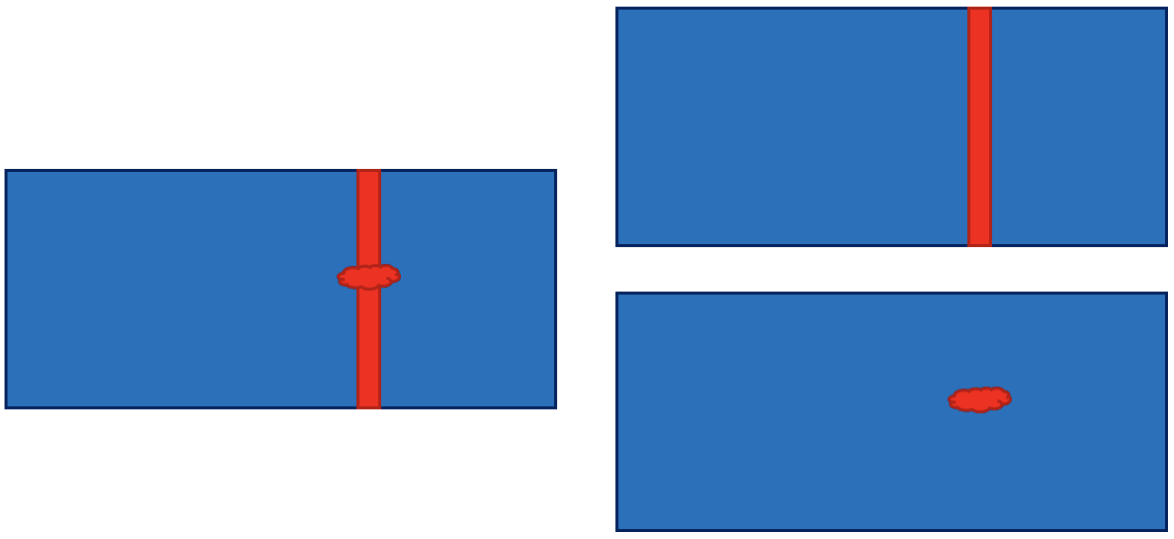


Figure 44. Image from the inspection of pipe presented in Figure 43 with no direction control implemented (left), compared to the image with direction control implemented (right).

Thus, the direction control is important to the data processing for the purposes of machine learning in two ways. First, it ensures there is no overlapping between two features existing in different directions improving the quality on the sample basis. Second, it makes more data available for creating negative (no feature) datasets. Assume drawing the negative samples from the images presented in Figure 44. The pristine span of the pipe before the features is effectively doubled by separating the directions.

3.3.2.2 T(0, 1) Weighting

After the directions of the signal are separated and the image is generated using the CSM approach described, an extra step is taken to improve the signal to noise ratio by weighting the amplitude of the image by a factor proportional to the T(0, 1) amplitude at the same axial distance.

The rationale behind the processing is:

1. T(0, 1) signal is less vulnerable to noise, as it is typically the highest amplitude of all the modes, and it can be extracted with minimal processing involved by summing up the circumferential displacements of all transducers.
2. T(0, 1) amplitude as a percentage of full reflection at axial location x corresponds directly to the change in cross-sectional area (pipe wall thickness loss) at that location.
3. The focused image of the pipe should be the map of the thickness change.
4. Thus, T(0, 1) amplitude at a location x is proportional to the sum of amplitudes of pixels in the image at the corresponding location.

Effectively, to perform this processing, the envelope of T(0, 1) signal is normalised between 0 and 1 within the inspection range. Assuming the image is a 2-D matrix I of $[x_n, y_n]$ shape, while the T(0, 1) trace is a vector of $x_n,$ length, the normalisation values are calculated as:

$$norm(x) = \frac{T(0,1)(x)}{\sum_{y=0}^{y_n} I(x,y)} \tag{67}$$

The resulting vector has the length of x_n, therefore the normalisation can be applied to the image by element-wise multiplying the vector by the image.

3.4 Data Issues

The introduction to the data used in this work is not complete without discussing the problems and limitations of the datasets, either the ones generated from FE or the experimental ones.

3.4.1 Real Data

The main limitation of the real dataset is the low amount of the data available. This is not, however, the focus of this section, which instead deals with the qualitative limitations of the data. These are typically not linked to the quality of the data itself, the noisiness or mislabelling, which are a common issue associated with public datasets, but the lack of standardisation and the processing methods hidden due to commercial interests.

3.4.1.1 Complex and Proprietary Processing

Unequivocally, the best data available for guided wave inspection are the GUL unrolled pipe displays. They are commonly the main tool used by the inspectors to localise and identify the feature of interest. While the imaging algorithms in their raw research form are available, the implementation leading to the best-in-class results is a result of a long and involved R&D process. Such a processing algorithm is obviously commercially valuable and thus it cannot be released outside of the company. The consequence is that every researcher working on ML for Guided Wave NDT and wishing to use images as the input data (which is the first instinct for much of the ML community) is bound to either develop their own imaging routine or use the one available to them. This, in turn, necessarily leads to the immense difficulty in comparing the quality of ML approaches between researchers, as any difference in the quality may be caused by either the pre-processing of the data or by the ML itself. Ideally, this issue would be solved by creating processing-as-a-service facility available to ML researchers, which would ensure the equal playing field as far as the access to the data is concerned. The second-best approach would involve creating a testing dataset, available at various levels of processing, so that an ML approach developed by an independent researcher can be validated against a known test.

3.4.1.2 Lack of Common Export Format

The second problem standing in the way of ML for NDT research from the side of industrial data is the lack of the common data storage and export format. This essentially necessitates the development of the full data pipeline matched to each source. While it has not been a huge hinderance in this work, as a singular data source was used, it would undoubtedly impact any attempt to create ML approaches based on data fusion or generalised across NDT modalities. Furthermore, the lack of the common file exchange format makes the data inherently less trustworthy, as it is not necessarily traceable. While some work is done on establishing the common exchange format [84], the problem essentially comes down to no organisation having strong commercial interest in making their data interoperable with anyone else. Conversely, the proprietary nature of the data format locks the customers into the hardware-software-training-support sales pipeline. While a similar issue has been recognised in the medical data processing and sharing, the medical field has the size and the public attention that forces the regulators and industry bodies to act in the interest of the patients, inadvertently making the data better suited for research. Lack of such characteristics of the NDT field coupled with the commercial organisation being the end user successfully prevents a parallel development.

3.4.2 Simulations

Opposite to the real data, the problems with the simulations are typically not related to the standardisation, as this can be tightly controlled using finite element software. They are, however, typically centred around the quality of the simulations, namely the lack in the ability to mimic certain characteristics of the real-world data. Such a category of issues may well be solvable in research setting where a limited number of simulations needs to be produced, but are difficult to generalise to parametric, large-scale generation, necessary for building ML datasets. The simulations were however used extensively in this work to develop proof of concept ML algorithms. Low performance of such models would point to their lack of ability to generally develop the understanding of GWT data and thus disqualify them from further research.

3.4.2.1 Simulating Complex Features

Two types of a feature are simulated in this work. The first one roughly corresponds to a defect and the second one roughly corresponds to a weld. Those are some of the most common features in the real dataset. The selection of welds is motivated by two factors: they are the most prevalent feature present in the real dataset and their geometry is simple, with the weld being a smooth and symmetrical local increase in the thickness. The defects are selected, as their detection is the goal of NDE making their omission in the simulated set a rather unreasonable proposition. However, looking at the composition of the real feature set, this approach to simulations fails to account for a range of other features. The second most common feature category are supports. Those, however, are quite a broad category, primarily split into simple supports (pipe resting on the support), welded supports, where a pipe is welded to the support structure and clamped supports, where screws, springs or hydraulic mechanism is applied to couple the support to the pipe. In each of those categories a wide range of geometries and sizes of the supports is available.

This brief description of the extensive range of the supports available elucidates the issue with their simulation. The specific wave interaction with the support is a very complex phenomenon, requiring an accurate modelling of the geometry of the support as well as its coupling with the main pipe. As such, every support to be simulated would need to be designed and validated by hand, making the generation of a dataset large and diverse enough for machine learning purposes an impossibility.

The second point that needs to be made on supports is the method of identifying this type of a feature employed by the inspectors. Supports are typically present in the guided wave inspection trace as weak non-axisymmetric features. While not necessarily difficult to spot, they are remarkably similar to defect signatures. The supports are distinguished from defects by their behaviour at various frequencies – the strength of their reflection typically changes with the changing inspection frequency. Secondly, when utilising the image view of the pipe allowing for the localisation of the signatures, the supports are usually located at the bottom of the pipe. The combination of the two characteristics allows the inspector to identify supports with good confidence. In conclusion, the simulation of support geometry and coupling is not feasible in the numbers required for machine learning, however, this could be alleviated by designing a feature of the simulation that exhibits similar frequency dependency and is located at the bottom of the pipe. While this has not proven necessary for this work it is a clear direction of development, should the need arise.

3.4.2.2 Simulating Realistic Noise Sources

The real world is very complex and there is a variety of noise sources that impact the predictive capability of the guided wave inspection. This work uses the coherent noise injection method that provides a good approximation of the coherent noise from a relatively clean pipe. There are, however, factors interfering with the signal that are significantly more difficult to simulate or more localised. The common ones are highly attenuative coatings, the environment of the pipe (i.e., it being buried or underwater), the impact of temperature variation and general corrosion. The main feature of those noise sources to distinguish them from the result of transducer unbalancing is their local character and potentially very high amplitude compared to signal. These two issues have been alleviated in this work by first, using relatively small defect indications, hence artificially depressing the SNR for defect indications, and secondly by using short snapshots of the trace disconnected from the broader context of the inspection, thus making every source of noise 'global' as far as the ML training is concerned. The ability to simulate realistic noise sources, however, would be of immense value to the future development of ML approaches to non-destructive testing, as it would enable the joining of local and global (inspection-wide) context.

3.5 Datasets Generated

Having introduced the data sources, characteristics, processing, the issues and the solutions, the final section of this chapter introduces the processed datasets upon which the ML algorithm can be trained. This section also provides the examples of the data from each dataset for the easy comparison of the simulated data to real.

3.5.1 Real Dataset

There are two real datasets generated, one is the result of extracting the raw export and running the imaging routine based on the common source method. The images are then manually labelled and segmented to generate the dataset. This dataset is divided only between the defect indications (positive class) and pristine (including non-defect features). The other dataset is generated from the processed Wavepro exports. This dataset is labelled and segmented automatically, with all the features labelled individually and fully traceable.

3.5.1.1 Imaging

The dataset containing the synthetically focused images of the defect indications in the experimental data has been created by first running the CSM imaging algorithm on the raw transducer data and applying the direction control. The result, presented in Figure 45 is an image similar to the ones typically analysed by the inspectors.

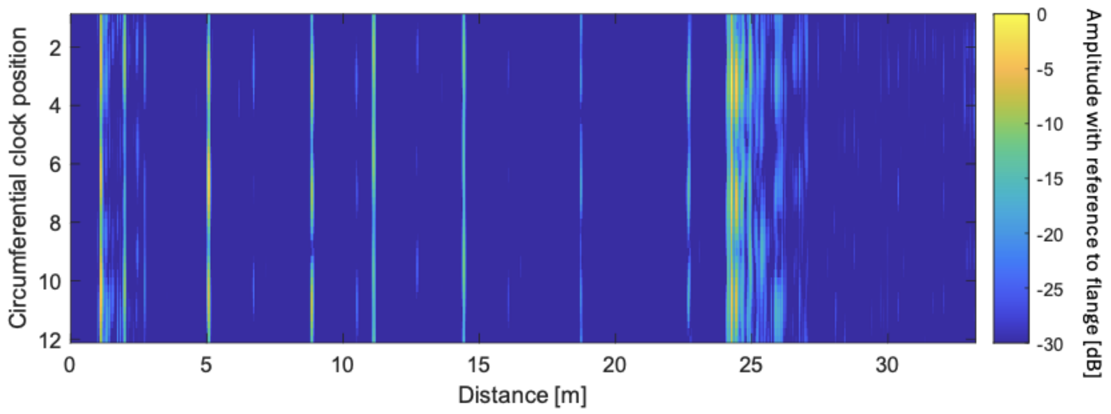


Figure 45. A Synthetically focused image of a pipe presented on a log scale with reference to the flange signal.

This image has the size of 8304 by 45 pixels. This is a very large size, which would require an extremely long time for both training and inference of the neural network. To rectify the issue, as well as to create as many training samples as possible from the limited dataset available, the image is segmented into 128-sample long windows (corresponding roughly to 1 m of two-way T(0, 1) wave propagation). An example of such a sample containing a defect indication is presented in Figure 46. As the raw export file contains the data corresponding to the range of probing frequencies, the image generated at the frequencies are joined, making the dataset a 4-D array of shape *[nImageSamples, 128, 45, nFreqs].*

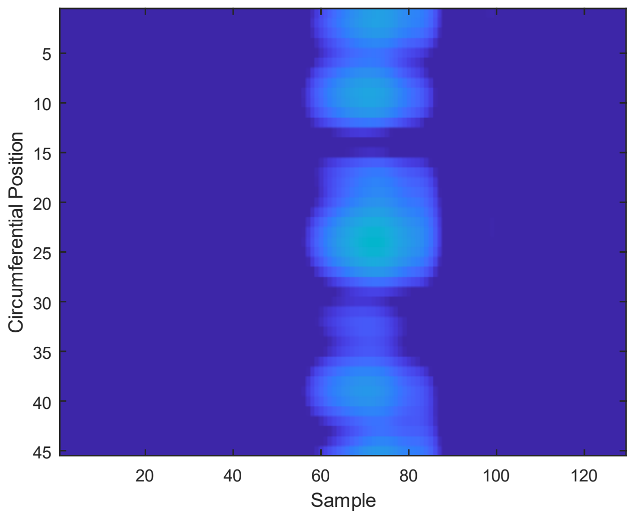


Figure 46. An image of a defect indication drawn from the pipe imaged in Figure 45, located at around 18.5 m axially.

The negative (pristine) samples are drawn from the data not containing any feature, which makes them quite abundant. The significant limitation of the data is the fact that there are only 32 positive samples available. It is important to note that there are more defect indications available as A-scans, as the raw exports are more difficult to work with and not all can be imaged to sufficient quality using the method described. The dataset has therefore been augmented by random shifting of the window, resulting in 160 defect indications in the augmented dataset (x5 augmentation).

To implement transfer learning, it is necessary to supplement the data with simulations. Similar processing has been performed on the simulated data to that of the experimental data. A comparison between the simulated and the real images is presented in Figure 47. Clearly, the simulated defect indication is significantly sharper and the whole image is less noisy, despite the efforts to inject the noise into the simulation.

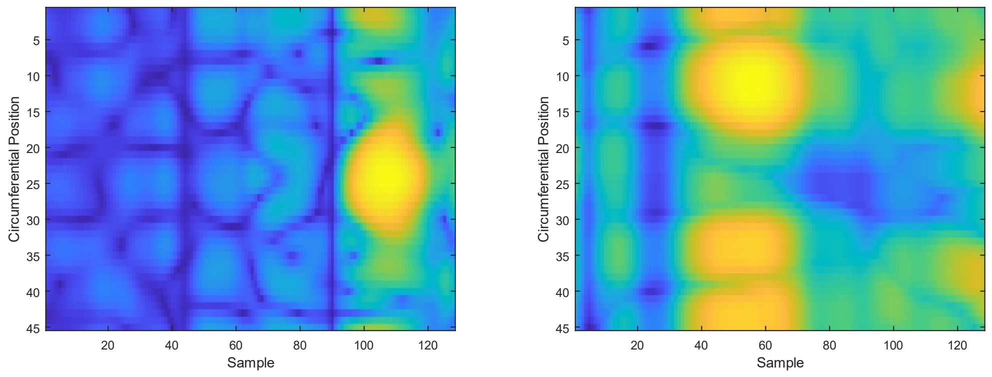


Figure 47. The comparison between an image of a simulated defect indication (left) and a real pitting-type defect indication (right).

3.5.1.2 A-Scan

The second version of the experimental dataset is composed of the time traces coming from the real inspections. As discussed earlier in the chapter, the data is drawn directly from the Wavepro export. To keep the A-scan and the image datasets comparable, the length of the trace is set to 128 samples. Each trace corresponds directly to a single feature or a pristine element of the trace. The features are fully traceable by retaining a unique file identifier and a unique feature identifier within the file. As the identifiers are separate for different types of features, it is immediately obvious upon the inspection of the dataset which features are present, or which have been classified correctly and incorrectly.

As opposed to the images, the processed exports are performed at a single but varying frequency, with the inspection frequency information not contained in the format. This is due to the data exported at a single frequency being dependent on the setting in the software at the moment of export. As such, the dataset is considered probing-frequency-agnostic. Figure 48 presents the examples of the data contained in the dataset. Each trace in the dataset is split between the T(0, 1) mode and F(1, 2) mode. Within each of those, both RF and envelope signals are retained. The dataset is thus of the shape *[nTraces, 128, 2, 2].*

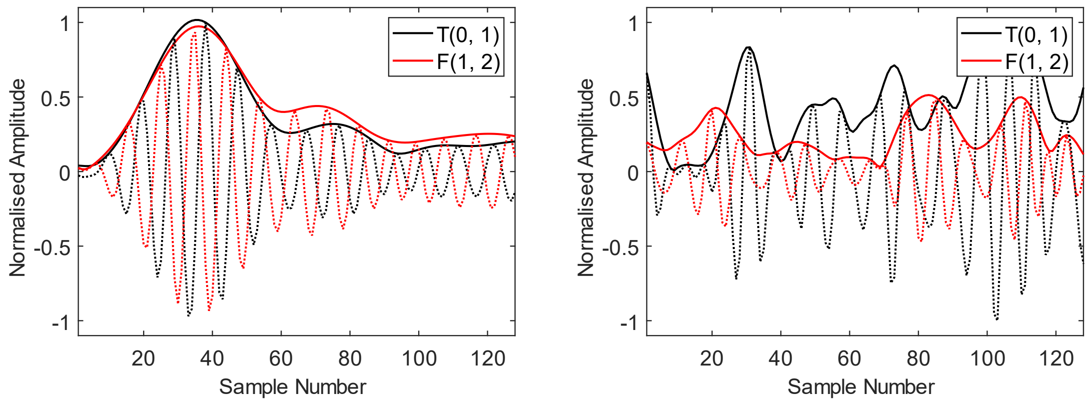


Figure 48. The examples of data contained in the A-scans dataset. A defect indication (left) and a part of pristine trace (right). RF trace is shown as the dotted line while envelope of the signal is traced in solid line.

The significant difference in this dataset, compared to the images, is its significantly higher quality, both trace-wise – the traces are processed using industry-standard techniques, and only windowed using the original algorithm, and dataset-wise – the export files are completely usable, thus, even comparing the number of defect indications, the total is 58, compared to the imaging positive dataset size of 32. This dataset too undergoes augmentation by shifting the window resulting in the dataset size of 406 (x7 augmentation). Finally, the supplementation of this dataset uses the data from the same simulations as the imaging dataset. In this case they are not fully processed into images, instead T(0, 1) and F(1, 2) modes are extracted. Figure 49 shows the comparison between the real and the simulated data.

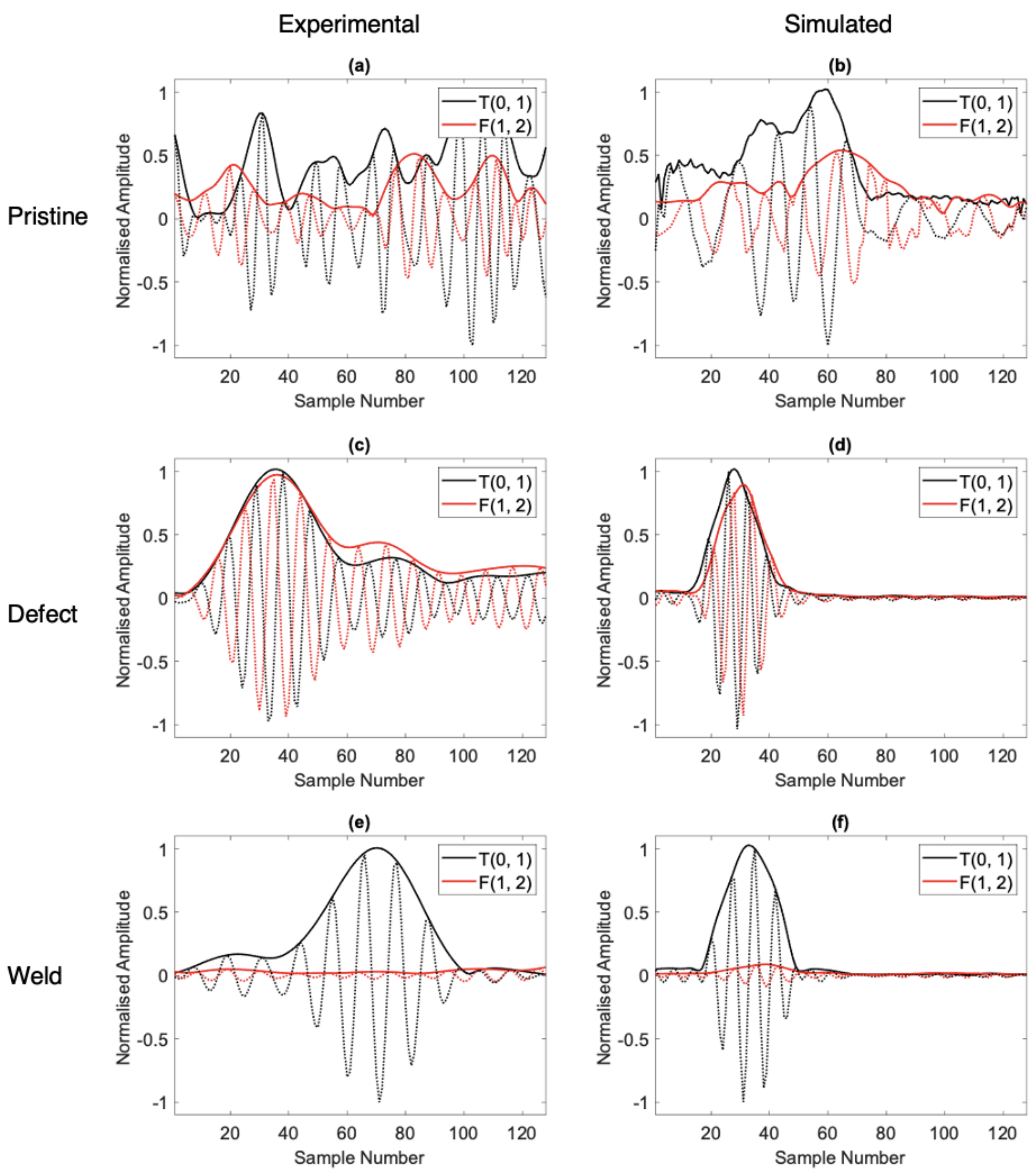


Figure 49. Examples of the data samples used for training the ML models. (a) Experimental pristine, (b) simulated pristine, (c) experimental defect indication, (d) simulated defect indication, (e) experimental weld, (f) simulated weld.

3.5.2 Simulated Dataset

The real datasets are supplemented by the simulations. As the previous sections introduced the examples from the simulated datasets as well as the detailed process for generating the simulations, this section introduces the datasets generated, their sizes and the types of features represented.

3.5.2.1 Simulated Data Range

As mentioned, the features simulated were welds and defects. At first, both types of features were simulated in a single model, which was then segmented into the traces or images, as required by the machine learning algorithm. However, as the need for larger datasets grew, the process was streamlined to simulating a single feature in a simulation. This has the advantage of simplifying the description of the feature (size, location, type etc.). Thus, the three main datasets used are composed of the various sizes of welds, various size of defects and pristine traces. Each of the datasets contains 6000 traces. Depending on the application and the algorithm, the size and selection of the dataset used for the training is varied. Furthermore, some special-use simulated datasets were generated, for example a dataset with zero, one or two defect indications designed for the training of a conditional generative adversarial network. Early in the project a data generation pipeline was developed allowing for mostly hands-off generation of large number of parametric simulations. When a new dataset was needed, only a modification of the model generation script was necessary, followed by the time needed for the simulation. In the case of a simulation described in this section, the single simulation is completed in around 1 minute, using a single Nvidia GeForce RTX2080Ti GPU.

4 Machine Learning Design

This chapter discusses the design decisions made when building the machine learning algorithm, from the metrics to the architectures and the training design. It builds heavily on the theoretical background, but it also introduces some of the finer points of the selected architectures and rationales for the specific design decisions. It introduces the novel approach to transfer learning based on the combination of simulated and real multimodal data.

4.1 Metrics

This section introduces the metrics used for the evaluation of ML models or indeed any decision-making processes. Subsequently, the selection of the specific metrics for the evaluation of the models developed in this work are discussed. This decision is critical for the success of the resulting ML model. Typically, the metrics used are specific to the class of problems rather than to the type of an ML algorithm used, thus in this work their selection is strongly motivated by the nature of guided wave inspection – a screening inspection expected to take in large amounts of data and intended to identify locations for follow-up inspection in a safety-critical environment.

4.1.1 Basic Metrics

The bases for all the following metrics are the concepts of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). For the following explanation, let us define the defect indication as a sample drawn from the positive class and a pristine trace as a sample from the negative class. The decision-making process under consideration must decide between the two classes. As such, we can summarise the problem and possible solutions in Table 2.

Table 2. Confusion matrix for a decision process between a defective and non-defective sample. The columns correspond to different states of reality and the rows correspond to different predictions.

True Positive (TP): Reality: The sample is defective Prediction: A defect indication is called Outcome: The unit under test is scheduled for further testing, prospective maintenance or replacement, breakdown averted	False Positive (FP): Reality: The sample is not defective Prediction: A defect indication is called Outcome: The unit is unnecessarily tested, replaced or maintained
False Negative (FN): Reality: The sample is defective Prediction: A defect indication is not called Outcome: The unit suffers breakdown in service	True Negative (TN): Reality: The sample is not defective Prediction: A defect indication is not called Outcome: Normal operation continues without interruption

Generally, the predictions can be grouped into true (correct) and false (incorrect). Every decision algorithm aims to maximise the true predictions and minimise the false ones. However, for most classifiers the output is a value between 0 and 1, with the decision threshold selected by the algorithm designer or user. If the output for a given sample is above the decision threshold, the sample is classified as positive, when below, it is negative. As such, when the decision threshold is reduced, the algorithm is weighted towards classifying more samples as positive (i.e., minimising false negatives while increasing false positives) with the reverse effect when increasing the threshold. The confusion matrix (a layout similar to Table 2, containing the counts of TP, TN, FP, and FN) calculated for every possible threshold is the full and unbiased metric of the performance of the algorithm. However, making comparisons on the performances of different algorithms based on the full size of this matrix, i.e. $(Number\ of\ Classes)^2 * (Number\ of\ Thresholds)$, would be rather challenging. Therefore, metrics summarising the confusion matrix are preferably used.

4.1.2 Accuracy

Accuracy is defined as

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} = \frac{TP + TN}{TP + TN + FP + FN} \tag{68}$$

It is the most basic and naturally understandable metric, and it does a good job for many tasks. Consider, however, a scenario in which the algorithm is applied to 100 samples, of which 90 are pristine and 10 are defective. The algorithm classifies one defect indication correctly and misses the rest of them, while being faultless on the pristine samples. Clearly this is a very poor defect indication detector that misses 90% of the defect indications. In this scenario TP = 1, TN = 90, FP = 0, FN = 9, the accuracy is therefore calculated as:

$$Accuracy = \frac{1+90}{1+90+0+9} = \frac{91}{100} = 91\%, \tag{69}$$

giving the incorrect impression that the model performs well. The cause of that behaviour is the imbalance between the positive and negative classes, which accuracy is very vulnerable to. This work is concerned with such an imbalanced dataset, leading to the need for metrics invulnerable to the problem.

4.1.3 Precision and Recall

Precision is defined as:

$$Precision = \frac{TP}{TP+FP} \tag{70}$$

It is therefore a measure of how likely the positive prediction is to correspond to a positive reality. A model that produces no false positives has the precision of 1.

Recall is defined as:

$$Recall = \frac{TP}{TP+FN} \tag{71}$$

It therefore answers the question of “What proportion of positives are true positives?”.

Referring to the example presented in the description of accuracy, the metrics would be:

$$Precision = \frac{1}{1+0} = 100\% \tag{72}$$

$$Recall = \frac{1}{1+9} = 10\% \tag{73}$$

Maximising precision requires the minimisation of false positives, while maximising recall calls for minimising false negatives. Clearly the two have conflicting imperatives, thus the two metrics must be considered together, as simply classifying every sample as positive results in 100% recall while classifying a single positive sample correctly (and no negative samples as positive) gives 100% precision.

A metric that combines both precision and recall is the F1 score. It is defined as a harmonic mean of the two:

$$F_1 = \frac{2}{recall^{-1} + precision^{-1}} = \frac{2TP}{2TP+FP+FN} \tag{74}$$

This metric combines precision and recall into a single number. They are, however, not necessarily equally important. In NDT and in screening methods in particular, high recall, i.e., not missing defect indications, is more important than not calling false positives. Secondly, the F1 score varies for the same algorithm depending on the selected classification threshold. Due to that issue, a set of metrics invariant to the selection threshold have been developed.

4.1.4 Area Under Precision-Recall and Receiver Operating Curves

As discussed, precision and recall values depend on the selected detection threshold. As such, it is possible to calculate their values at different thresholds and plot the values on a graph, known as the precision-recall curve (PRC). A similar approach can be used for a different set of values – true positive rate (TPR) and false positive rate (FPR). True positive rate is synonymous to recall and defined as:

$$TPR = \frac{TP}{TP + FN} \tag{75}$$

False Positive Rate is defined as:

$$FPR = \frac{FP}{FP + TN} \tag{76}$$

The plot of TPR vs. FPR at various detection thresholds is known as Receiver Operating Curve (ROC).

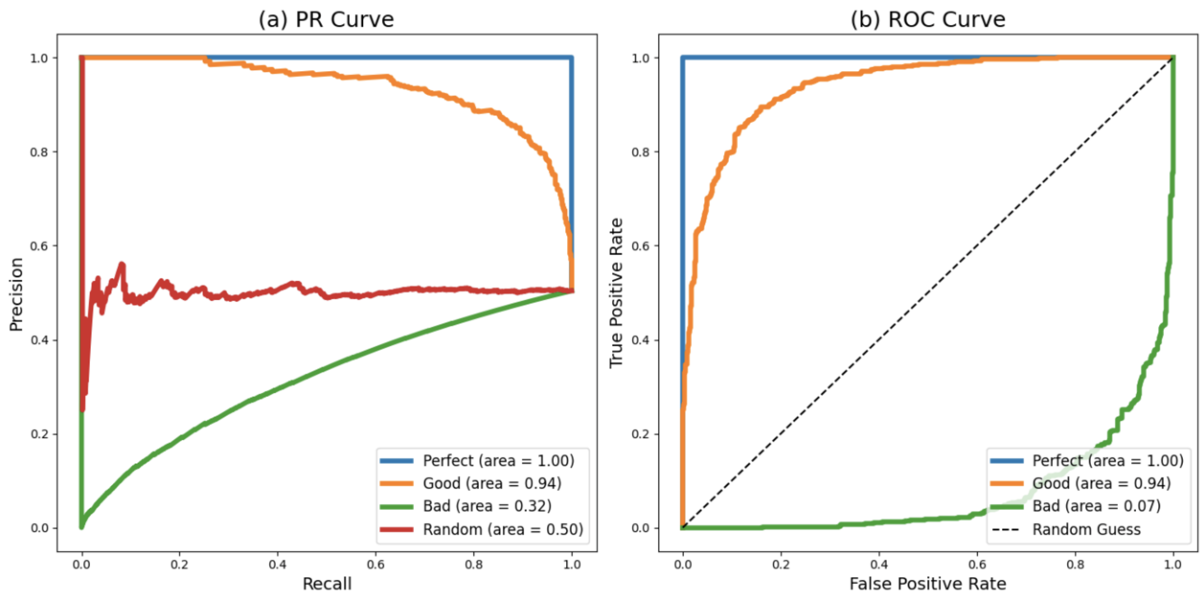


Figure 50. Schematic depiction of Precision-Recall Curves (a) and Receiver Operating Curves (b).

Figure 50 presents an example of the PRC and ROC. For PRC the perfect classifier is represented by the point in the top right corner – with precision and recall both equal to 1.

PRC is not necessarily monotonic, as exemplified by the random classifier graph. For ROC the true positive rate increases monotonically with false positive rate, and the speed of the increase corresponds to the quality of the classifier, with the perfect classifier corresponding to the curve which includes point (0, 1), i.e., 0 FPR with 1 TPR. The ROC is very human-readable, and it provides an immediate overview of the level of performance of the classifier and the type of the FPR/TPR trade-offs it tends to make when the decision threshold changes. Its downside is the lack of a numerical output, making the comparison between large numbers of models difficult. To rectify this issue, area under ROC (AUROC) can be calculated. While this metric loses some of the trade-off information, it is nonetheless useful, and it is considered the gold standard for model assessment.

Compared to the ROC, PRC is used in rarer instances, as it is sensitive to the decision of which class is to be defined as positive and which as negative. This work utilised both curves for some of the experiments, but the results for PRC and ROC closely tracked each other and given this redundancy for much of the work ROC is utilised.

One of the major draws of AUROC is its detection threshold invariance, allowing for the comparison of models 'as such'. However, in the NDT context the goal is the successful classification of all the defect indications, with less importance attached to the avoidance of false calls, in line with the relative cost of either mistake. Furthermore, false calls can be checked with other methods, such as conventional UT, making them a lower priority. In fact, in guided wave testing the calls are typically verified with a follow-up inspection. This has motivated the development of a bespoke performance measure for the ML models, designed for NDT.

4.1.5 False Positive Rate at 100% True Positive Rate

As the machine learning algorithms in guided wave testing are meant as a method of assisting the inspectors by flagging all the points at which the defect indications may occur, it is a requirement for an algorithm that none of the defect indications are missed. The false positives are a secondary concern as all the positive indications are assessed by the human operator. Thus, the false calls can be filtered out before expensive repairs are allowed to take place.

Given that the non-negotiable characteristic of any ML algorithm utilised in this way would produce virtually zero false negatives, and that the number of false negatives vs. false positives can be manipulated by varying the detection threshold, a measure of the quality of the algorithm is the false positive rate calculated at the threshold at which none of the positives are missed. This metric is termed FPR@100%TPR.

The advantage of this metric is its immediate industrial usefulness, as it directly gives the number of the false positives that need to be investigated, which has a clear equivalent cost in the inspector's labour.

A caveat of the metric is that the threshold for not missing any defect indications is determined by the testing set, thus an overly easy or difficult to classify dataset inevitably skews the results. This problem can be addressed by extensive cross-validation using a variety of training and testing set combinations. Secondly, the metric threshold necessarily needs to be relaxed from 100% in the case of larger datasets, as even under the normal distribution conditions the metric-breaking outliers exist, due to the probability density function never reaching zero.

Furthermore, the metric does not allow any flexibility by imposing the hard limit of zero false negatives. As a result, a single very difficult-to-classify positive sample may have a significant impact on the metric's value. This problem is most acute if a positive sample is actually a negative one being mislabelled. As such, this metric needs to be utilised carefully, ideally by looking for patterns across the various testing sets and carefully investigating positive misclassified samples. Such an approach has been adopted in this work. Furthermore, in order to maintain the statistical meaning of the metric, the actual TPR used is 99.7% rather than 100% in line with the three-sigma principle.

4.1.6 Metrics Selection

Accounting for the considerations presented in this section, the metrics selected for the appraisal of the machine learning models are divided into the category used for manual, qualitative evaluations of a specific model and another one used for large-scale statistical comparisons between models.

The first category encourages using the richest data possible, as it is examined by hand. The ROC curve is chosen, as it is easily readable, and provides the detail of the classification performance at the single sample-level granularity.

The category used for the statistical analysis requires a numerical output. At various stages of the research, different metrics are used. AUROC is used in all contexts, providing a useful overview and a comparative metric. At the initial stages AUPRC is used in addition to AUROC to check for any discrepancies. Similarly, accuracy is used to provide a quick, intuitive overview of the quality of the model. Once the high-performing model is identified, FPR@100%TPR is used to provide a metric that has industrial relevance and is sensitive enough to show a tangible difference between various very high-performing models.

4.2 Architectures

This section deals with perhaps the most complex element of the machine learning model design – the selection and development of the architecture. As this work is mostly focused on the proof-of-concept and as no architectures have reached industry-standard level in NDT, it takes a broad view in utilising the well-established architectures and modifying them to suit the problem. The modifications do not involve structured hyperparameter tuning studies or large-scale architecture comparisons, as the primary requirement for this type of work is major computational resources and the typical outcome is marginally better performance. While valuable, such undertakings are better-suited to implementation work in the industrial setting than to the academia. The architectures implemented are Multilayer Perceptron (MLP) – the first implementation of deep learning, VGG-Net – a representative convolutional neural network, and U-Net – a modern convolutional neural network, considered a gold standard in many image-based applications. This section does not present the comparison between them, but the rationale for their selection and the modifications made to increase their suitability for the GWT question.

4.2.1 Multilayer Perceptron

Multilayer perceptron [85] is a neural network whose origins can be traced to the beginning of the machine learning as a discipline. It is a generalisation of Rosenblatt's perceptron [35] using multiple neuron layers and introducing non-linear activation functions. The mathematical operation of the neurons, activation functions and the training process for such an architecture is introduced in more detail in the Theoretical Background.

MLP is used in this work as a benchmark scenario, enabling the assessment of whether the architecture selection has any impact on the performance of the model and thus addressing the common question of whether the machine learning approach provides any improvement over the alternative methods.

The input data to MLP in this work was limited to the A-scans. The exclusion of artificially-focused (C-scan) data is motivated by the very high computational requirements of training a fully-connected network when compared to CNN when they are used on higher-dimensional image-type data. Still, the A-scan as used in this work is 3-dimensional data, albeit with two very small dimensions (T(0, 1)/F(1, 2) and envelope/RF). As fully connected layers are not suited to multidimensional data, the data first needs to be flattened. The flattening process disjoints the values of various modes for the same point; from the perspective of the model, the values of T(0, 1) and F(1, 2) referring to a same temporal instant are not connected. This departs from the human approach to the inspection process and removes the engineering insight. Consequently, the architecture is predicted not to be best-suited to the problem but is useful as a baseline case.

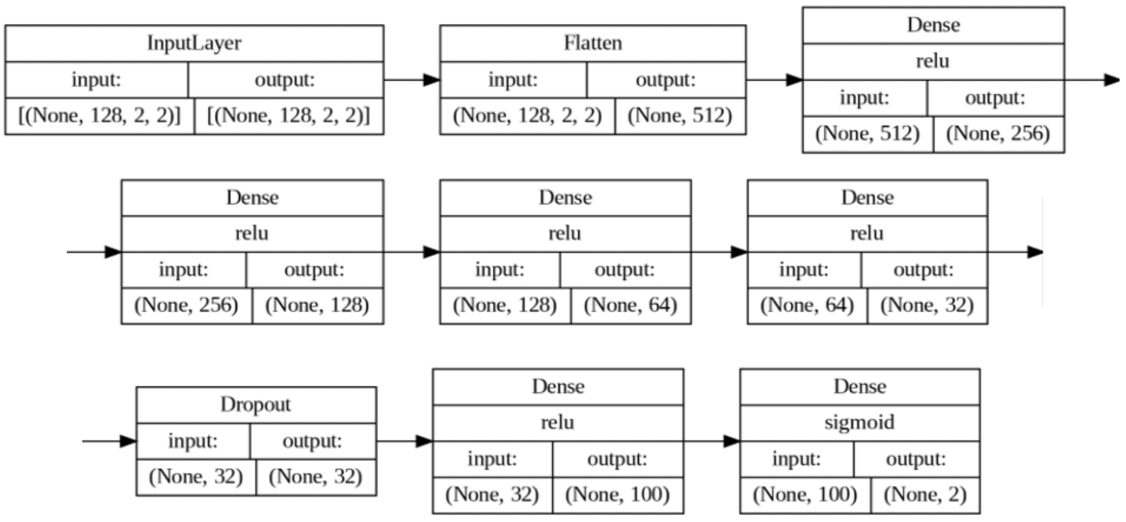


Figure 51. MLP implementation utilised in this work. The first two rows represent the feature extractor part of the network, while the third row is the classifier head.

The implementation of MLP in this work is presented in Figure 51, with the hyperparameters of the network tuned by sequentially increasing its complexity while assessing the performance. As the input data is three-dimensional, it is first flattened and then passed into a series of four dense layers, which are followed by a 40% dropout layer and two additional dense layers. The activation function is ReLU at all layers except from the output, where it is sigmoid. Figure 51 presents the subsequent layers of the network, The type of the layer, activation function, input and output shapes. The input and the output shapes follow the Tensorflow convention [(None, 128, 2, 2)]. The first dimension, None, is a placeholder denoting an example is a part of the batch. It is not defined at the network architecture definition stage, as in principle the network can take any batch size, thus it is defined at runtime. Following, 128 is the length of the sample, 2 is the T(0, 1) and F(1, 2) channel and the last 2 is RF and envelope channel. The same convention is utilised throughout the rest of this work. The output of this network is of shape (None, 2), similarly the first dimension is the batch size while the second are the probabilities for two classes (feature and pristine).

The design decisions pertaining to the architecture are mostly motivated by the data characteristics. The network is relatively shallow, which is motivated by the few data points available to train it. Similarly, the high dropout rate of 40% is meant to allow the network to learn only the most general characteristics of the data. This is especially important, as the network is meant to learn both on the simulated and the real data. It is understood that while the general characteristics of the simulations and the real data are similar, the details are significantly different. The simulated data comprises a vast majority of the training dataset, thus allowing the model to learn the fine details of the training data would inevitably cause it to focus on the simulations, contrary to the goal of performance on the real data.

4.2.2 VGG-Net

VGG-Net [86] is a canonical example of a convolutional neural network, whose main building blocks are convolutional layers and max-pooling layers, followed by a densely-connected classification head, all of which are introduced in the Theoretical Background. The first CNNs started appearing in the early 2010s, but the VGG-Net has made the major stride of increasing the depth of the CNN while solving the computational cost challenge by utilising small receptive fields (filter sizes). While the computational cost problem has been alleviated by the improvement in the computational capacity, the deep CNN as a concept is still a primary approach to a novel location-sensitive classification task, such as the GWT question.

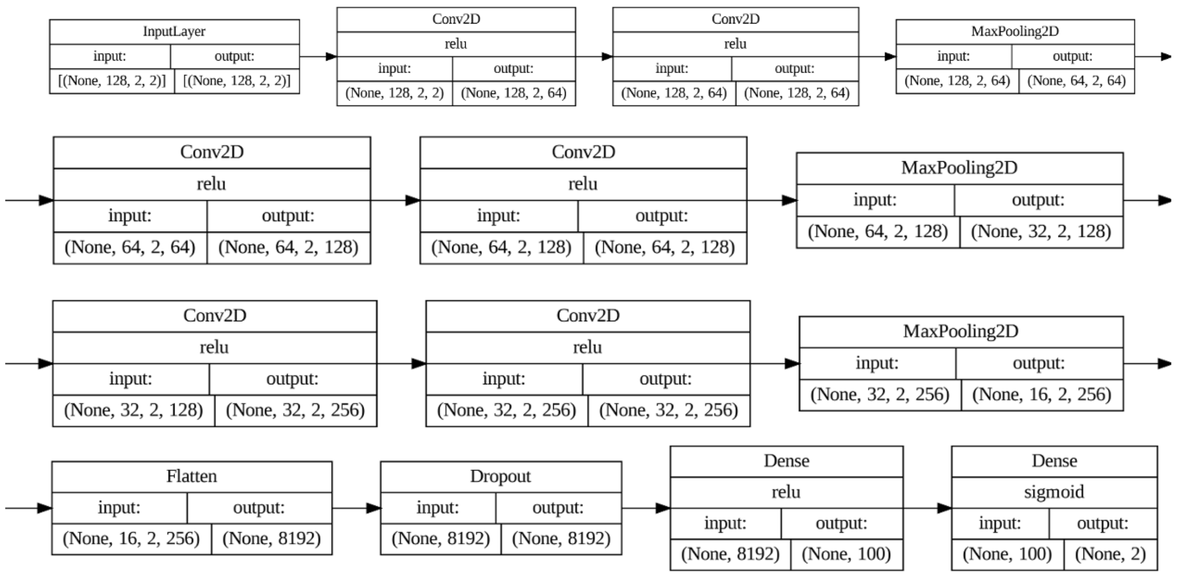


Figure 52. VGG-Net architecture implemented for this work. Rows 1-3 represent the feature extractor part of the network, while row 4 is the classifier head.

The implementation used in this work is presented in Figure 52; the input and output shapes follow the same format as Figure 51 with the same input and output shapes. The convolution in this neural network occurs over the dimensions -2 and -1 (i.e. 128 and 2 in the first layer). The final two dimensions are effectively interchangeable, they could also be concatenated to lower the dimensionality of the data, making it $(128, 4)$ shape. However, due to their different physical meaning, they are retained to facilitate experimentally disabling dimensions. Similarly to MLP, the changes in the architecture are motivated by the nature of the data, however in this case the major motivator for redesign is the number of the training data points. The original implementation was designed as an entry to ImageNet large scale visual recognition challenge (ILSVRC) [87]. The dataset for training in the ILSVRC challenge contains over 14 million images, over 1000 times more than the dataset used in this work, allowing for far larger number of filters and layers. Additionally, the challenge calls for the recognition between 1000 classes, making the problem significantly more complex than the GWT question.

The original VGG-Net uses a receptive field sized 3x3. This was motivated by the input data, being square images from the ImageNet dataset. In the case of the implementation in this work the receptive field is 20x2, with the second dimension limited by the data size. The larger receptive field demonstrably improved the performance of the ML model, probably due to most features of interest having quite large spatial extent. The second major change from the original VGG-Net is the introduction of the dropout layer between the convolutional layers and the classification head. This is motivated by the same factors as the analogous operation in MLP. Finally, the classifier head is shallower than the original, with two layers as opposed to three and a significantly reduced number of neurons. This is motivated by the much smaller training dataset and a visible tendency of the network to overfit when using many parameters in the classification head.

4.2.3 U-Net

U-Net is a more advanced architecture, which was first introduced in 2015 for biomedical image segmentation [63], and later used in a variety of contexts including non-destructive testing [88], [89], [90], the discriminator sections of generative adversarial networks [91], [92], image denoising [93] or speech enhancement [94]. U-Net and its evolutions are state of the art in segmentation tasks and thus also well suited to detection. The architecture is based on a contraction path followed by a symmetric expansion path. Given *n* layers in the contraction-expansion region of the architecture, the shapes of layers 1 and *n*, 2 and *n*-1… are identical which allows for the introduction of skip connections between them.

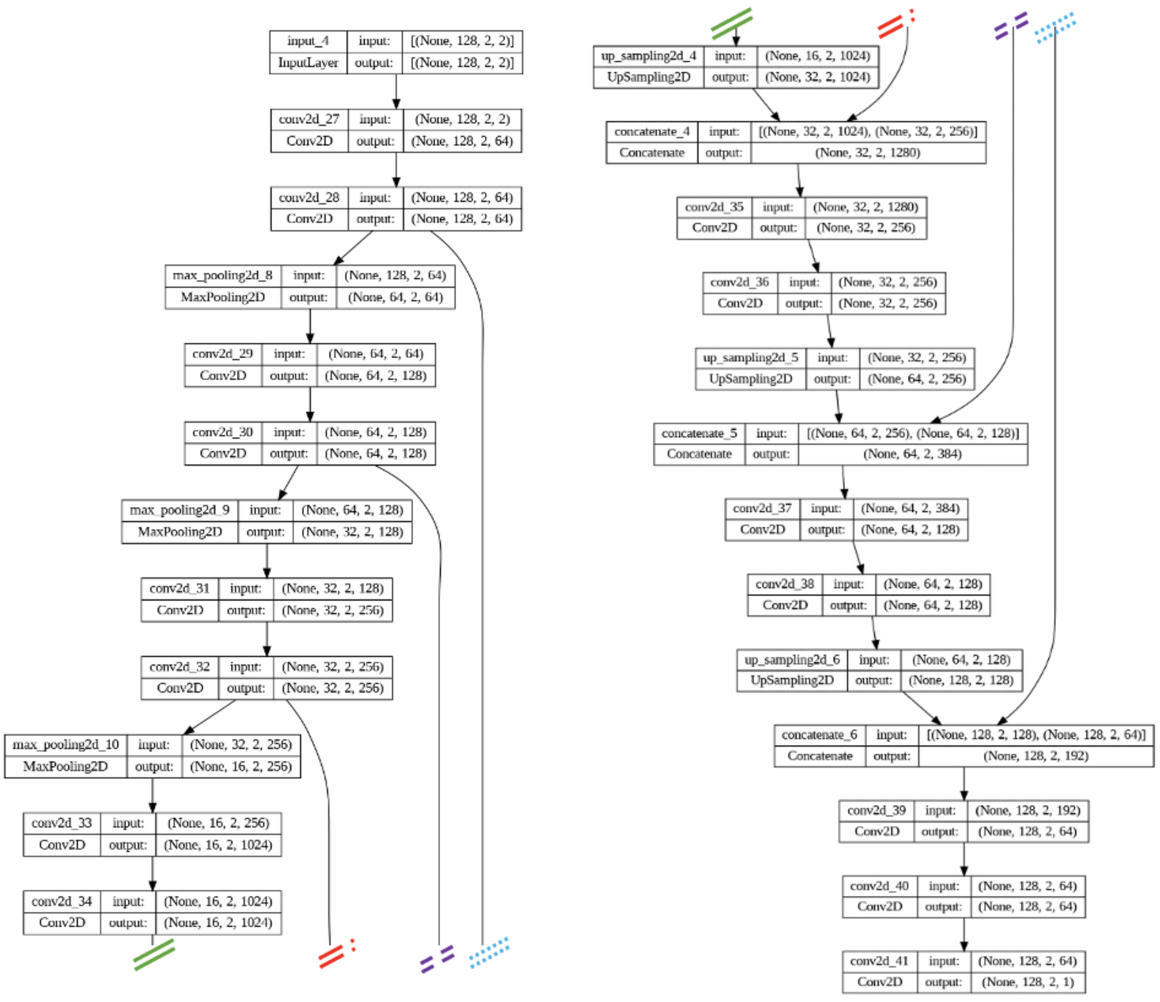

input_4 | input: | [(None, 128, 2, 2)]
InputLayer | output: | [(None, 128, 2, 2)]
conv2d_27 | input: | (None, 128, 2, 2)
Conv2D | output: | (None, 128, 2, 64)
conv2d_28 | input: | (None, 128, 2, 64)
Conv2D | output: | (None, 128, 2, 64)
max_pooling2d_8 | input: | (None, 128, 2, 64)
MaxPooling2D | output: | (None, 64, 2, 64)
conv2d_29 | input: | (None, 64, 2, 64)
Conv2D | output: | (None, 64, 2, 128)
conv2d_30 | input: | (None, 64, 2, 128)
Conv2D | output: | (None, 64, 2, 128)
max_pooling2d_9 | input: | (None, 64, 2, 128)
MaxPooling2D | output: | (None, 32, 2, 128)
conv2d_31 | input: | (None, 32, 2, 128)
Conv2D | output: | (None, 32, 2, 256)
conv2d_32 | input: | (None, 32, 2, 256)
Conv2D | output: | (None, 32, 2, 256)
max_pooling2d_10 | input: | (None, 32, 2, 256)
MaxPooling2D | output: | (None, 16, 2, 256)
conv2d_33 | input: | (None, 16, 2, 256)
Conv2D | output: | (None, 16, 2, 1024)
conv2d_34 | input: | (None, 16, 2, 1024)
Conv2D | output: | (None, 16, 2, 1024)
up_sampling2d_4 | input: | (None, 16, 2, 1024)
UpSampling2D | output: | (None, 32, 2, 1024)
concatenate_4 | input: | [(None, 32, 2, 1024), (None, 32, 2, 256)]
Concatenate | output: | (None, 32, 2, 1280)
conv2d_35 | input: | (None, 32, 2, 1280)
Conv2D | output: | (None, 32, 2, 256)
conv2d_36 | input: | (None, 32, 2, 256)
Conv2D | output: | (None, 32, 2, 256)
up_sampling2d_5 | input: | (None, 32, 2, 256)
UpSampling2D | output: | (None, 64, 2, 256)
concatenate_5 | input: | [(None, 64, 2, 256), (None, 64, 2, 128)]
Concatenate | output: | (None, 64, 2, 384)
conv2d_37 | input: | (None, 64, 2, 384)
Conv2D | output: | (None, 64, 2, 128)
conv2d_38 | input: | (None, 64, 2, 128)
Conv2D | output: | (None, 64, 2, 128)
up_sampling2d_6 | input: | (None, 64, 2, 128)
UpSampling2D | output: | (None, 128, 2, 128)
concatenate_6 | input: | [(None, 128, 2, 128), (None, 128, 2, 64)]
Concatenate | output: | (None, 128, 2, 192)
conv2d_39 | input: | (None, 128, 2, 192)
Conv2D | output: | (None, 128, 2, 64)
conv2d_40 | input: | (None, 128, 2, 64)
Conv2D | output: | (None, 128, 2, 64)
conv2d_41 | input: | (None, 128, 2, 64)
Conv2D | output: | (None, 128, 2, 1)

Figure 53. U-net architecture utilised in this work. The double lines mark the connections between the layers with the separate colours and line types used to differentiate between the connections. The architecture is based on the VGG-Net shown in Figure 52.

The skip connections work by helping to retain the information from the early stages of the neural network. The input to the neural network in the case of GWT data is the amplitude of the wave at each location. Clearly, that information is important in the decision whether the sample indicates a defect, as the amplitude can be considered the only input to thresholding, which in turn is used to assist the operators in determining where the defect indications are (recall call level DAC). The neural network transforms the input information and extracts the features it considers useful. Thus, the skip connections are used to inject the outside knowledge into the neural network decision-making process by making sure the amplitude information is not lost in the feature extraction process.

The U-Net was selected as the modern architecture for comparison, as it is simple enough to modify and to compare directly with the VGG-Net and MLP (opposed to state-of-the-art architectures, such as YOLOv8 [95]), but at the same time it allows for the localised skip connections (as opposed to architectures such as ResNet). As a result, in the final layer pre-classification contains localised information about the initial trace amplitude as well as the features extracted by the deep neural network.

The U-Net is modified for the GWT problem mostly due to the slightly different question posed. The original U-Net outputs a segmentation map, while in NDT the ideal output is binary classification. While the segmentation approach could be utilised in GWT, in the interest of comparing the outputs of different architectures, classification is used.

The U-Net implemented is based directly on the VGG-Net presented architecture in Figure 52. The feature extractor part of that architecture is used as the contraction path, followed by the symmetrical expansion path. The expansion path, in turn, is followed by the classification head identical to the two previously discussed architectures.

4.3 Training Design

This section deals with the parameters for training the machine learning models. The first sections introduce the parameters common to all the architectures mentioned: the loss function, optimiser, and the number of epochs. The second part addresses the biggest challenge in the approach faced in this work – the combination of the real and the simulated data so that a machine learning model performs better than one trained on just one type of data.

4.3.1 Training Parameters

4.3.1.1 Number of Epochs

The number of epochs in machine learning context is defined as the number of runs through the training set. This should not be confused with the number or updates (also known as training steps). The full training set is split into batches, thus the number of steps per epoch can be calculated as $Steps\ per\ Epoch = \frac{Training\ Set\ Size}{Batch\ Size}$. Selecting the number of epochs for which the model is trained is one of the important training design considerations. Generally, the upside of increasing the number of epochs is fitting the model better to the training data. There are two potential downsides – first, increasing the number of epochs linearly increases the duration of training. Second, increasing the number of epochs improves the fit to the training set, which, especially in the case of training sets as small as the one used in this work, may lead to overfitting and the decrease in the performance in testing. The overfitting problem is solved by splitting the data into training and validation – the latter not being used for model parameter updates. After each epoch the model performance on the validation set is checked and recorded. Furthermore, the parameters of the model performing best on validation are saved and reloaded at the end of training. This approach effectively solves the problem of overfitting. Thus, the number of epochs is limited only by the computational resources and the willingness to wait for the training to complete. The number of epochs necessary for fitting the model should be assessed on the largest dataset available and the most complex neural network architecture, as it is the size of the dataset and the network that affect the speed of training.

The largest dataset is the simulated dataset while the most complex neural network is the U-Net. Thus, these inputs are used in an experiment where the network is trained over 100 epochs, with the loss values recorded for both training and validation sets. The results of this experiment are presented in Figure 54. While the loss value decreases with the training, much of the improvement for this problem takes place over the first 10 epochs. Technically, to further lower the loss, the training could be continued for longer than 100 epochs. It should be noted, though, that the dataset the model is trained on is simulated. The loss function for the binary classification is calculated as the difference between the output probability for the correct class and the ground truth (0 or 1). As the output of the sigmoid function, the output of the neural network is only asymptotic to 1 and the loss will never go to 0. Indeed, the linear fall on the logarithmic graph points to the loss function lowering in line with the sigmoid function output getting closer to the target. Furthermore, it should be noted that the AUROC value for training and validation reaches 1.0 at the similar time as most of the loss decrease takes place. Thus, for the applications where it is used as the pre-training dataset, the number of epochs is limited to 30, as the performance on this dataset is not the target. It is worth noting that validation loss is consistently lower than training loss. While seemingly counterintuitive, the cause lies in the different model behaviour in training and in validation. To prevent overfitting to the training set, dropout (randomly deactivating neurons) is used. Thus, in training the performance of the network is artificially lowered, with the same not being the case in validation.

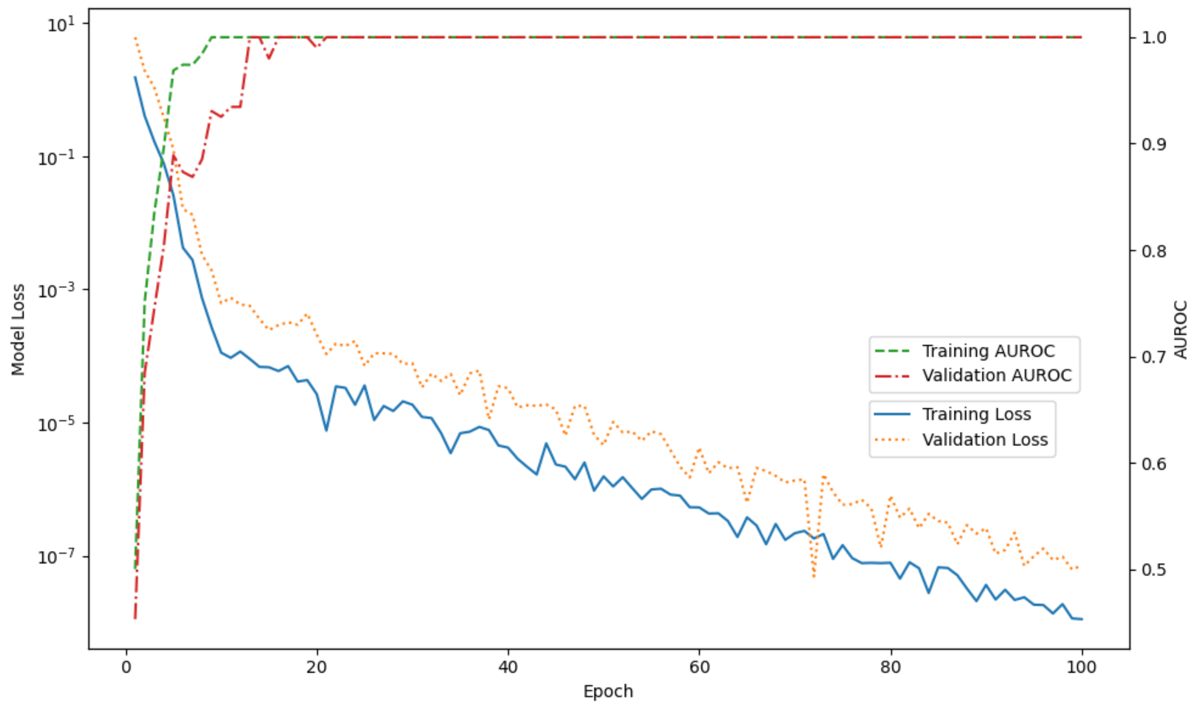


Figure 54. Training and validation loss values as well as training and validation AUROC values for a U-net trained on 10000 simulated examples for the first 100 epochs.

4.3.1.2 Loss Function

The decision on the loss function depends on the research question asked. In the case of GWT question the problem is binary classification. For binary classification problems, by far the most used loss function is the previously introduced binary crossentropy. The other loss functions under consideration were hinge loss and focal loss.

Hinge loss is a loss function given by the formula:

$$Loss = \max([1 - y_{true} y_{pred}], 0) \tag{77}$$

where y_{true} values are expected to be 1 or -1. The main characteristic of hinge loss is the punishing of the model's indecisiveness, specifically by assigning loss to correct predictions close to the decision boundary, lowering linearly when approaching the precisely correct prediction.

This behaviour is undesirable in the task, as it clashes with FPR@100%TPR as a metric and removes the information on how sure the neural network is of its predictions, which is useful in the context of explainability.

The previously introduced focal loss is designed to deal with highly imbalanced datasets. The defect datasets seem well suited to this loss. In the experimental practice the focal loss showed no improvement over binary crossentropy, with the added downside of the loss requiring the hyperparameters to be tuned to the specific dataset. Therefore, it is rejected in favour of standard binary crossentropy.

The final selection of binary crossentropy as the loss function is motivated by the performance on par with the alternatives and the fact that it is the most used loss function for binary classification questions. The second characteristic makes the potential modifications simple and the results of this work potentially applicable to a wider variety of problems (explainability, confidence calculations etc.).

4.3.1.3 Optimiser

The selection of the optimiser is a two-stage process – first to determine the algorithm itself, then to establish the parameters. Depending on the optimiser used, various parameters are needed, with the learning rate used for every optimiser and more advanced ones using additional terms including regularisation and momentum. The parameter with the most impact is the learning rate, governing the overall pace of training. Its value can be set to a constant or varied throughout the training process using learning rate schedulers. Clearly the decisions are manifold and based to a large degree on the experience of the designer. In this work the initial settings are chosen based on the best practices and experience and validated experimentally against viable competitors.

The decision on the optimiser algorithm is based on the vast prevalence of Adam optimiser. Figure 55 shows that as of the 30th of September 2023 Adam had been used in 13573 of research papers included in the "Papers with Code" database [69] compared to 1592 using Stochastic Gradient Descent and 466 using Adafactor.

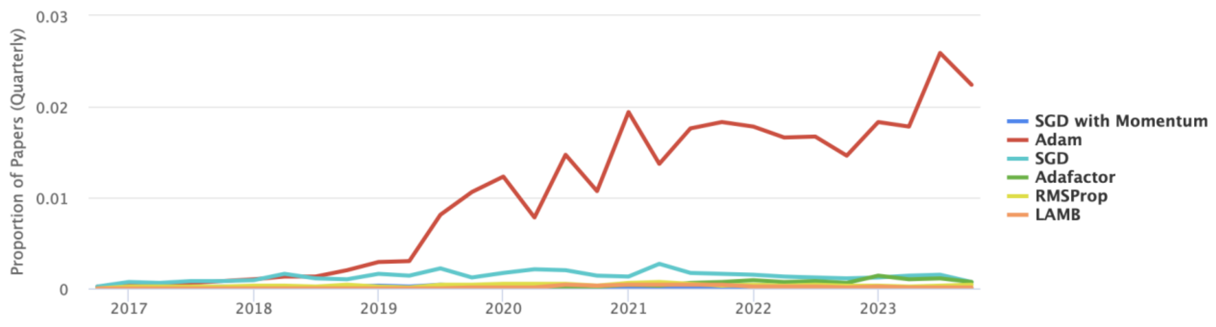


Figure 55. Proportion of new research papers in Papers with Code database using the most popular optimisers [69].

Based on that data, Adam is the natural first choice for the optimiser and it is expected to outperform the competition. The second most prevalent optimiser is the stochastic gradient descent; thus, the first experimental comparison is the training of the same neural network and comparing the training curves for Adam vs. SGD.

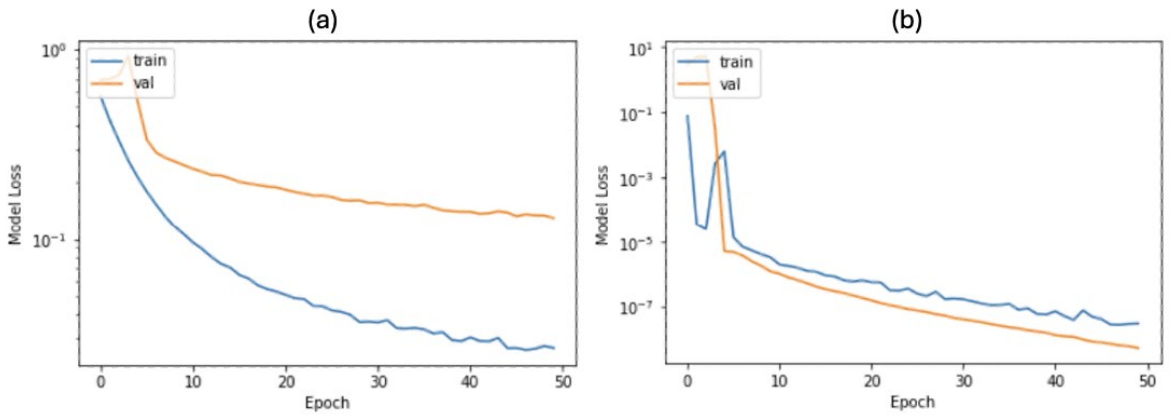


Figure 56. Performance in optimising U-Net architecture over the course of 50 training epochs and 10000 simulated training examples using SGD (a) and Adam (b). Both optimisers used the learning rate of 5e-5, tapering off exponentially with the progress of the training.

Figure 56 shows that Adam optimiser vastly outperforms SGD, with the final training loss in the order of 10^{-7}, compared to SGD's 10^{-2}. As the loss reduction is the primary function of the optimiser, Adam is the clear selection going forward. It is worth noting that validation loss is consistently lower than training loss. While seemingly counterintuitive, the cause lies in the different model behaviour in training and in validation. To prevent overfitting to the training set, dropout (randomly deactivating neurons) is used. Thus, in training the performance of the network is artificially lowered, with the same not being the case in validation.

Adam has three main tuneable parameters: learning rate, β_1 and β_2. In the original paper the learning rate is set to 0.01, β_1 to 0.9 and β_2 to 0.999. The latter two parameters are generally not modified unless there is a strong application-specific reason why they need to be changed. The application in this work is standard in the terms of the optimisation process and thus the β parameters are not modified. Adam is an adaptive algorithm, tuning the learning rate internally (see section 2.2.3.6.2 for details), thus the change in the parameter is less impactful than in simpler algorithms, but it still warrants optimisation in most cases. The proposed learning rates for this application are $5 * 10^{-6}$, $5 * 10^{-5}$, $5 * 10^{-4}$ and $5 * 10^{-3}$.

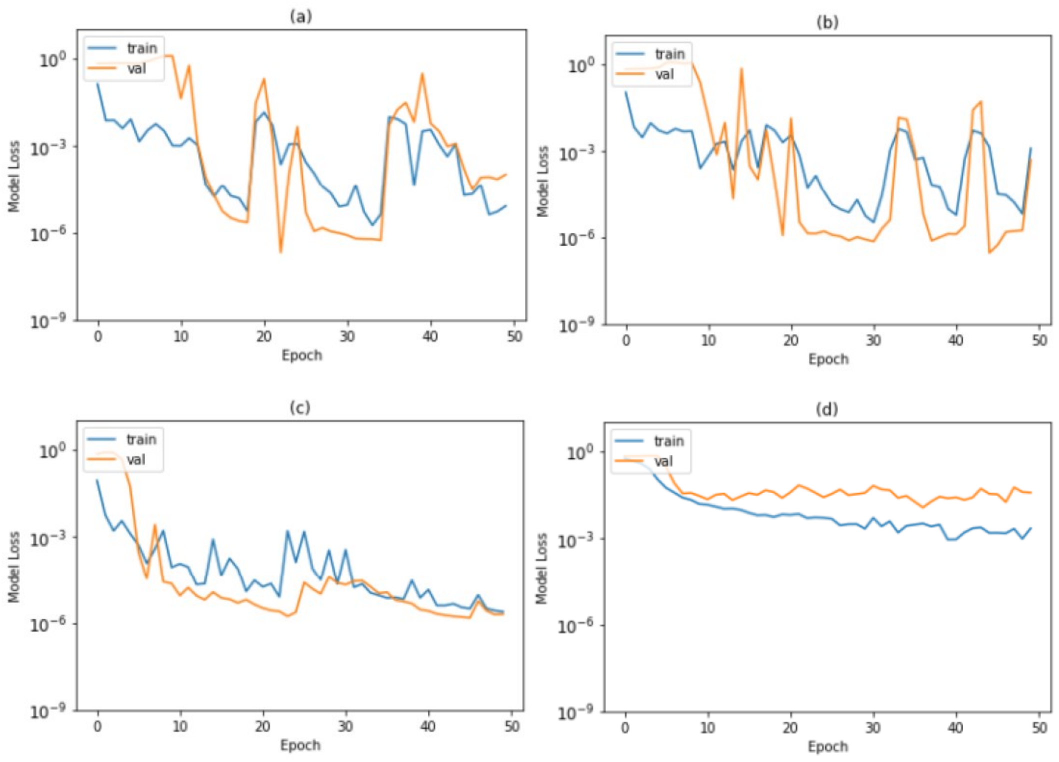


Figure 57. Performance in optimising U-Net architecture over the course of 50 training epochs and 10000 simulated training examples using Adam Optimiser with learning rate of 5e-3 (a) 5e-4 (b), 5e-5 (c), and 5e-6 (d). Each of the graphs is the training history of a single initialisation of a neural network.

Figure 57 presents the results of the optimisation of a neural network using a range of learning rates. The learning rate of $5 * 10^{-5}$ (c) is clearly the best performing one, achieving the loss value 3 orders of magnitude lower than the second-best option, $5 * 10^{-6}$ (d). The higher learning rates notably reach low loss values quicker, but the values are unstable, which points to then exiting the loss landscape minima. This is not suitable for the training, as it prevents consistent improvement in the performance of the neural network.

Finally, the learning rate can either be held steady over the training process or decreased according to an exponential curve. The rationale for such a design choice is the need for quick training at the beginning, when the parameters of the model are far from the global minimum and lower when making minor adjustments close to the final values.

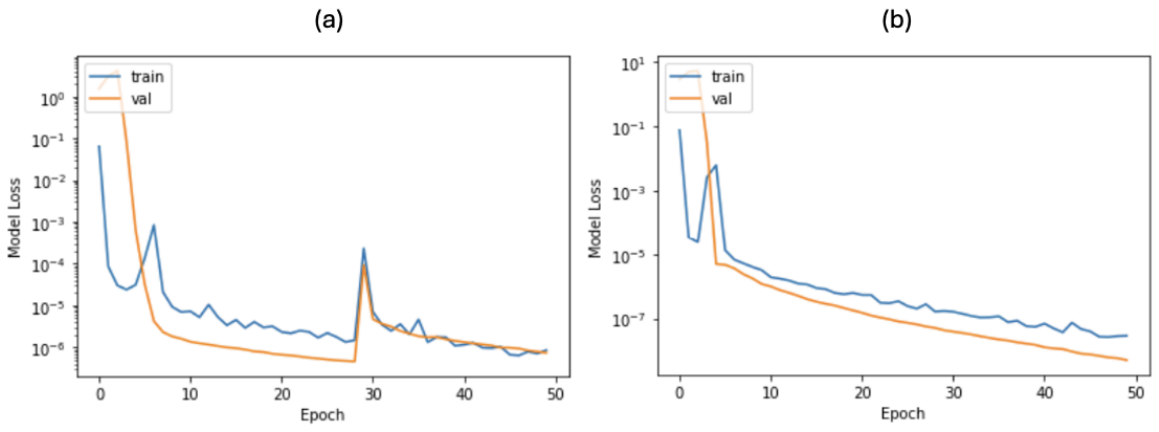


Figure 58. Performance in optimising U-Net architecture over the course of 50 training epochs and 10000 simulated training examples using Adam. Optimiser uses the learning rate of 5e-5 (a), or starting at 5e-5 and tapering off exponentially with the progress of the training (b).

Figure 58 presents the difference between the constant learning rate (a) and adaptive (b). The gradual lowering of the learning rate allows the optimiser to smooth out the training and avoid suboptimal weight updates (visible with constant learning rate around epoch 30). The final loss is, similarly, lower for the adaptive learning rate, making it a clearly superior choice. The final parameters of the training are presented in Table 3.

Table 3. ML Training Parameters.

Number of epochs on simulation set	30
Number of epochs on real set	30
Learning rate	5e-5 initially, exponentially decaying
Optimizer	Adam [67]
Loss function	Binary Cross-entropy

4.4 Transfer Learning

The transfer learning approach is the major contribution of this work to the body of ML research in the context of GWT. The rationale and process for designing transfer learning has been introduced previously, thus this section describes the specific implementation used in this work.

The transfer learning of an ML model is split into two parts – the pre-training and the fine tuning. The goal of the process is the optimisation of the performance on the target dataset, which in the case of GWT consists of in-operation inspection data. The model needs to be pre-trained on as large a dataset as possible, thus the simulation dataset is a reasonable candidate.

The approach in this work involves the training of the machine learning model first on the simulated data and then fine-tuning on a subset of real inspection data. Traditionally, just the classification head of the architecture would be re-trained on the real data, however experimental process has shown that omitting the re-training of the classification layers in favour of fine-tuning the whole model leads to better performance on the real data.

This approach is followed in every experiment, with the earliest work comparing the performance of the ML algorithm trained purely on simulations or purely on the experimental data with the transfer learning approach. At a final stage of the work a quantification of the impact of transfer learning is undertaken, with a selection of architectures trained on 16 different combinations of real and simulated dataset sizes, as detailed in Table 4. Training for each combination of dataset sizes is repeated 5 times on different randomly selected training sets in the attempt to incorporate the effects of random factors affecting the training procedure, such as the random initialisation of all parameters. At each training instance, the validation sets (separate for simulated and real datasets) are drawn at random as 20% of their respective training sets, and the model that is retained at the end of each training stage (on simulated data first, on real data then) is that giving the lowest validation loss.

Table 4. Combinations of real (nReal) and simulated (nSims) training dataset sizes used in the transfer learning impact study.

ID	nReal	nSim	ID	nReal	nSim	ID	nReal	nSim	ID	nReal	nSim
1	0	0	5	270	0	9	810	0	13	1800	0
2	0	900	6	270	900	10	810	900	14	1800	900
3	0	2700	7	270	2700	11	810	2700	15	1800	2700
4	0	10000	8	270	10000	12	810	10000	16	1800	10000

The implementation of transfer learning in this work is based on merging the two training processes together, with the learning rate scheduled to fall exponentially, automatically providing the low learning rate for fine tuning. This also allows for the merging of the training statistics from the two processes.

5 Machine Learning Performance

This chapter presents the performance of the machine learning algorithms trained and tested over the course of the research contributing to this thesis. It starts by investigating the performance of machine learning on the simulated dataset. This fully controlled and rich environment is used to investigate the impact of design decisions:

- The dataset size.
- The use of synthetically focused data.
- T(0, 1) and F(1, 2) compared to just T(0, 1).
- Both enveloped and RF data compared to either one individually.

The second part involves the domain shift to real data and presents the performance of the transfer learning approach using either synthetically focused or A-scan type data. Finally, the third part introduces the shift from defect indication detection to feature detection. As the ML performance on this question shows the promise for industrial usefulness, it is investigated further to quantify the impact of the amount of the data on the performance as well as to select the best performing machine learning architecture.

5.1 Performance on Simulations

Having identified suitable candidate architectures and the learning process, the first step in the project is the validation of machine learning as an approach to the problem. The similarity of the ultrasonic testing data to known existing ML applications (seismology, speech recognition) is a cause for optimism, but validation on guided wave data is needed.

Furthermore, provided a good performance using the baseline model is proven, simulations as the training and testing datasets provide a flexible canvas for testing the variations to the ML approach. This section will investigate three such modifications – using just T(0, 1) mode instead of both T(0, 1) and F(1, 2), using just the RF or envelope data instead of both, and using the data that has gone through the imaging process. While the good results on simulations are not necessarily directly transferrable to the real datasets, such experiments allow the identification of promising research directions.

5.1.1 Impact of the Amount of Data

The first question that can be answered using the simulated dataset is the impact of the size of the dataset on the performance of the ML model. It is expected that the performance will increase with the size of the training dataset, but the more interesting question is the specific number of datapoints that saturate the performance of the model.

The point at which the addition of simulated data points no longer improves the performance of the model on the testing dataset corresponds to the model extracting all the knowledge from the training dataset (i.e., perfectly modelling the underlying distribution) or alternatively that it has reached the limit of complexity it can accommodate. The second explanation is deemed unlikely in this case, as the models consistently perform at up to 100% accuracy on simulations. As the distributions are specific to the simulation parameters, the value observed should not be used as a hard limit for the use in other work. It can however be used as a starting point for the development of ML algorithms for other NDT applications. This knowledge is important, as the capacity to generate a large, simulated dataset varies widely between the NDT modalities and organisations.

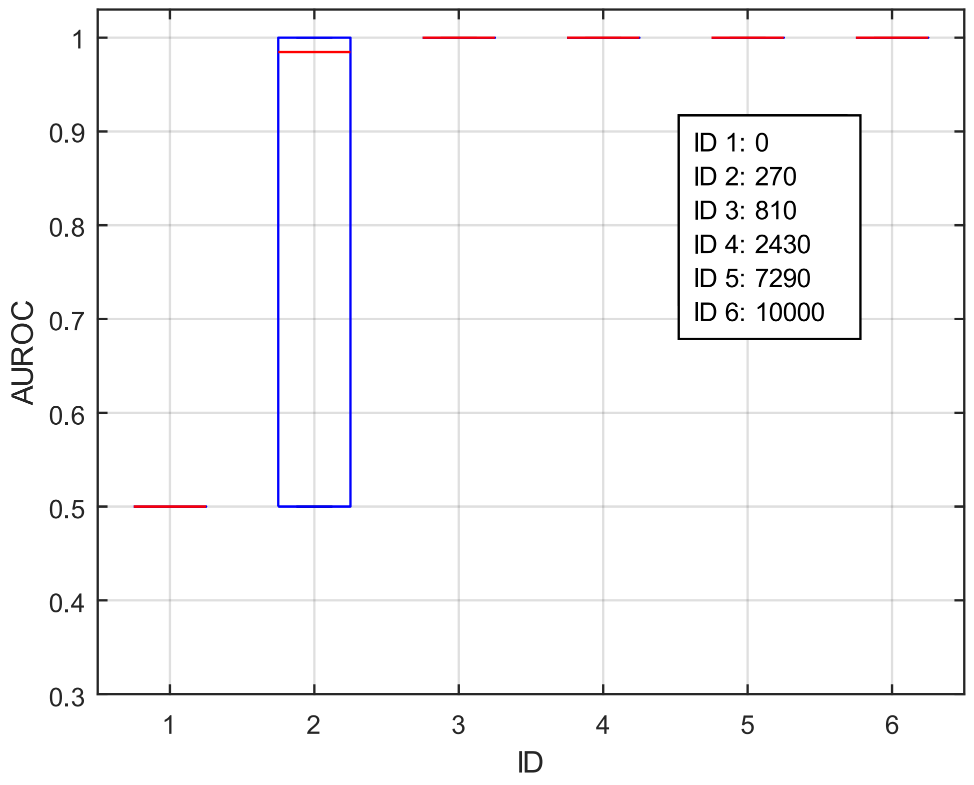


Figure 59. AUROC of ML models trained on various sizes of simulated datasets. The size of the dataset increases with ID. The red line is the median and the box denotes the top and bottom 75th percentile. Each neural network is initialised 5 times with varying random seed to draw the statistical conclusions.

Figure 59 presents the impact of the number of training examples on the performance on the testing dataset. The first characteristic clearly visible in the graph is the fact that all the models trained on 810 examples and above have performed at 1.0 AUROC – corresponding to not misclassifying a single test example by the end of the training. This points mostly to the fact that the problem of automated GWT classification is simple if applied to simulated data. The second conclusion that can be drawn is that although extremely few training examples are needed to reach very high performance, there is an appreciable difference between using 270 and 810 training examples, with the latter needed to confidently reach high performance on this problem. Thus, it can be concluded that for this type of problem a training set of 1500 is large enough. This is the training set size which is used in the experiments on the simulated dataset.

5.1.2 Impact of the Number of Modes

Both torsional and flexural modes are used in assessing the GWT data for classifying the signal as either corresponding to a feature or not. However, the torsional mode is significantly easier to extract, and it is not possible for a feature to have no torsional mode reflection. Thus, there is value in checking whether there is a difference in the detection capacity when using both the torsional and flexural modes vs. just the torsional.

This experiment is performed by training the model on a large, simulated dataset (1500 samples) and assessing the validation performance. As no changes to the architecture are proposed and the intended training and testing datasets are extracted from the same distribution, there is little need for an additional testing set. Furthermore, using the validation allows for an observation of the evolution of the performance in the training process.

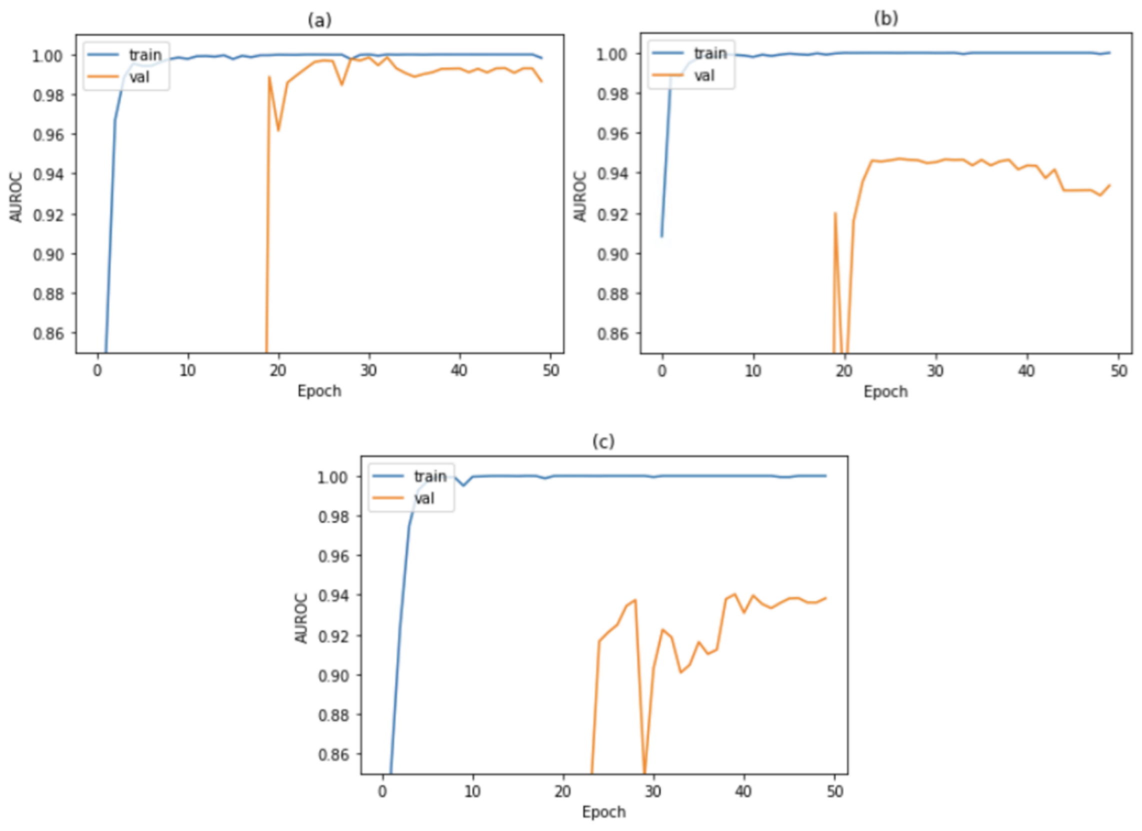


Figure 60. Comparison of the learning capability of a U-Net trained on 1500 examples using T(0, 1) and F(1, 2) (a), using just T(0, 1) (b), and using just F(1, 2 (c)).

Figure 60 illustrates the difference in the neural network learning capability (as measured by AUROC). The performance of the neural network in this chapter uses AUROC instead of the previous metric of model loss, as it is more relevant to the real-life performance of the model. depending on the modes available in the training data. Just using the torsional mode makes the training process less stable, as evidenced by the more jagged appearance of the training loss curve. Secondly, the performance on the unseen validation dataset plateaus very rapidly, pointing to the network lack of ability to generalise well based on the sparser data. It is therefore concluded that using both the axisymmetric and non-axisymmetric modes is necessary for maximising the performance of the ML model.

5.1.3 Impact of Enveloped and RF Data

The inspector sees just the enveloped data when classifying the results of a test, which is mainly due to the more readable nature of this presentation of the data, but the secondary rationale is that it is the absolute amplitude of the symmetric and antisymmetric reflection that characterise the reflector, not the underlying carrier wave signal characteristics. It is, however, true that the raw RF data contains all the information from the envelope and the underlying carrier wave data. As a result, training the model on both forms of data is in effect drawing the attention of the neural network to the amplitude, while also providing the underlying data.

The neural network being able to extract features which may be better suited to the characterisation of the traces is the motivation for training it solely on the RF data. If the amplitude turns out to be the most important feature, the network should be able to extract it without intervention. On the other hand, using just the envelope may potentially remove much of the irrelevant information from the training data, allowing the model to focus on the feature-pristine detection task rather than sifting through the noise. This may be especially impactful with the shift to the real dataset, as the impact of noise is magnified as the dataset gets smaller.

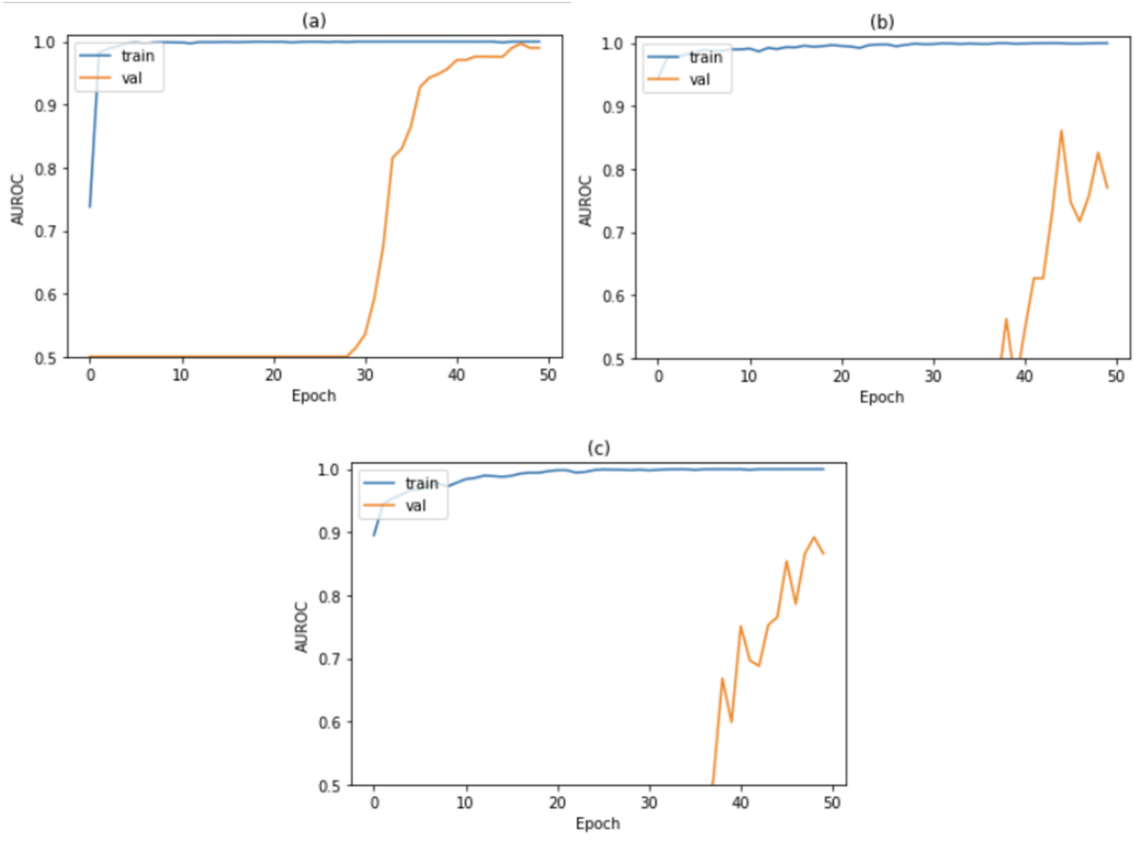


Figure 61. Comparison of the learning capability of a U-Net trained on 1500 examples using both RF and enveloped data (a), only enveloped data (b), and only RF data (c).

Figure 61 shows the differences in the learning process of an ML model trained on either RF and enveloped data, just the envelope and just the RF. From the perspective of both the smoothness of the training process and the learning speed, using both types of data motivates the decision to use both types of data in the rest of this work.

5.1.4 Impact of Imaging

The final experiment on simulations involves the most complex processing method employed in this work – full synthetic focusing using the improved CSM method described in the previous chapter. The motivation, similarly, to the previous experiment, comes from the human inspection analysis. There, the unrolled pipe display – an image is the main tool the inspectors are using for the identification of features.

The downside of directly using the images is their size – the training time of an ML model is proportional to the input data size. Additionally, imaging, like enveloping, is a processing method, thus inherently removing some information and striping the ML algorithm of some of its ability to extract features. On the other hand, images are generated using richer data than the A-scan, thus potentially allowing the algorithm to use otherwise inaccessible data. In the case of images, it is crucial to note that the comparison is not entirely fair, as the change in the dimensions of the input data necessitates the change to the size of the layers, making the neural network application to the imaged data more complex when compared to the A-scans. Nonetheless, the comparison is important, as significantly better performance on the imaged data would indicate that the C-scan data provides more useful information.

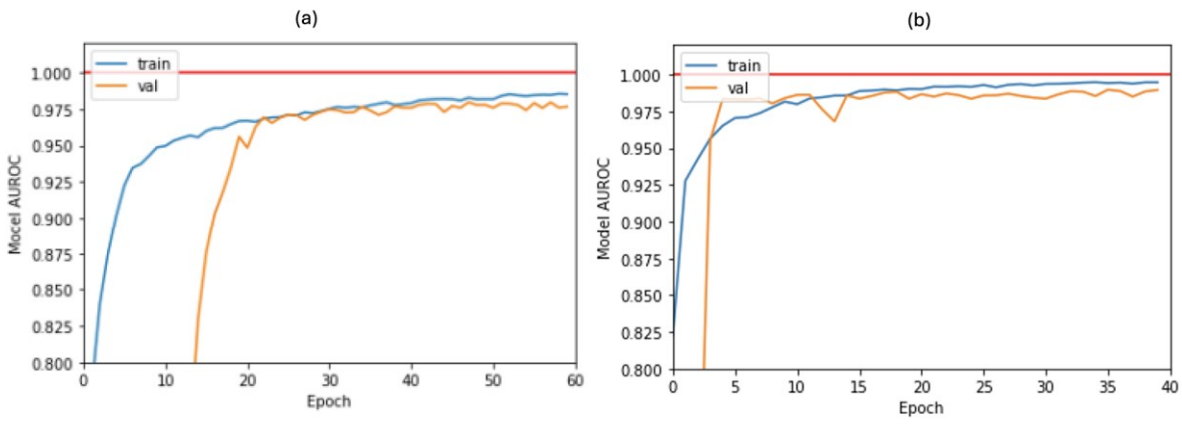


Figure 62. Comparison of the performance of U-Net architecture on C-Scan type data (a) and A-scan type data (b). Both models are trained on 10000 samples and validated on 2000. The AUROC target of 1 is marked with a red line.

Figure 62 presents the performance of the two input data formats in the terms of AUROC. Images are slower to learn and the performance on the validation set plateaus at a slightly lower level than A-scans. These results may be caused by the lower quality of the imaging than used in the industry-standard software, but as the development of the imaging algorithms is not the goal of this project, the proposal to use synthetically focused ultrasonic data to train the ML models is rejected due to its slightly lower performance compared to A-scans, as well as the readily available experimental A-scan datasets.

5.2 Performance on Real Dataset

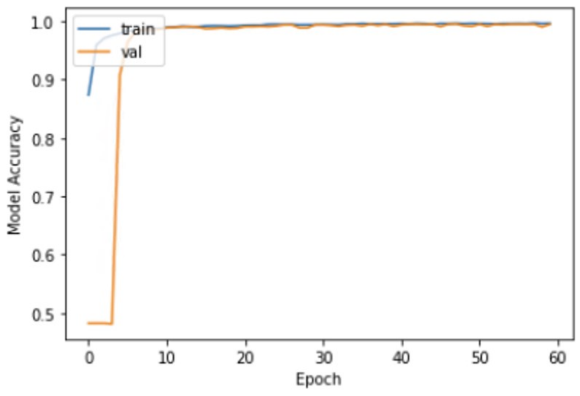
Having proven the performance of the ML approach on the simulated data, the logical step is to shift to the same question (defect indication detection) on the real data. This section presents the results of ML models on the real data, first attempting to solve the question of defect indication detection and later shifting to the more general problem of feature detection.

5.2.1 Impact of Transfer Learning

The first question to be answered is whether the simulated data does improve the performance of the ML algorithms when they are applied to the real datasets. To answer it, this work proposes an experiment of training the same U-net architecture using three approaches:

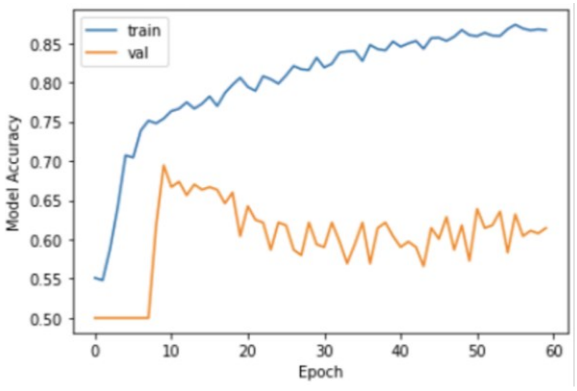
- on the simulated dataset (using the Simulated Pristine and Simulated Defect datasets)
- on the real dataset (using Real Pristine and Real Defect (augmented) datasets)
- using an introduced transfer learning approach and all the mentioned datasets

The testing set is to be drawn from the real dataset.



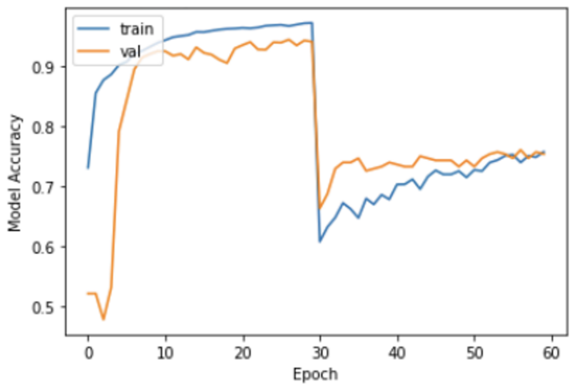
	Acc	AUROC	AUPRC
def	61.81%	0.6278	0.6017
prist	50.69%	0.5142	0.5413

Figure 63. Accuracy on testing and validation datasets for a U-net trained and validated on the simulated dataset (left). Performance of the trained model on the real testing dataset, split into pristine and defective in the terms of accuracy, area under ROC and area under PRC (right).



	Acc	AUROC	AUPRC
def	50.69%	0.5655	0.5905
prist	87.50%	0.8837	0.8232

Figure 64. Accuracy on testing (real) and validation(real) datasets for a U-net trained on the real dataset (left). Performance of the trained model on the real testing dataset, split into pristine and defective in the terms of accuracy, area under ROC and area under PRC (right).



	Acc	AUROC	AUPRC
def	79.17%	0.8896	0.8991
prist	68.06%	0.6769	0.657

Figure 65. Accuracy on testing and validation datasets for a U-net trained and validated on the simulated dataset in epochs 1-30 and fine-tuned and validated on the real dataset in epochs 31-60 (left). Performance of the trained model on the real testing dataset, split into pristine and defective in the terms of accuracy, area under ROC and area under PRC (right).

Figure 63, Figure 64, and Figure 65 show the results of the experiments. It is important to note that the graphs show the accuracy of models, and thus are vulnerable to the class imbalance problem in the real data. The accuracy is used in these figures as they are generated using an earlier version of the software and solve a defect detection rather than feature detection task, thus their values are not directly comparable with the other figures in this section. The graphs show the accuracy on the validation and training datasets. It is important to note that those datasets vary, as described in the relevant captions. The visualisations do underline, however, the different characteristics of the three training processes. Training on the simulated data quickly reaches extremely good accuracy and exhibits no difference between the performance on the training and validation sets. However, what is clear from the performance on the testing dataset is that learning on a purely simulated dataset does not translate to performance on the real data, with the accuracies just slightly better than random. Training on the real data looks promising in the first 10 epochs, with the accuracy rising both in the training and validation datasets. From this point on, however, the deficiency of a small training dataset makes itself known by strongly overfitting the model. Furthermore, the vulnerability of the model to the imbalance in the dataset is clear from the performance on the testing dataset – the model classifies most of the pristine traces correctly, but its performance on the defect indications is worse even than that of the model trained on the simulations. This is concerning, as the defect indication detection rate is more important than identification of pristine samples in the context of safety. Finally, in Figure 65 the proposed two-stage training shows the initial performance rise in line with the simulations until epoch 30, at which point the training switches to the real data. When training on the real data, the validation performance improves in the first couple of epochs, very quickly plateauing. In comparison, the training performance keeps improving over the length of the training. Most importantly, the trained model has the defect indication detection accuracy of 79.17%. While this result is significantly better than any other training process, giving hope for the transfer learning approach, the overall accuracy of around 70% is significantly below the performance necessary for any practical use. Even if 80% was to be considered a good enough performance in the terms of defect indication detection, the corresponding accuracy on the pristine samples is 68%. This indicates a false positive rate of 32%. Considering the same format of data as used in this work, a single sample for classification corresponds to a 1 m span of pipe. A 32% false positive rate in these conditions would mean that one in every three metres is flagged as containing a defect indication. Each of the flags would need to be manually reviewed by the inspector, effectively resulting in quite insignificant workload reduction. This conclusion leads to the next stage of this work.

5.3 Performance on Real Features

The main cause of the low performance of machine learning on a real defective dataset is identified as the very small number of datapoints in the positive class of the real training dataset. Therefore, it is proposed that a richer dataset including all features is used as the positive class in the training process, see Table 1 for reference. It is theorised that good performance on this dataset can be used in the industrial context, but also as a proof of concept showing the number of positive real samples necessary to successfully train an ML model using the transfer learning approach.

5.3.1 Industrial Use Case

While defect indication detection has a use case that needs no explanation, the shift to detecting features requires some justification. Testing of pipelines, after all, is not performed so the welds and supports are identified. To understand the usefulness of the feature detection capability, it is crucial to remember the intended place of ML in GWT – the automatically identified positive flags are meant to be passed to the inspector, who makes the final call and classification. As many of the features (bends, welds, supports, flanges, buried sections etc.) are designed into the pipeline, they can be quickly rejected by the inspector making final calls, leaving only the more suspicious signals. Furthermore, providing a method to automatically flag the features, including defect indications, is a benchmark for the guided wave technology, as the usage of this inspection process provides a known probability of detection and probability of false alarm regardless of the inspector skill.

5.4 Architecture comparison

With the shift to the training on a larger real dataset allowing the models to perform at high enough accuracies that they could be considered candidates for industrial use, it has become paramount to compare the various ML architectures. The candidate architectures have been introduced earlier in this chapter:

- MLP being the simplest architecture which could be considered deep learning
- VGG-Net, a typical example of a convolutional neural network
- U-Net, a commonly used modern CNN

To assess the performance of ML approaches they would ideally need to be compared to the current state of the art. This proves difficult, as at present there is no standard procedure to automate the detection of pipe features (PFs) in guided wave-based inspection of pipes; instead, this relies on the experience of trained inspectors that would assess the absolute and relative enveloped amplitudes of both T(0, 1) and F(1, 2) modes. In particular, typically PFs are characterised either by a high T(0, 1) component (for axisymmetric PFs) or a high F(1, 2) to T(0, 1) ratio (for non-axisymmetric PFs). An attempt is made to mimic such a procedure, in order to form a baseline performance against which to assess the ML algorithms developed in this work. To address this, thresholding on three different enveloped signal amplitudes is considered, namely on (*Th1*) T(0, 1) only, (*Th2*) the F(1, 2) to T(0, 1) ratio, and (*Th3*) on the following linear combination of T(0, 1) and F(1, 2): *0.7 x T(0, 1) + 0.3 x (F(1, 2)/T(0,1))*. The parameters of this linear combination are based on an arbitrary decision. An attempt to optimise the formula is made, but the optimised version is extremely close to pure T(0, 1), with the parameter over 0.99. As a result it is decided that a suboptimal, but deemed more representative example is to be used. Note that such thresholding methods are applied to the same ~1-metre-long segments in which the experimental traces are split, as explained in the previous section, though in this case only the enveloped, non-normalised, DAC-corrected traces are used.

An example of application of the proposed thresholding approaches is given with the aid of Figure 66, which shows a fabricated set of T(0, 1) and F(1, 2) traces including three PFs at the positions indicated by the red dashed vertical lines. The leftmost PF has a high T(0, 1) component, the middle one has both T(0, 1) and F(1, 2) components at rather high levels, while the third PF has a weaker T(0, 1) reflection, though with a F(1, 2) component nearly at the same level as T(0, 1). For example, when the threshold is set to 0.6 (shown with a dotted black horizontal line in figure), the use of T(0, 1) alone (i.e., of *Th1*) would not detect the small rightmost PF, while the use of the F(1, 2) to T(0, 1) ratio (i.e., of *Th2*) would be insensitive to the large symmetric leftmost PF. The proposed linear combination of T(0, 1) and F(1, 2) (i.e., of *Th3*) shown in green would instead detect all three PFs at the considered threshold. When the thresholding approach is practically tested using Python scikit-learn library, the threshold is set automatically to maximize the classifier's accuracy.

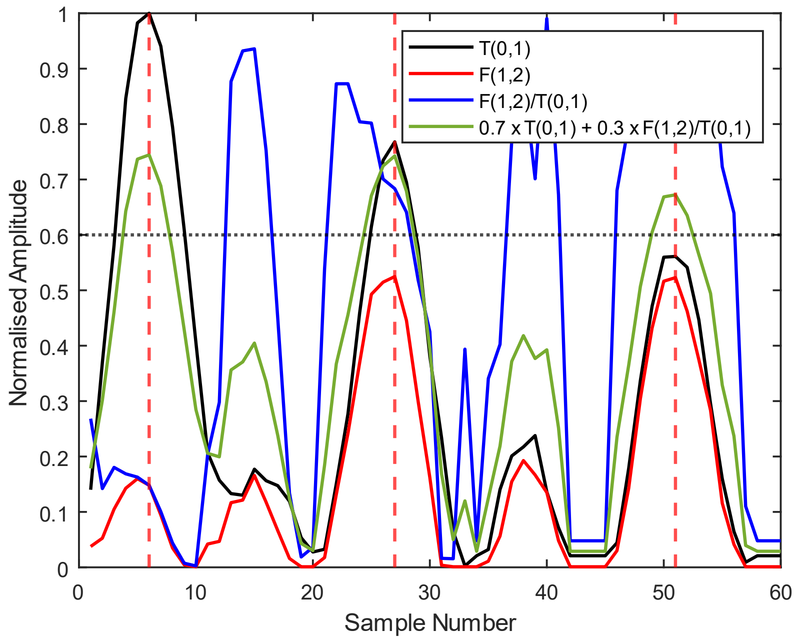


Figure 66. Purely illustrative example of application of thresholding on a fabricated dataset of T(0, 1) and F(1, 2) traces. The locations of three features are indicated with red dashed lines and a tentative threshold is shown as a black dotted line. Thresholding is performed independently on the enveloped signals shown as black, blue and green solid lines. A successful detection occurs when any of these signals exceeds the threshold in the vicinity of a feature. Crossing the threshold away from features is a false positive, not crossing it in the vicinity is a false negative.

Table 5. Performance of the three proposed thresholding approaches.

	Th1	Th2	Th3
AUROC	0.9936	0.2747	0.9629
FPR@1TPR	0.4131	0.9972	0.4843

The performance offered by the three thresholding approaches and by the three ML architectures when trained with various sizes of simulated and real data is discussed below. Table 5 gives AUROC and FPR@1TPR when thresholding is applied to all available 2400 experimental data samples. The best results overall are obtained when using *Th1*, i.e., when thresholding on T(0, 1) only. However, despite its relatively high AUROC sized at 0.9912, the FPR@1TPR at 0.4246 essentially indicates that there exist some PFs giving low T(0, 1) reflections which can only be flagged by thresholds that would also call an excessively large number of false positives, i.e., almost one false positive for every two negative samples. The extremely poor performance of *Th2* shows that the sole use of F(1, 2) cannot reliably discriminate between positive and negative samples. This is expected, since all axisymmetric PFs such as welds do not produce significant non-axisymmetric reflections. Finally, the attempt to combine the amplitudes of T(0, 1) and F(1, 2) into a single parameter via *Th3* did not yield improvements over the sole use of T(0, 1), as both AUROC and FPR@1TPR for *Th3* are slightly worse than those of *Th1*. These results essentially suggest that the experience of trained operators on the analysis of signals acquired by GWT of pipes cannot be replaced by an approach that only makes use of the local amplitudes of T(0, 1) and F(1, 2) time-traces, and that their shapes should also be considered.

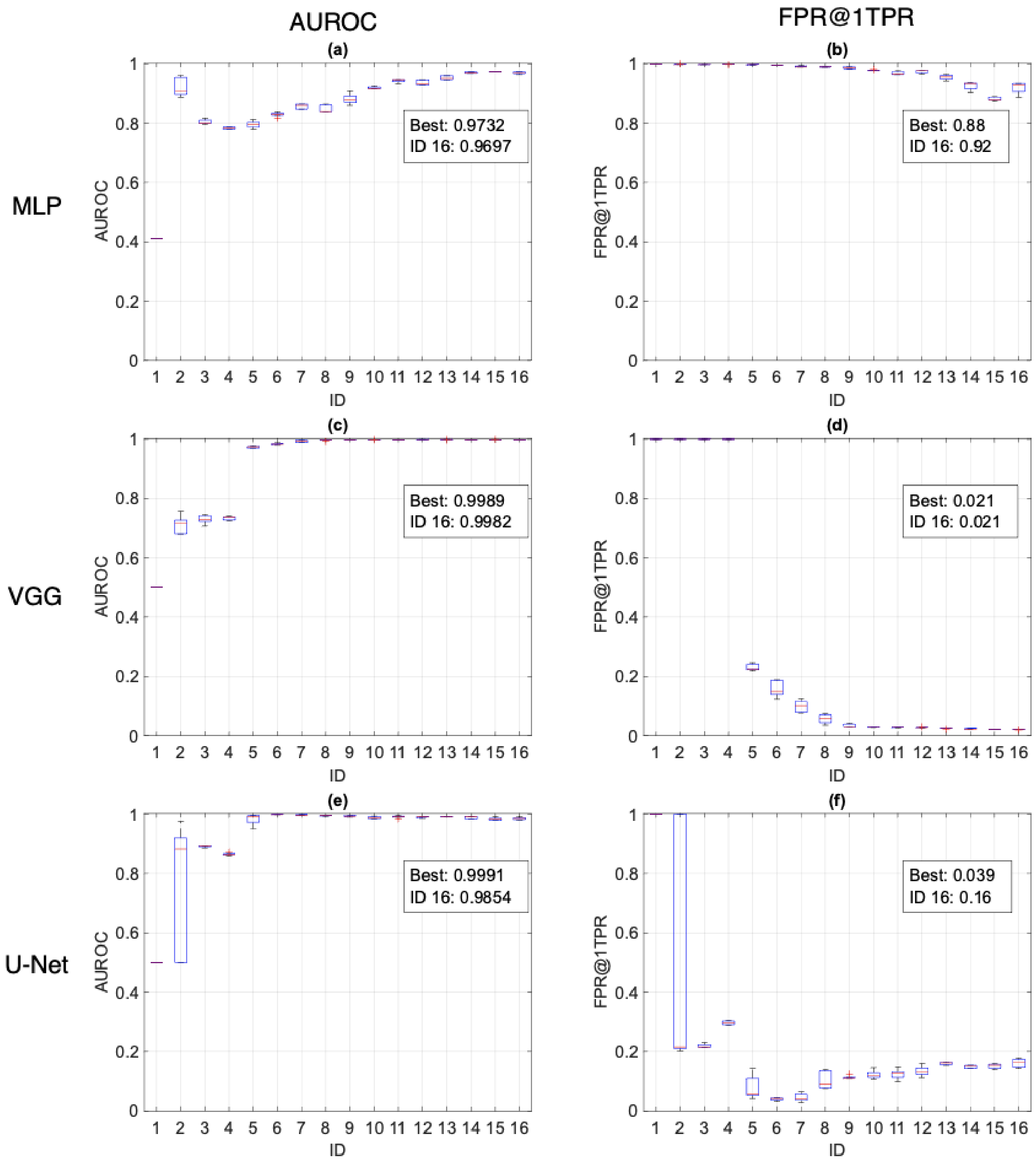


Figure 67. AUROC of MLP (a), FPR@1TPR of MLP (b), AUROC of VGG-Net (c), FPR@1TPR of VGG-Net (d), AUROC of U-Net (e), FPR@1TPR of U-Net (f). On each box, the central mark indicates the median, and the bottom and top edges of the box indicate the 25th and 75th percentiles, respectively. The whiskers extend to the most extreme data points not considered outliers, and the outliers are plotted individually using the '+' marker symbol.

Figure 67 gives the performances of the three ML architectures evaluated on the same, randomly selected testing set of 480 experimental samples and across the 16 combinations of training and testing set sizes (IDs), as given in Table 4. For each ID, the metrics obtained from the five separate training instances are summarized in a boxplot where the central mark indicates the median, while the bottom and top edges indicate the 25th and 75th percentiles, respectively. Figure 67 (a, b) show that MLP does not possess the required modelling capabilities to reliably characterise true and negative samples, and despite the expected general improvement as more simulated and real input data are employed for its training, its best overall performance remains significantly inferior to that of *Th1* in terms of both AUROC and FPR@1TPR. It is worth emphasizing here that the input to MLP and the other ML architectures significantly differ to that of thresholding, as in the latter case samples are not normalised.

Figure 67 (c, d) shows that when VGG-Net is solely trained on simulated data (i.e., IDs 1 to 4), the performance on experimental data is extremely poor. When, instead, a relatively low number of 270 real samples are used for fine-tuning (IDs 5 to 8), both AUROC and FPR@1TPR gradually improve as the amount of simulated data is increased. Then, once a more substantial number of real samples are available, the benefit of pre-training the model with simulated data starts to fade. In fact, the performance at IDs 13 to 16, where 1800 real samples are used, is essentially flat as the number of simulated samples is increased from 0 to 10000. Such performance is significantly superior to that offered by thresholding on T(0, 1) (i.e., *Th1*), with AUROC surpassing 99.8% and FPR@1TPR yielding 2.1%. In order to investigate whether this overall behaviour and quality of performance are specific to the particular testing set used to produce Figure 67, the same procedure is repeated in five more instances by randomly selecting five additional testing sets, and the resulting FPR@1TPR are displayed in the left column of Figure 68. All plots confirm the conclusions drawn above. It is also worth noting that for each of the six testing sets depicted in Figure 67 (c, d) and in the left column of Figure 68, when 1800 real samples are used for training (IDs 13 to 16) the use of simulated data has a negligible effect, whereby when only up to 810 real samples are available, the performance of the classifier can be improved by the addition of simulated samples. This is particularly evident in Figure 68 (c). Unsurprisingly, the plots also show that the FPR@1TPR varies as different testing sets are evaluated, and it ranges between ~1.5 and ~4% when 1800 real samples are used for training.

Finally, Figure 67(e, f) shows that the employed U-Net algorithm struggles to yield a consistent performance, which, on average, unexpectedly deteriorates as the number of real samples used for the fine-tuning of the model is increased. This is confirmed by the results shown in the right column of Figure 68, where U-Net is tested on the same five additional testing sets described above. Only in one instance, i.e., the case of Figure 68(j), the expected behaviour is obtained, and a similar performance as VGG-Net is achieved. This unpredictable behaviour of U-Net was extensively investigated as the hyperparameters of the model were sequentially varied, though the investigation remained inconclusive, and VGG-Net was chosen to carry out the study described in the next section.

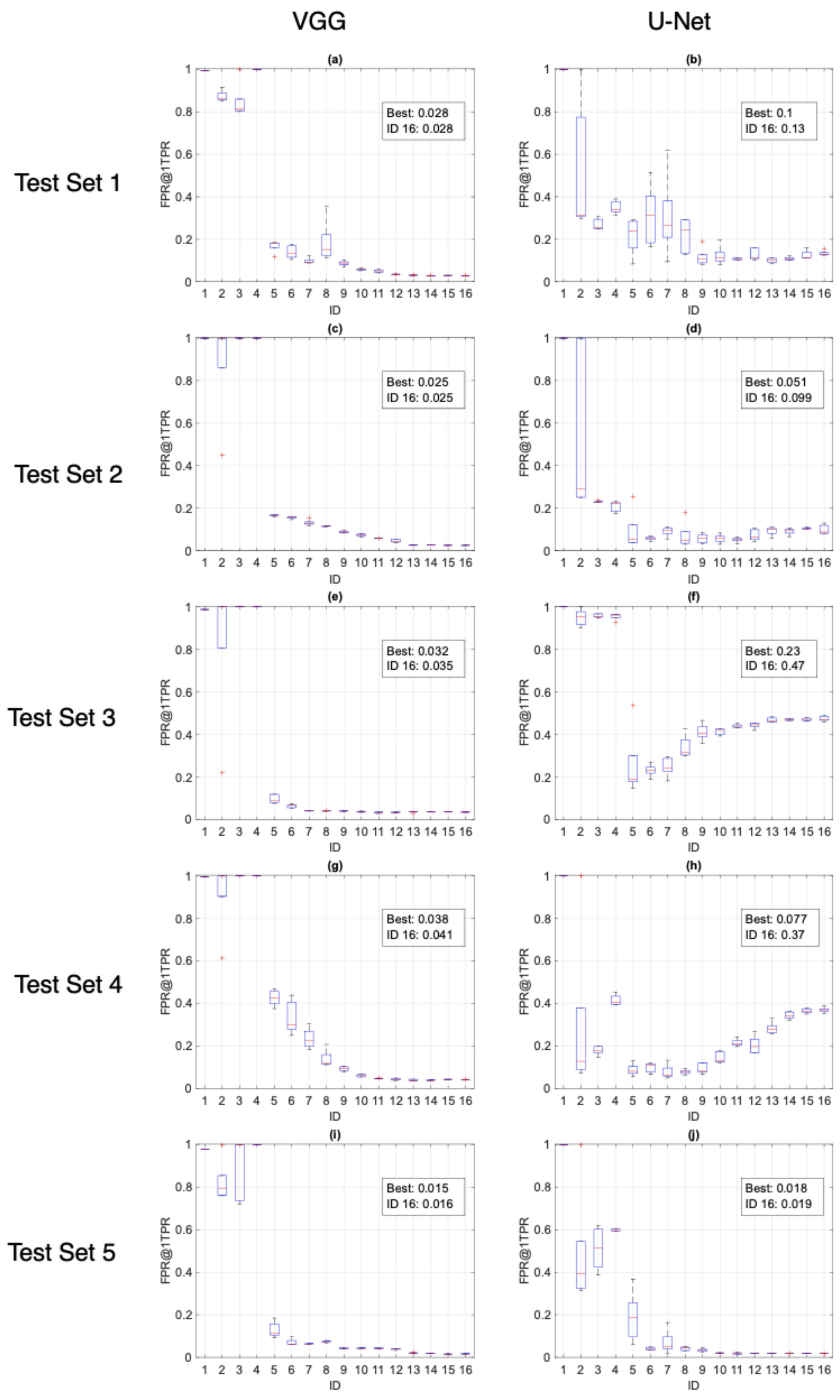


Figure 68. FPR@1TPR as VGG-Net (a, c, g, e, i) and U-Net (b, d, f, h, j) are evaluated on five different testing sets. The same testing set is used in each row of plots. Boxplots are as in Figure 67.

5.5 Investigation of Misclassified Samples

It is of practical interest to investigate which PFs are more difficult to identify correctly using the VGG-Net model trained with the largest sets of real and simulated data considered in this work (i.e., on the scenario indicated as ID=16). In order to make this study statistically more relevant, training and testing on VGG-Net at ID=16 was performed for 44 additional randomly selected testing sets. Again, for each testing set the training was repeated five times. The full database obtained by adding these additional results to those described in the previous section consists of 50 different testing sets, each including 125 PFs selected at random among the available 634, with Table 6 giving statistics on the numerosity of the various types of PFs included in each testing set. Since each testing set is evaluated by five independently trained models, this equates to 31250 real positive samples being tested. A study on the composition of this ensemble of experimental samples shows that each of the 634 PFs appears at least 20 times.

Table 6. Statistics on the number of occurrences of the different types of pipe features across all testing sets used in this study.

Feature Type	Mean	Standard Deviation	Maximum	Minimum
Weld	59.08	6.01	69	45
Support	34.28	4.83	48	24
Defect indication	11.3	3.14	19	6
Bend	8.16	1.98	12	4
Flange	6.84	2.39	13	2
Reverberation	5.8	2.17	11	2
Earth Entrance	0.94	0.91	3	0
Unknown	2.22	1.40	6	0

The rule that is used to determine the threshold against which to mark the false negatives in each of the 250 sets of results (as obtained from the 50-by-5 VGG-Net training and testing instances) is to set it to the value giving a FPR of 2% in the specific set. Following this procedure, a total of 784 positive samples (~2.5% of 31250) are misclassified as negative, with 32 out of the total 634 PFs (~5%) being misclassified at least once. The composition of this set of 32 PFs is given in Table 7, where PFs are listed in descending order according to the ratio between misclassified cases and their total number. The table shows that false echoes, defect indications and supports are the PFs that are most liable to be misclassified based on their signatures on T(0, 1) and F(1, 2) signals. This can be easily explained, as false echoes are often marked by operators based on the presence of large PF reflections in the time-traces propagating in the other direction of test, information that is not given to VGG-Net. Similarly, often supports give very weak signal reflections, and they are marked based on their physical presence as confirmed by visual inspections. Finally, reflections from very small defect indications can be barely distinguishable from background noise, which may force an operator to conservatively flag an indication despite significant uncertainty. On the other side, welds, bends, flanges and entrances into earth usually give very distinctive and significant reflections, hence the low probability of VGG-Net missing any of them.

Table 7. Number and percentage of misclassified PFs according to the specific PF type, using the VGG-Net model trained with 10000 simulated and 1800 experimental samples (i.e., the scenario indicated as ID=16).

	Cases misclassified at least once	Total number of cases (misclassified and not)	Percentage of misclassified cases (%)
False Echo	4	28	14.3
Defect indication	7	58	12.1
Support	17	164	10.4
Weld	4	293	1.4
Bend	0	41	0
Flange	0	36	0
Unidentified anomaly	0	10	0
Entrance into earth	0	4	0

With regards to the 32 PFs that have been misclassified at least once, it is of interest to investigate those with a high prevalence rate (i.e., the number of times a sample has been misclassified divided by the number of times it has been tested), since those with low prevalence can be filtered out via ensemble methods [96]. Attention here is devoted to defect indications, since they are the PFs of most concern if missed. Of the seven misclassified defect indications listed in Table 7, only two have prevalence rates exceeding 20%, and their signatures are shown in Figure 69. Interestingly, they both come from the same inspection, whose WavePro [9] trace is displayed in Figure 70. The inspection of the original trace shows that the marked position of defect indication (a) (denoted as +F6 by the operator) is slightly off and most of the major F(1, 2) signature of the defect is absent from the trace presented to the ML algorithms. Notably, when the windowing segment labelled (a) in Figure 69 is moved ~0.6m to the left, and when such better-centred version of this defect indication is fed to all VGG-Net models used in this investigation at the set thresholds corresponding to 2% FPR, a 100% true positive rate is obtained. More challenging, instead, is to find reasons for the consistent misclassification of the defect signature shown in Figure 69(b) as well as in the red box labelled (b) in Figure 70, which is characterised by relatively similar levels of T(0, 1) and F(1, 2) reflections. This highlights one of the fundamental issues plaguing virtually all ML algorithms currently used worldwide, i.e., their black-box nature, and therefore suggests focussing on ML explainability [97] for future research.

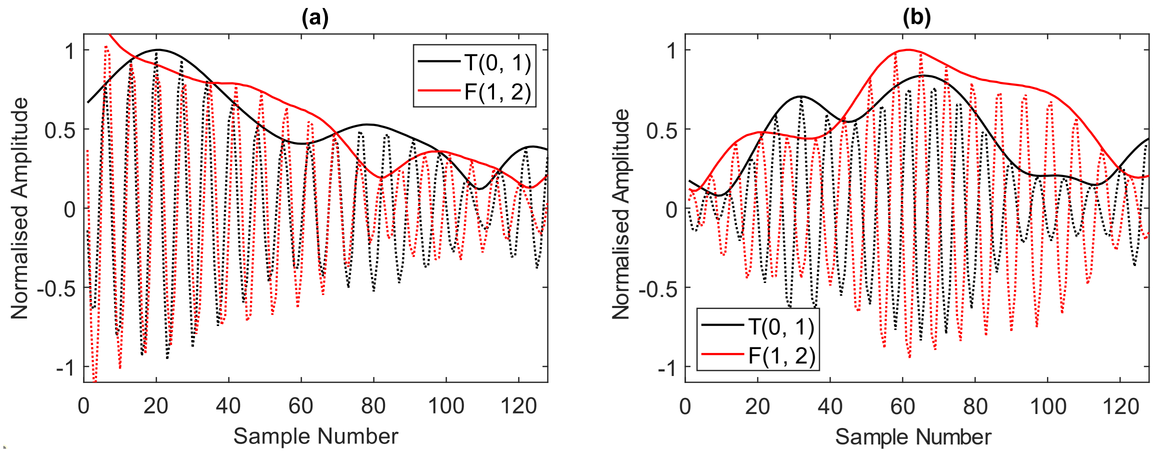


Figure 69. Samples of defect indications that are consistently misclassified as negative samples by VGG-Net (prevalence rates of 91%(a) and 98%(b)).

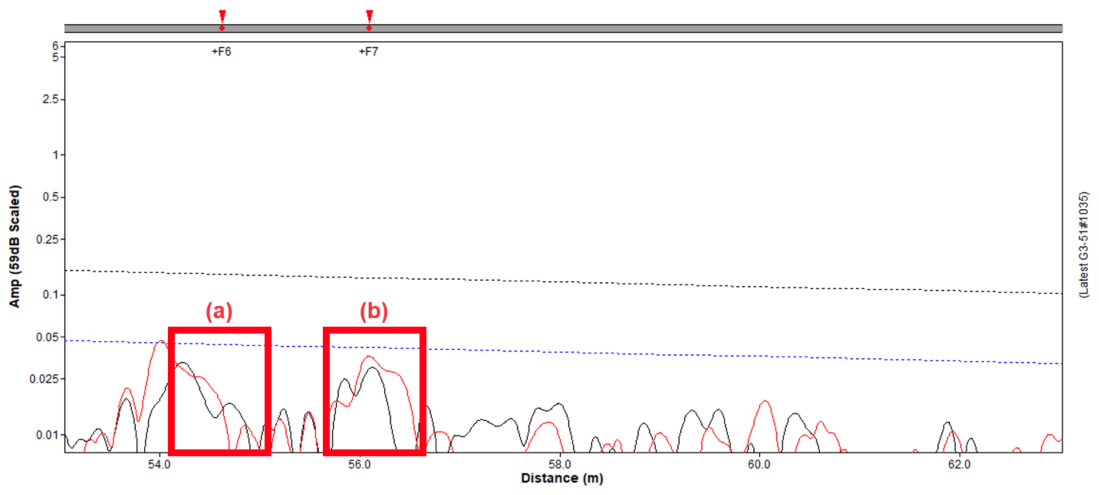


Figure 70. Portion of the original inspection trace containing the defect indications shown in Figure 69 . Defect indications are marked as +F6 and +F7 on the trace and they are located at a distance of approximately 54 and 56 m from the sensor position. The segments of trace fed to the ML algorithms are highlighted in red boxes. Reproduced from WavePro, courtesy of [9].

5.6 Machine Learning Investigation Outcome

This chapter has proposed a transfer learning framework that allows augmenting an experimental dataset collected via GWT of pipes with synthetic data produced via FEM in order to train a ML algorithm to inspect the signals and to detect reflections from pipe features. The transfer learning between synthetic and real data is achieved by first pre-training the chosen ML model on the simulations, and then fine-tuning it via additional training on the set of experimental signals. In particular, three types of ML models have been considered for the task, namely MLP, VGG-Net and U-Net, and their performances have been also compared to those given by classical thresholding approaches. Unexpectedly, VGG-Net was found to yield more consistent results than U-Net, while they both significantly outperformed MLP and thresholding. VGG-Net performance was a surprise as it is a relatively simple architecture. Furthermore, the U-Net residual connections were initially expected to simplify guided wave trace classification. Restricting the analysis to the VGG-Net results, the investigation has shown that when scarce amounts of real data are available, significant gains in the detection performance can be obtained by employing the suggested pre-training approach. In particular, the performance monotonically improves and eventually plateaus as the synthetic dataset is increased, at which point further improvements can only be obtained by enlarging the size of the experimental dataset. However, once a sufficiently large number of real data are available for training, the benefit of pre-training the model on simulations

starts to fade. This was shown to occur for this particular application once 1800 data samples (roughly equivalent to 1800 m of inspected pipes) are fed to the VGG-Net model, at which point false positive rates in the order of ~1.5 to 4% at the fixed true positive rate of 99.7% are achieved. From a practical standpoint, the adoption of this model would greatly reduce the amount of data that needs to be manually inspected by qualified operators for the classification of the detected pipe features. Future studies will focus on automatising this subsequent classification step, although it is expected that additional experimental samples would be required to better characterise the signatures of those pipe features that are less frequently found, such as flanges, bends, and, most importantly, defect indications.
A further investigation was carried out to pinpoint specific pipe features that VGG-Net would consistently miss. They included two of the 58 defect indications available in the experimental dataset, although it was later found that the position of one of the two had been misreported by the inspector in the original inspection trace. Once that defect indication position was corrected, VGG-Net was actually able to detect it. It is more concerning instead that all efforts devoted to understanding why the other defect indication was left undetected remained inconclusive. This highlights the importance of being able to explain the decisions made by any given ML model, therefore suggesting focusing on ML explainability as a further potential avenue of future research.

6 Generative Learning and Modern Machine Learning Topics

This chapter expands the application of machine learning in GWT towards emerging and exploratory research directions. It refers to the background information on the modern methods and issues in machine learning and contextualises them for NDT. The first part of the chapter introduces the Generative Adversarial Network implemented in this work. It briefly discusses the design considerations and presents the results achieved by the GAN in generating ultrasonic traces *ex nihilo*. Finally, it suggests some potential use cases for generative machine learning in NDT. The second part of this chapter focuses primarily on the questions of explainability and reliability of ML approaches and how the questions related to these areas hinder the more widespread development of ML. It concludes by proposing some directions of action that could build the trust in ML, which is critical for its adoption in safety-critical applications like NDT.

6.1 Generative Adversarial Networks

Generative adversarial networks have been introduced on a conceptual level in the theoretical background of this work, and therefore this section will focus on the finer points of the design of GANs for guided waves applications and introduce the architecture used in this work.

6.1.1 GAN Design

A design of a generative adversarial network is more complex and time-consuming than that of a typical neural network. That is mainly caused by the fact that the training process is, in effect, a competition between two networks, the generator, and the discriminator. The operation of a GAN is schematically shown in Figure 71.

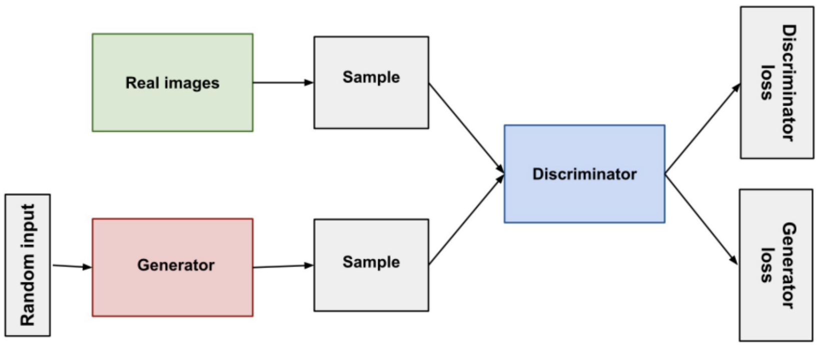


Figure 71. Diagram of Generative Adversarial Network.

From the perspective of a designer, therefore, the task is not as simple as designing a neural network that performs as well as possible. In this case it is crucial to design the generator and discriminator networks that can reach as high a level of performance as possible, while ensuring that neither of them learns too quickly, as a 'victory' of either of the networks effectively stops the learning process. This consideration can be accommodated for by implementing one of two design approaches. One is employed by large AI R&D centres, which develop very large and complex generator and discriminator networks. To offset their potential for quick learning, multi-stage training processes are implemented, where the performance of the generator and discriminator is assessed separately and based on the results the training routine is modified to ensure the networks' synchronic development. The main problem with this approach is the huge need for computational power exhibited by GANs, which is further exacerbated by the employment of large networks. The second approach, used in this work, is typical of early-stage applications and it involves using relatively small and primitive neural networks that can be trained quickly enough to enable experimentation with architectures and hyperparameters.

Figure 72 shows the architectures of the generator and the discriminator. The networks are very similar to each other, the discriminator is a standard convolutional neural network while the generator is a deconvolutional network. The input to the generator is a 19-sample long vector of random noise, used as a seed to generate realistic-looking data. The length of the vector is defined by 16+3 samples, with 16 being the arbitrary latent space dimensionality, which the GAN maps to the output domain (i.e., realistic examples). The final three positions correspond to one-hot encoded classes of sample to be generated (no defect indication, one defect indication, two defect indications). The work presented in this section was performed only on the simulated data, with a roughly 10 m long stretch of a simulated pipe used to allow GAN to model the long-range relationships and multiple features.

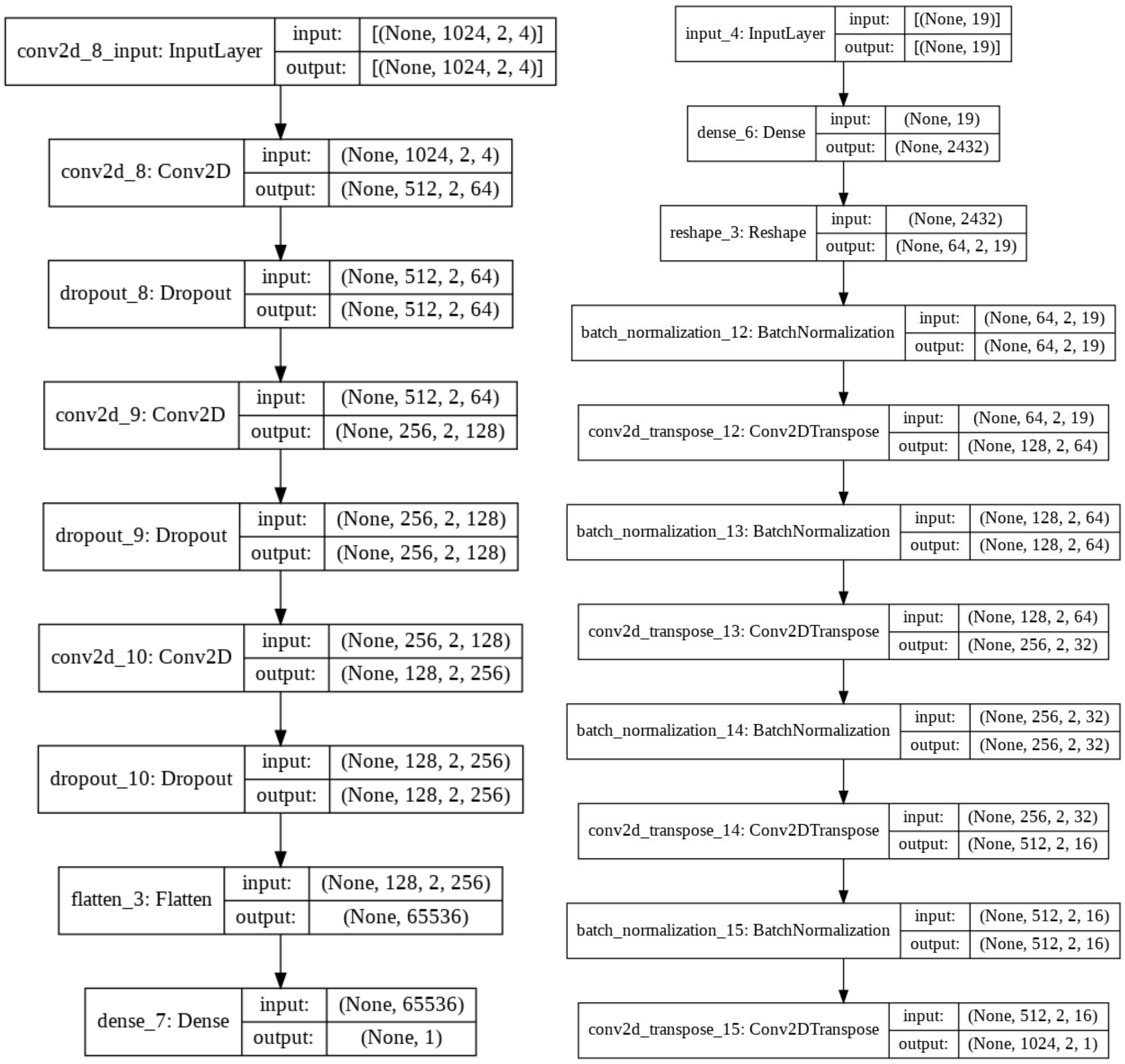

conv2d_8_input: InputLayer | input: [(None, 1024, 2, 4)] | output: [(None, 1024, 2, 4)]
conv2d_8: Conv2D | input: (None, 1024, 2, 4) | output: (None, 512, 2, 64)
dropout_8: Dropout | input: (None, 512, 2, 64) | output: (None, 512, 2, 64)
conv2d_9: Conv2D | input: (None, 512, 2, 64) | output: (None, 256, 2, 128)
dropout_9: Dropout | input: (None, 256, 2, 128) | output: (None, 256, 2, 128)
conv2d_10: Conv2D | input: (None, 256, 2, 128) | output: (None, 128, 2, 256)
dropout_10: Dropout | input: (None, 128, 2, 256) | output: (None, 128, 2, 256)
flatten_3: Flatten | input: (None, 128, 2, 256) | output: (None, 65536)
dense_7: Dense | input: (None, 65536) | output: (None, 1)
input_4: InputLayer | input: [(None, 19)] | output: [(None, 19)]
dense_6: Dense | input: (None, 19) | output: (None, 2432)
reshape_3: Reshape | input: (None, 2432) | output: (None, 64, 2, 19)
batch_normalization_12: BatchNormalization | input: (None, 64, 2, 19) | output: (None, 64, 2, 19)
conv2d_transpose_12: Conv2DTranspose | input: (None, 64, 2, 19) | output: (None, 128, 2, 64)
batch_normalization_13: BatchNormalization | input: (None, 128, 2, 64) | output: (None, 128, 2, 64)
conv2d_transpose_13: Conv2DTranspose | input: (None, 128, 2, 64) | output: (None, 256, 2, 32)
batch_normalization_14: BatchNormalization | input: (None, 256, 2, 32) | output: (None, 256, 2, 32)
conv2d_transpose_14: Conv2DTranspose | input: (None, 256, 2, 32) | output: (None, 512, 2, 16)
batch_normalization_15: BatchNormalization | input: (None, 512, 2, 16) | output: (None, 512, 2, 16)
conv2d_transpose_15: Conv2DTranspose | input: (None, 512, 2, 16) | output: (None, 1024, 2, 1)

Figure 72. Design of the discriminator (left) and the generator (right) networks used in the GAN.

As opposed to the classification neural network design, where objective metrics for the performance of the model could be devised, GAN can only be assessed by a human in a subjective manner. As the GAN-focused work is a very early-stage proof-of-concept activity, the design decisions for the GAN are made in line with the best practices. The optimiser is Adam, with the learning rate selected to ensure equal speed of learning between the generator and the discriminator. The loss for the discriminator is binary crossentropy, as it is, in essence, a classification network designed to differentiate between 'real' and generator-made examples. The loss for the generator is mean absolute error, with the rationale based on the continuous nature of the output.

The work on GAN focuses on simulating defect indications and cut ends of the pipe while using only the T(0, 1) mode. This is motivated by the relatively unexplored area of generative

ML and this work was primarily a proof-of-concept demonstration. The data used for training the GAN is simulated at 21 kHz carrier wave frequency and just the time-domain signal is used for training. It is therefore interesting to investigate whether the frequency-domain characteristics of the guided wave signal are accurately modelled by the GAN. Thus, the experiment is simple enough that it can be easily adapted to other NDT modalities, where different data sources and features would need to be generated, but the outputs here are complex enough to allow for the assessment of GAN capability for modelling ultrasonic data more generally.

6.1.2 GAN Results

The GAN design is a multi-stage process, and the final version is a network able to generate either a pristine trace, a trace with a single defect indication or one with two defect indications. All of the traces additionally contain a backwall signature.

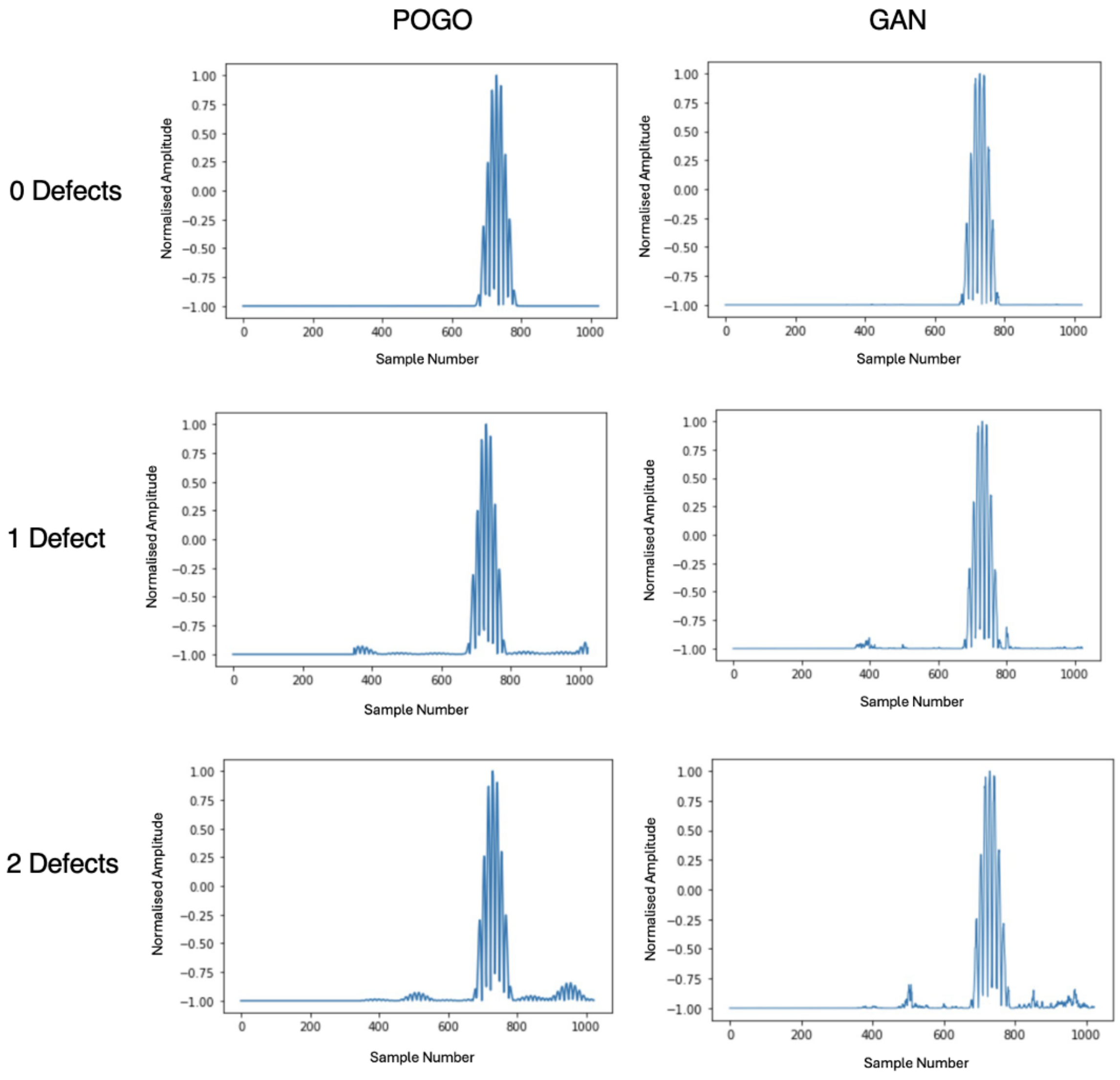


Figure 73. The comparison of the data simulated using Pogo FEM package (left) and the data generated using GAN (right). The first row shows the pristine trace, second contains one defect indication and third contains two defect indications. All the signals contain a back wall around sample 700. The second defect indication appears as if it exists behind the back wall, in fact it is generated in the opposite direction.

Figure 73 shows the comparison of the data generated using Pogo FEM package and the ones generated using GAN. Even to an untrained eye it is quite clear which one is which, with the FEM time traces being significantly smoother. It is worth noting, however, that the GAN has captured the difference in the amplitudes of the back wall and the defect indications, indicating some capability for identifying and replicating different features.

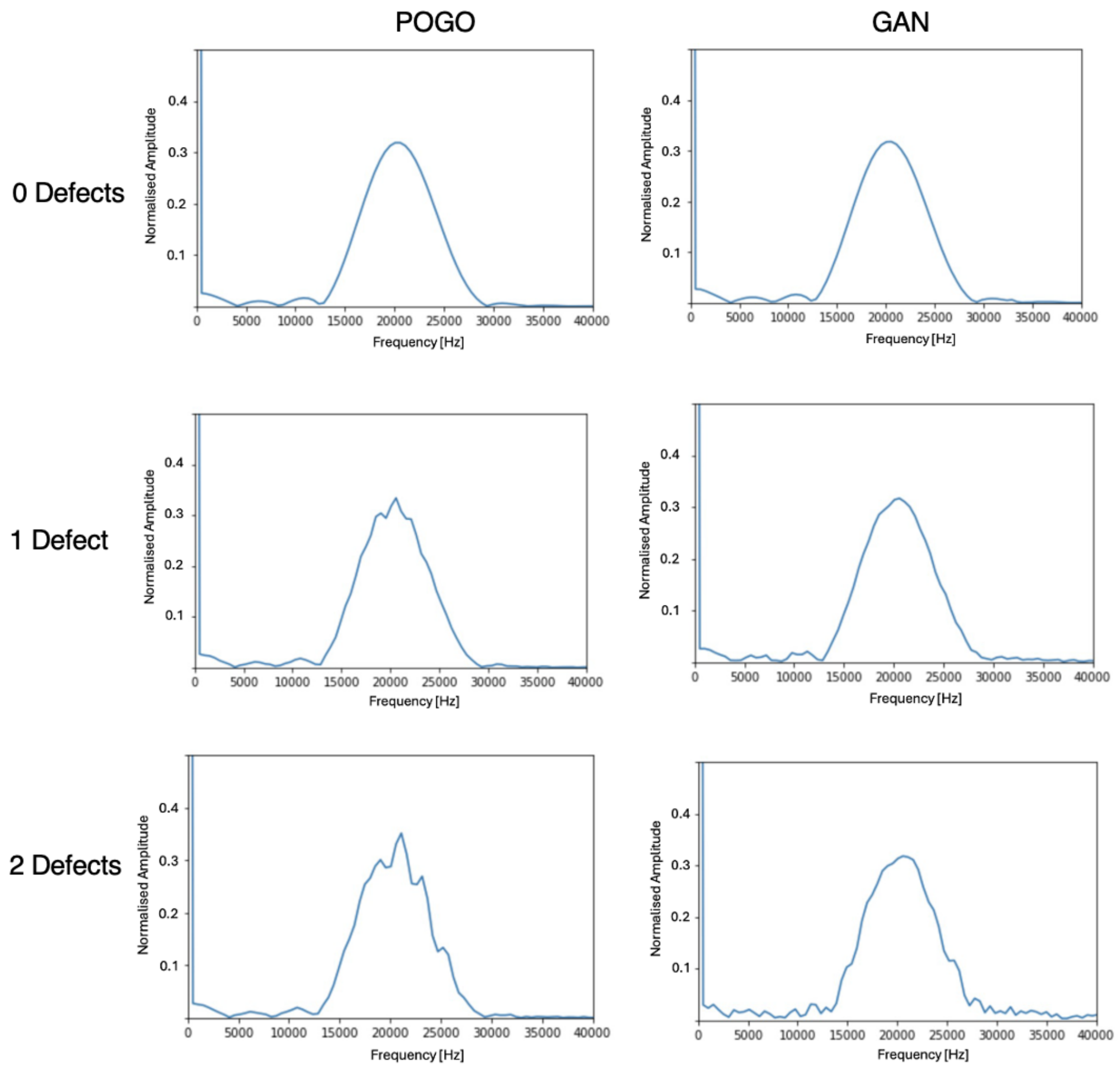


Figure 74. The frequency domain comparison of the data simulated using Pogo FEM package (left) and the data generated using GAN (right). GAN is used directly to simulate a rectified signal. The first row shows the pristine trace, second contains one defect indication and third contains two defect indications.

Figure 74 shows the frequency domain version of Figure 73. In a similar manner to the time-domain results, the GAN has the capability to generate the pristine frequency spectrum with very good accuracy. This is mostly visible when investigating the side lobes around the carrier frequency. They are smooth and similar between the generation methods for the pristine trace but diverge with the addition of defect indications. Importantly, the network correctly models the carrier frequency of 21kHz with much of the energy of the signal contained around this point. Less optimistic is the fact that in FEM data the inclusion of defect indications mostly leads to the changes in the shape of the main lobe in the frequency domain, while in the case of the GAN the main lobe barely changes, with much of the shift in the amplitude of the broadband signal. This leads to a conclusion that GAN has some capability to model the underlying frequency spectrum based on the time domain signal, but it needs either a more complex architecture, a larger dataset or the inclusion of the frequency-domain similarity in a custom loss function.

In conclusion, GAN is a promising technology, and the generation of guided wave ultrasonic signal is not beyond its capability. It is however a very complex form of machine learning with careful and systematic research needed to achieve satisfactory results. Once trained, though, it has the generation speed of 20000 traces per second when generating in the experimental conditions of Google Colaboratory T4 GPU runtime [98]. This corresponds to 200 km of pipeline simulated each second. This scale is beyond the capabilities of any available FEA modelling software, not to mention the experimental data gathering. Thus, developing the ability to use ML methods for generating data would provide an immensely valuable new method for realistic data acquisition.

6.1.3 GAN Applications in GWT

The main characteristic of GANs is their ability to generate novel realistic samples on demand, which can be used for training machine learning classification models. This is especially important if the models are to move from the qualitative detection to quantitative characterisation of defect indication severity. The added complexity of using ML for quantitative inspection in the guided wave context is that the already scarce defective data is further reduced when the 'defect indication' category is divided into the multiple sizes. A conditional GAN can generate a dataset of defect indications sized according to the researchers' choice on demand, thus reducing the problem.

Furthermore, GAN architectures can be used for neural style transfer. This technique is based on the concept of separating the content of the example from its style. It is easy to understand it in the context of art. Van Gogh's "Sunflowers" can be separated into the content – sunflowers and the style – characteristic impressionist brushwork of Vincent Van Gogh. In the NDT context the content are the features present in the trace, while the style consists of all the other factors impacting the trace. Thus, using this approach it is conceivable to build an algorithm translating between the real in-service data and FEM-simulated clean signal. This would have two potential use cases – the denoising of inspection traces so they are easier for the inspectors to analyse and the addition of realistic noise to simulated data. Such traces could in turn be used for training classification neural networks. This follows the concept of transfer learning using the pre-training data as close to the target distribution as possible. Finally, the ability to generate realistic samples on demand would be useful in the training of the inspectors and tracking their progress in the analysis of test results. This would create an objective measure of the quality of the inspector's skills and thus improve the trust in guided wave testing as a method. This final use for the GAN-originated data is least technical and would require a training and certification regime overhaul, but it could be one of the ways to utilise advanced machine learning to improve the quality of GW inspections without raising serious safety and reliability concerns.

6.2 Trust and ML Reliability in NDT Context

The second section of this chapter deals with the issues of trust and reliability of machine learning solutions in the NDT context. Some of the explainability and interpretability methods have been introduced previously and the goal of this chapter is not to provide the technical information on the implementation of the methods, for which an interested reader is referred to [99]. The chapter considers instead the implications of those trends in the context of NDT. The chapter will conclude with proposals for approaches that could improve the trust in ML solutions when applied to safety-critical NDT.

6.2.1 Trust and Reliability Issues

NDT is traditionally a conservative industry, which is due to the safety-critical nature of the work performed; failure of NDT to detect a defect indication could result in catastrophic failure and loss of life or significant financial impact. As such, trust in the inspection method and the inspector is of paramount importance. Currently, methods are qualified for inspections by independent assessors or, in the case of large NDT users, internally validated for the specific case. The inspectors, in turn, are trained and examined in their ability to use a specific NDT modality and certified accordingly. This certification is typically issued by institutions and industry associations, such as American Society of Non-destructive Testing (ASNT) [100] or British Institute of Non-Destructive Testing (BINDT) [101]. The landscape is more complicated when it comes to more recent, less widespread inspection methods, such as guided wave testing. In the latter case the training and certification role is fulfilled by Guided Ultrasonics Ltd. (amongst others), who is also a hardware provider. Additionally, large

NDT users often develop bespoke testing setups and internally certify the inspectors in the usage of their specific method.

Different industry sectors have different standards for NDT techniques appropriate to their needs. However, a uniform theme is one of conservatism. This makes the introduction of novel methods, such as machine learning, extremely difficult. Thus, achieving trust in ML applied to NDT is not strictly a technical issue, but the problem of garnering the confidence of industry. Inevitably for business, the main justification for development of ML techniques is cost reduction, and any change must be sympathetic to existing approaches to avoid costing more. Thus, any attempt to roll out ML for NDT more broadly is dependent on developing ML measures that save operating expenses without incurring capital expenses by changing the established processes in a drastic way.

An alternative approach is proposed by, for example, the European Network for Inspection and Qualification (ENIQ) [102]. It proposes a novel method for the qualification of ML-based approaches to non-destructive testing based on the existing qualification methods. See Figure 75 for details.

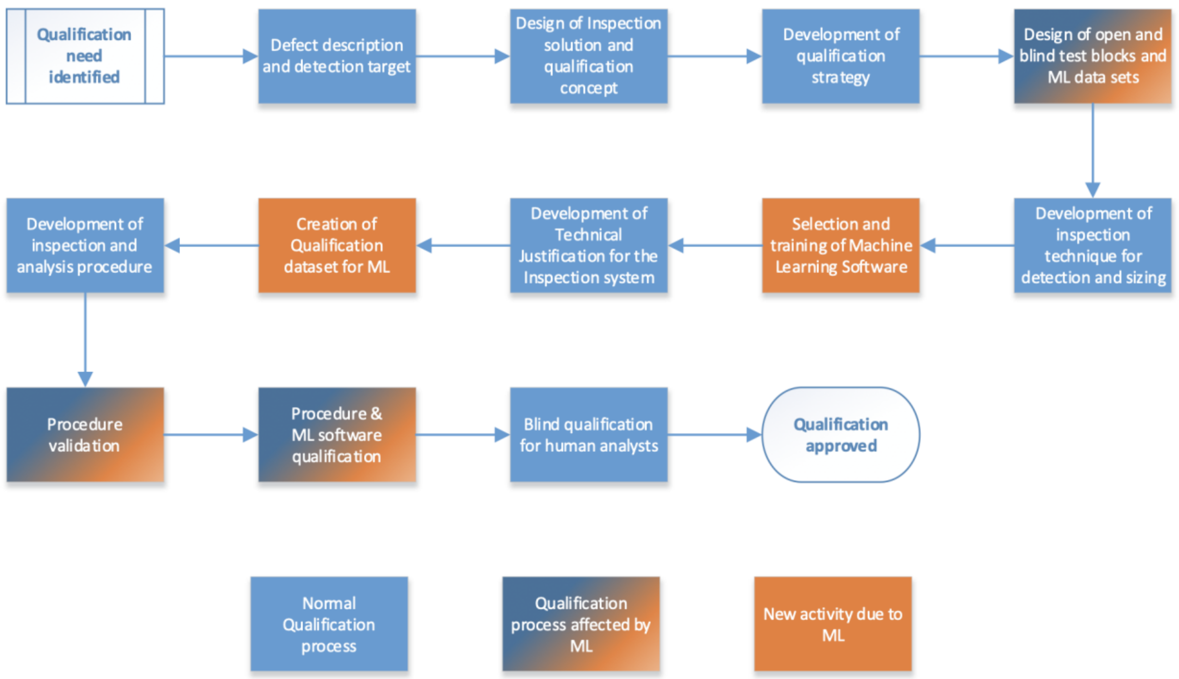

Qualification need identified
Defect description and detection target
Design of Inspection solution and qualification concept
Development of qualification strategy
Design of open and blind test blocks and ML data sets
Development of inspection technique for detection and sizing
Selection and training of Machine Learning Software
Development of Technical Justification for the Inspection system
Creation of Qualification dataset for ML
Development of inspection and analysis procedure
Procedure validation
Procedure & ML software qualification
Blind qualification for human analysts
Qualification approved
Normal Qualification process
Qualification process affected by ML
New activity due to ML

Figure 75. Flow chart highlighting the stages in the qualification process where ML has an impact. Reproduced from [102].

Both the report and the qualification method provide a useful approach to building trust in the machine learning methods and creating databases of ML models that can be assessed against each other. It does not, however, answer the question of building trust to the stage that allows for the transition to qualifying a novel ML-based approach. The report concludes that ML software qualified using the method proposed can be used in the same way as any other automated inspection software.

7 Machine Learning for Limited View Compensation

This chapter describes a standalone project in the utilisation of a ML-based methodology to solve a long-standing problem in NDT – the limited view artefacts in tomographic imaging. The chapter introduces the limited view imaging problem. It follows by introducing the mathematical basis for the distortion caused by limited view transducer configuration and the state-of-the art compensation algorithms. It follows by exploring the area of ML for image denoising to justify the architecture selection. Following, it describes the training and testing datasets as well as the ML architecture utilised and the hyperparameter tuning process. Finally, it shows the results in the terms of performance on laser-scanned corrosion patches as well as some interesting out-of-distribution samples. The performance is compared to state-of-the-art positivity regularisation algorithm.

7.1 Introduction

Tomographic reconstruction from ultrasonic data is a widely used technique in areas of medical diagnostics, such as breast cancer detection, non-destructive testing (NDT), and geophysics. Compared to the more common reflection images, these reconstructions provide quantitative information about the local material properties of the interrogated medium, most commonly the speed of sound. To perform the reconstruction of a sound speed map of the volume under inspection, it would be ideal to have access to the data probed from all possible angles – the so-called full view configuration. However, in many scenarios it is impractical or impossible to achieve this idealised situation, meaning that a reconstruction must be produced from a subset of the angles in reconstruction. It could be that data is subsampled because of time or cost limitations restricting the number of measurements that can be taken, or that there is a physical restriction meaning that only a range of angles is measurable.

This problem typically becomes ill-posed, as multiple sound speed maps can exist that correspond to the measured dataset, and addressing this is critical for producing accurate reconstructions in such cases. In many algorithms, particularly traditional direct approaches, an absence of data is effectively treated as having a measurement of zero, and the reconstruction is produced accordingly. However, this introduces strong artefacts into the reconstructed image, which can, in turn, obscure features of interest [103].

Alternative approaches attempt to compensate for the missing data. One class of approaches is to pre-process the data, replacing the unknown angle data with an estimation of what it should be. This approach has been investigated in the area of medical X-ray computed tomography [104]. However, X-ray behaviour diverges from ultrasound, being approximated (sufficiently accurately for almost all CT routines) by a straight ray assumption where the attenuation is reconstructed from changes in measured amplitude. Therefore, little work in this area (e.g. [105], [106]) is applicable to ultrasonic applications.

The other conventional approaches to limited view compensation are integrated into the image generation algorithms and can broadly be delineated into minimisation and projection regularisation approaches. Minimisation involves modifying the unknown data to solve an optimisation problem in which the resulting image is a best fit to both the measured data and some external restriction, often corresponding to a physical restriction. While this approach has been proven effective for some scenarios [107], it has the significant downside of requiring multiple iterations running some form of forward model in order to fit the measured data, and may also suffer from traditional challenges that often arise with iterative methods, including instability and the presence of local minima. The second approach, projection regularisation, involves the post-processing of the image. In this, the image is taken, and a particular projection is performed, often aligned with some physical limitation. The most common example is the restriction of positivity (such approach being termed "positivity regularisation"): in CT, for example, negative values are impossible since they would correspond to negative x-ray attenuation. Therefore, the projection involves removing any negative values, typically by setting these to zero. This process means that the image will be moved away from fitting the data, so an iterative process begins where the image is alternatively projected onto the data and then onto the physical constraint, ultimately converging onto a solution which matches both. This was developed within ultrasonics into the VISCIT technique [108], which used a non-zero adaptive threshold and iteratively reconstructed the image from high contrast through to low contrast features.

Machine Learning (ML) can be applied to a broad range of imaging applications, with such techniques utilised to eliminate artefacts specific to the imaging modality, from recovering the resolution of optical images to speeding up the acquisition of MRI and improving contrast in ultrasound imaging [109]. The ML algorithms can be applied either directly to image-domain data [110], unrolled images (transformed into 1D vector of pixel values) [111], or domain-transformed images [112]. Most success has been found in the application to image-domain data itself, where there has been a significant amount of interest from the computer vision community.

Within NDT, there are two main areas of quantitative ultrasound tomography application: guided wave tomography and early-stage damage detection. In the former, corrosion damage causes thickness loss in a pipe wall or plate-like structure, which affects guided wave speed due to dispersion and hence provides a mechanism for reconstruction. Early-stage damage, i.e. prior to the formation of discrete defects like cracks, will often manifest as subtle localised changes in material properties, which can be measured through tomographic ultrasound methods. In both cases, physical access is often limited: in guided wave tomography a pipe support may surround the pipe preventing measurements in all directions, and the physical geometry of a component is likely to prevent the ability to inspect a region with potential damage in from all directions.

This chapter aims to develop machine learning techniques for developing accurate reconstructions in limited view ultrasound tomography, and specific focus will be on guided wave tomography for corrosion mapping, although the principles should be broadly applicable both within NDT and medicine.

The background theory is described in section 7.2. Section 7.3 presents the experimental design and methodology. Section 7.4 discusses the performance of the proposed ML-based approach on both artificial thickness maps and experimental corrosion-type images. Finally, section 7.5 concludes the work.

7.2 Theoretical background

7.2.1 Limited view imaging

The ultrasound scattering problem will be considered under the Born approximation [113]; this is a linearisation of the wave scattering problem which makes inversion practical in the case of low contrast, small-scale scatterers. Modifications have been developed to account for cases where the approximation is not applicable, such as the Born Iterative Method [114], Distorted Born Iterative Method [115] and HARBUT [116]. The underlying principles of the Born approximation are still valid in these methods (although the linearisation is typically applied within a single iteration) so the techniques developed under the Born approximation are taken to generalise to such scenarios.

Under the Born approximation, the following is valid:

$$u_s(\boldsymbol{r}) = \int g(\boldsymbol{r} - \boldsymbol{r}')o(\boldsymbol{r}')u_0(\boldsymbol{r}')d\boldsymbol{r}' \tag{77}$$

where u_s is the scattered field, g represents the Green's function in free space, $\boldsymbol{r}$ the distance, and u_0 is the incident wavefield. $o(\boldsymbol{x}) = k_0 \left[\left(\frac{c_0}{c(x)}\right) - 1\right]^2$ is the object function with c_0 being the background sound speed, c the local sound speed, and k_0 as the background wavenumber.

Considering an array exciting in the far field, the wavefronts intersecting with the scatterer can be considered to be planar, as well as the scattered components received at the measurement array, and therefore eq. (1) becomes

$$u_s(\boldsymbol{r}) = \int o(\boldsymbol{r}') \exp[ik_0(\boldsymbol{s}_0 - \boldsymbol{s}).\boldsymbol{r}']\, d\boldsymbol{r}' \tag{78}$$

where vectors $\boldsymbol{s}_0$ and $\boldsymbol{s}$ represent the incident and scattered wavefields respectively. Under the assumptions made (far field, Born approximation) each measured scattered value is a component of the Fourier transform of the object function. This provides a mechanism for inversion of the field, by collecting the measured data then performing an inverse two-dimensional Fourier transform. The exact form of this can vary, with some approaches suggesting resampling the measured data to a uniform grid to enable the Fourier transform, to non-uniform techniques where additional weightings need to be included [117] [118] [119]. This provides a mechanism for evaluating the limited view problem, since missing measured components can be directly mapped to missing components in the Fourier transformed space.

Figure 76(a) presents a schematic of a representative limited view configuration, and Figure 76(b) then illustrates the components which are measurable from this setup. As highlighted before, the challenge of limited data imaging is to establish what values are best to use for the unknown components to ensure the final image is as accurate as possible.

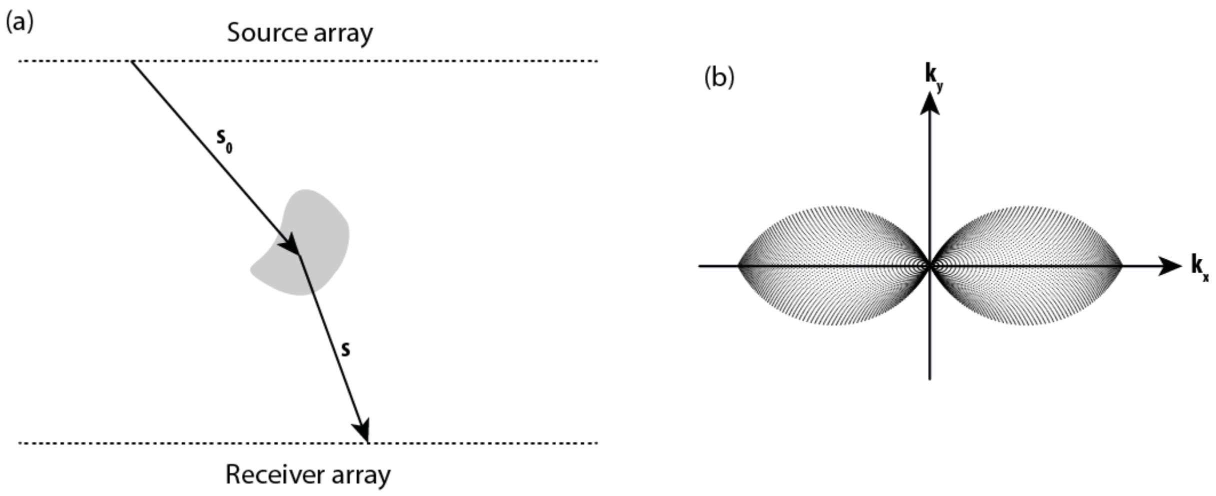


Figure 76. (a) limited view transmission configuration, with a source array above the scatterer and a receiver array below. (b) positions of measurements in K-space (spatial frequency domain) showing that there are significant angles which cannot be measured.

7.2.2 Machine learning

7.2.2.1 Machine learning for image quality improvement

Image processing can be defined as the operations where an input is an image, and the output is either an enhanced version of the image or a collection of information and insights gathered from one. Outside of the ML domain, the example of the former is bandpass filtering, while the latter is signal-to-noise ratio. Traditional methods remain popular thanks to their determinism and integration within processes, but ML is increasingly gaining usage share. The rationales for using ML vary between the users but can generally be classified into ease of use (i.e. replacing analytic image reconstruction), operational cost (i.e. replacing iterative reconstruction) and solving novel problems (i.e. semantic segmentation). This breadth of potential applications motivates the quick development of ML for image processing. This section intends to briefly introduce the methods used for the improvement in the quality of images and explain the rationale behind the selections made in the work.

This review omits two of the prevalent deep learning-based image processing techniques: ML-based image reconstruction algorithms [120] and ML-based semantic extraction algorithms [121]. The reconstruction algorithms take in the sensor data, either raw or initially pre-processed and output a predicted image. Conversely, the semantic extraction algorithms take in an image and output the relevant information content. For both categories, there is a lack of direct input-output mapping. The work undertaken is concerned with the improvement in the quality of the image, thus the only type of algorithm considered is an image-to-image translation.

The most common category of deep learning applied to image-to-image translation is performed directly in the image domain. This is motivated by a broad availability of well-tested algorithms from the area of computer vision. Indeed, much of the early progress of Machine Learning was based on the gradually improving image-based algorithms, such as LeNet [122], VGG-Net [123], ResNet [124] and in the more modern times U-Net [125], EfficientNet [126] and YOLO [127]. The availability of well-tested algorithms makes image domain preferable to sensor or frequency domains.

Limited view imaging requires, in effect, an artefact compensation algorithm. The complicating factor is the difference in the application area. Conventionally image processing algorithms are applied to natural images, scraped from the web, while the use case for this work is corroded metallic samples. Thus, while it is possible to transfer the methods, they need to be carefully evaluated for suitability. While the authors were unable to find reports of the application of ML to limited view imaging in the context of NDT, such investigations have been conducted, although in limited numbers, in medical photoacoustic imaging [128], [129]. The preferred architecture for the image-to-image denoising tasks is an encoder-decoder architecture. This has been utilised directly [130], [131] and with modifications based on the domain requirements such as the usage of a U-Net type architecture [132], [128], [129]. A U-Net is a combination of an autoencoder and a ResNet, originally used for image segmentation, but since then proven to be the preferable network for a wide range of applications, including non-destructive testing [88], [89], [90], the discriminator parts of generative adversarial networks [91], [92], image denoising [93] or speech enhancement [94]. It is most notable for this work, as it has been used in a series of attempts to solve the limited-view problem in photoacoustic imaging [128], [129]. Some of the work attempted the use of multilayer perceptron on image data [133], which has recently been adapted to the mainstream encoder-decoder paradigm by the usage of Dense U-Net [129].

Image domain learning is the predominant choice in image quality improvement, but some alternative approaches have been attempted, most importantly solving the image quality problems in the sensor domain and using generative learning. The former involves employing the ML approach before the image reconstruction, based on the premise that the sensor output is the richest data source available, thus an ML algorithm may be able to analyse and modify it, so the noise is not reconstructed into the image in the first place. This approach can also be used to speed up the acquisition by hallucinating the signals sensors would provide if placed in between the existing sensors [134]. Finally, perhaps the most modern image quality improvement technique is the usage of Generative Adversarial Networks (GANs). This type of neural network generates completely novel data based on the input. GANs are notorious for hallucinating impossible data, however, this issue has been solved by the usage of cycle-consistent GAN (CycleGAN) [135], which ensures the original data can be retrieved from the generated image. This approach has been successfully applied to medical CT [136]. However, GANs are difficult to train, suffer from very low explainability and have high data requirements, making this approach a choice of last resort.

Taking into consideration both the results of [128], [129] and the fact that encoder-decoder architectures are generally considered the models-of-choice for image-to-image denoising tasks [25], [26], this work primarily investigates the use of denoising autoencoders.

7.2.2.2 Denoising autoencoder architecture

The work presented in this chapter performs an image-processing task that is essentially denoising. Based on the review presented in the previous section, the design paradigm best suited to image quality improvement is an encoder-decoder type of architecture. It is defined by containing an encoder, which maps the input to a low-dimensional latent space and a decoder which uses the latent space as an input to construct the output. Therefore, the encoder can be understood as refining the information present in the original data, filling the role of a feature engineer. The decoder, in turn, is the decision-making network.

An autoencoder is a generic term for an encoder-decoder, whose output is identical to the input [137]. A denoising autoencoder is a specific application, where the input is first corrupted with noise before being passed through the ML architecture. The original motivation for the denoising autoencoder was based on the theory, that corrupting the input signal may help the encoder learn to reject the irrelevant information, thus improving its feature extraction capability. Now, it is used extensively by the signal processing community to restore the noisy inputs to de-noised outputs.

The denoising autoencoder is reliant on a training dataset composed of two matched subsets, the noisy inputs, and the known noiseless outputs. This requirement naturally makes the design paradigm well suited to the applications where two conditions are met: the noiseless data is abundantly available, and the noise sources are deterministic and easy to simulate. An example of a perfect application is sharpening out-of-focus photographs, given the abundant availability of in-focus examples and the ease of blurring. If one of the conditions is not met, either the noise or the noiseless data needs to be artificially generated, which introduces the need to carefully test the trained model against the intended use case and to be aware of its limitations.

Finally, an architecture achieving similar goals is a much more modern CycleGAN. The decision on the utilisation of either of the two architectures is ultimately dependent on the available dataset. CycleGAN, as an unsupervised learning architecture, does not require the matching between the noiseless and the noisy set, which is important for some applications, especially ones where numerically simulating an idealised scenario is possible, but simulating realistic noise is not. However, as with many a generational architecture CycleGAN requires very large training sets, and the training set must be drawn from the same distribution as the intended application. Furthermore, generational architectures are less explainable, making them less preferable than conventional architectures in safety-critical applications.

7.3 Methodology

7.3.1 Data sources and processing

The ML model developed in this work is trained on artificial data and is tested on both artificial and experimental samples. Each numerical or real data sample represents thickness maps obtained via ultrasonic inspection and consists in a pair of single-channel (i.e., grayscale) images having 128x128 pixels, where each pixel value corresponds to the local thickness change. The two images forming each pair are obtained using limited and full views, and are used as input and output (i.e., ground truth) to the model, respectively. All images are normalised such that their pixel values range between -1 and 1.

A physical limited view configuration is considered. This is far field, with a transmission array above the object and reception array below, as illustrated in Figure 76(a). The viewing angle of each array is 145.6 degrees.

7.3.1.1 Artificial data

Full view thickness maps are artificially generated as “blobs”, i.e., circular changes in thickness whose value is maximal centrally and falls off according to a Hann (raised cosine) function, overlaid on a background consisting of zero-mean uniform noise with values ranging between -0.2 and 0.2. Examples of such maps are given in Figure 77(b, d, f). The noise is used to facilitate the ML training, since a uniform background encourages the algorithm to generate images as close to the background level as possible.

Each full view thickness map is then modified to simulate a limited view inspection scenario. In this process, the image has a 2D Fourier transform applied to it, and each component is then assessed to evaluate whether it is measurable using the limited view configuration. Any unmeasurable components are set to zero, and the result is then inverse Fourier transformed back. Examples of the resulting limited view images are given in Figure 77(a, c, e).

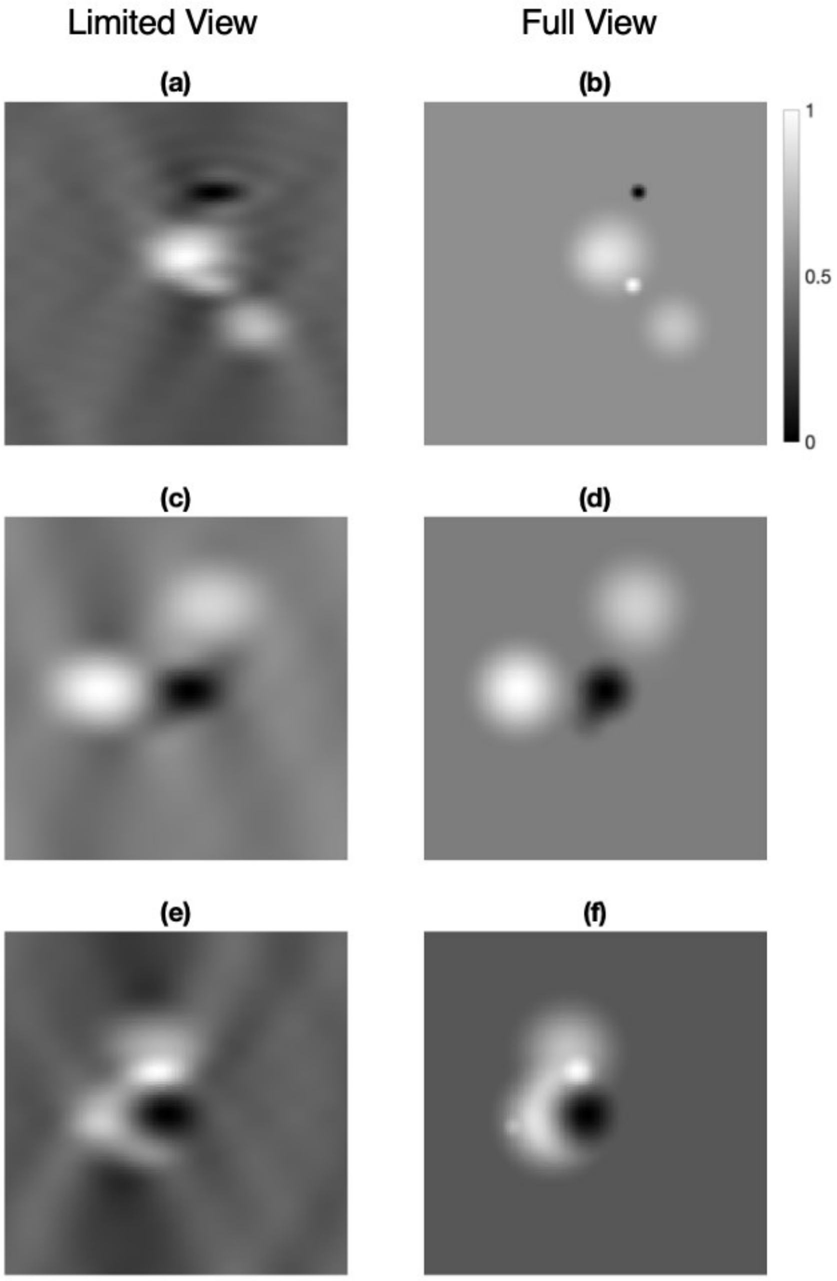


Figure 77. Examples of limited view registrations of artificial thickness maps (a, c, e) and their full view version (b, d, f). This type of data was used for the training of the ML algorithm.

The ML algorithm is trained on a dataset of 10,000 data samples, each containing five blobs, whose diameters and maximum amplitudes vary between 2% and 20% of the image dimension and between -1 and 1, respectively. Such dataset is split in training and validation sets with a 80/20% ratio. The artificial testing set is composed of 2,000 further images generated with the same procedure. Note that a preliminary study showed that using different numbers of blobs per image did not affect the performance in any significant manner, hence the choice of using a constant number of five blobs per image.

7.3.1.2 Experimental testing data

A set of ten experimental data samples is used to test the performance of the novel limited view compensation algorithm in a realistic inspection scenario. The data is acquired using laser scanning of in-service corroded samples, thus resulting in high-quality thickness maps of corroded samples. These are representative of the real cases which a guided wave tomography routine would need to successfully reconstruct.

Since the available raw images have non-uniform dimensions, they are first zero-padded to make them squared, and they are then downsized to 128x128 pixels using MATLAB "imresize" function [138]. This uses bicubic interpolation to calculate the output. Finally, the limited view projection of the thickness maps is calculated using the same process described previously for the artificial data.

7.3.2 ML architecture and training design

The ML architecture utilised in this work is a fully convolutional autoencoder. It follows the typical design of an autoencoder, but instead of using a fully connected latent layer, it is convolutional, as this design choice demonstrated a better performance when tested. As seen in Figure 78**,** the encoder part of the network is composed of three 2D convolutional layers with stride parameter 2, effectively compressing the input image by a factor of 8 in both directions. The latent dimension is therefore 16x16. This is followed by a 2D convolution on the latent dimension that does not further reduce the dimensionality, and then by the decoder portion of the network, consisting in three transposed convolutional layers that return the image to the initial size.

Some architecture modifications were tested, including the addition of skip-connections like a U-Net and the addition of Max Pooling or Dropout layers [139]. However, neither of the modifications improved the performance of the network.

The compiled architecture was trained on the dataset described in section 7.3.1.1 using Adam optimiser [140] and Mean Squared Error (MSE) loss. The Adam optimiser was selected as it is the first-choice optimiser in ML research (over 320 times more common than the second-placed Adafactor [141]). Adam is a self-tuning optimiser; therefore, the learning rate selection is significantly more lenient than in many older-generation optimisers. MSE is a standard loss function in image-to-image translation tasks.

Both the architecture and the training process are characterised by a limited number of tuneable hyperparameters. This has the advantage of enabling a broad range optimisation exercise. The parameters to be tuned are the initial number of filters, the size of the receptive field, the learning rate and the number of training epochs. The optimisation of the first three was performed using the Keras Tuner [142], which implements Hyperband hyperparameter tuning strategy [143]. Table 8 presents the hyperparameters to be tuned as well as their possible values. The tuner is set to use a reduction factor of 3 [143] and to select the best-performing model based on the validation loss when using a progressive number of training epochs (up to 20).

Table 8. The hyperparameters to be tuned in Karas Tuner [142] and their possible values.

Parameter	Possible Values
Initial Filters	16, 32, 48, 64, 80, 96, 112, 128
Receptive Field Size	1, 3, 5, 7, 9, 11
Learning Rate	1e-2, 1e-3, 1e-4, 1e-5, 1e-6, 1e-7

As a result of hyperparameter tuning, the initial number of filters was set to 80, the receptive field size to 3 and the learning rate to 1e-4, with the diagram of the resultant neural network shown in Figure 78**.** The optimal number of training epochs was then manually determined to be 100.

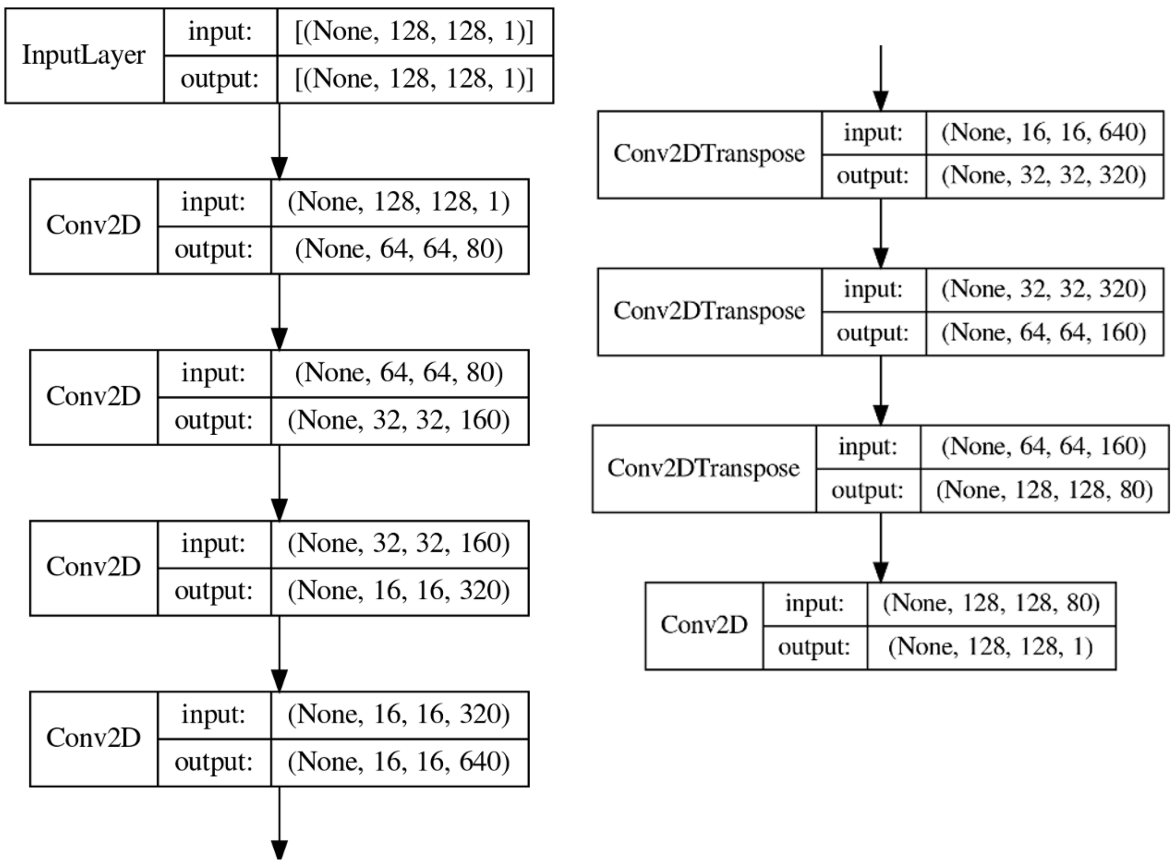

InputLayer
input: [(None, 128, 128, 1)]
output: [(None, 128, 128, 1)]
Conv2D
input: (None, 128, 128, 1)
output: (None, 64, 64, 80)
Conv2D
input: (None, 64, 64, 80)
output: (None, 32, 32, 160)
Conv2D
input: (None, 32, 32, 160)
output: (None, 16, 16, 320)
Conv2D
input: (None, 16, 16, 320)
output: (None, 16, 16, 640)
Conv2DTranspose
input: (None, 16, 16, 640)
output: (None, 32, 32, 320)
Conv2DTranspose
input: (None, 32, 32, 320)
output: (None, 64, 64, 160)
Conv2DTranspose
input: (None, 64, 64, 160)
output: (None, 128, 128, 80)
Conv2D
input: (None, 128, 128, 80)
output: (None, 128, 128, 1)

Figure 78. Optimised autoencoder architecture implemented in this work. The left column corresponds to the encoder path and the right column to the decoder path.

7.3.3 Metrics

The metrics for the evaluation of similarity of grayscale images are quite varied, but the application investigated in this work is characterised by one-to-one pixel mapping, thus justifying the usage of pixel-level similarity metrics. The standard pixel similarity metric is the MSE, which, as mentioned in the previous section, is used as the loss function during training. However, at the stage of the assessment of the results, the Root Mean Squared Error (RMSE) is more useful, as its values are on the same scale as the pixel values of the image. Furthermore, in the context of visualising changes in the thickness, the highest changes are the most important to get right, as they correspond to the potential points of failure. As a result, in addition to RMSE, the per-pixel Maximum Absolute Error (MAXAE) is also used.

7.4 Results and discussion

7.4.1 Performance on the artificial data

The initial assessment of the performance of the proposed algorithm trained on the 10,000 input data samples has been made on the testing set consisting of the 2,000 examples drawn from the same artificial generation process used for the training set. On this dataset, average values of 0.0223 and 0.2284 were obtained for RMSE and MAXAE, respectively.

As described in Section 7.3.1.1, noise was added to the artificial images used in the training dataset. This solution was implemented after noticing that the usage of uniform backgrounds (a grayscale level of zero, i.e., black) was encouraging the algorithm to generate images close to the background level, since such outputs generally give rather low values for the loss function, i.e., MSE.

Figure 79 shows an example of the results obtained when training the autoencoder on the noiseless dataset. The ground truth full view image of Figure 79 (c) features four blobs. These blobs are still discernible in the limited view registration of Figure 79 (a), though are significantly out of focus. Figure 79 (b) shows that the trained autoencoder has simply produced a uniform black image, thus completely ignoring the thickness changes. This still gives a low MSE since the vast majority of pixels in the ground truth image are equal to zero. While the global MSE minimum of the correct solution is better than the all-black output, it is a significantly more complex solution, thus in the absence of input perturbation the neural network tends to default to all-black local minimum.

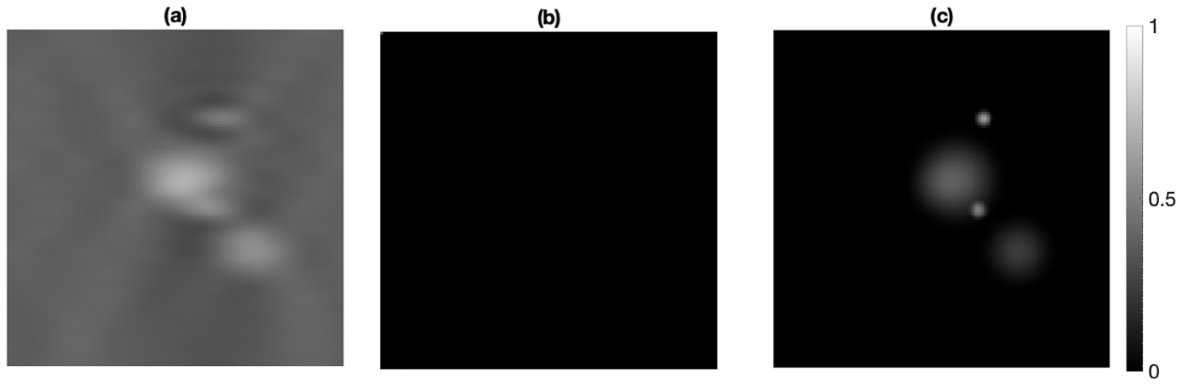


Figure 79. An example of the results obtained when the autoencoder is trained on the noiseless dataset. (a) input limited view registration; (b) output of the autoencoder; (c) ground truth full view image.

Figure 80 shows three representative examples of the results (one example per row of plots, with the left plot displaying the limited view registration input to the autoencoder, the middle one the output of the autoencoder and the right one the ground truth). Compared to Figure 79 the difference is the addition of small random noise to the input. All examples show visible improvements in the sharpness and contrast of the images processed by the autoencoder when compared to the original limited view registrations. However, Figure 80(i) shows a blob that has not been reconstructed by the autoencoder. That blob roughly coincides with another blob of similar amplitude, which probably explains the autoencoder’s inability to reconstruct it. This result serves as a reminder that even though the autoencoder appears to reconstruct nearly invisible thickness changes, it can still fail in extreme cases.

In summary, the trained autoencoder appears to perform very well in sharpening the simulated dataset and addressing the impact of the limited view approach. This is an expected result, as the training was performed on a dataset with the same distribution as the testing set considered in this section. The next section considers the important scenario of actual experimental data as well as a set of interesting cases of artificial images drawn from very different distributions.

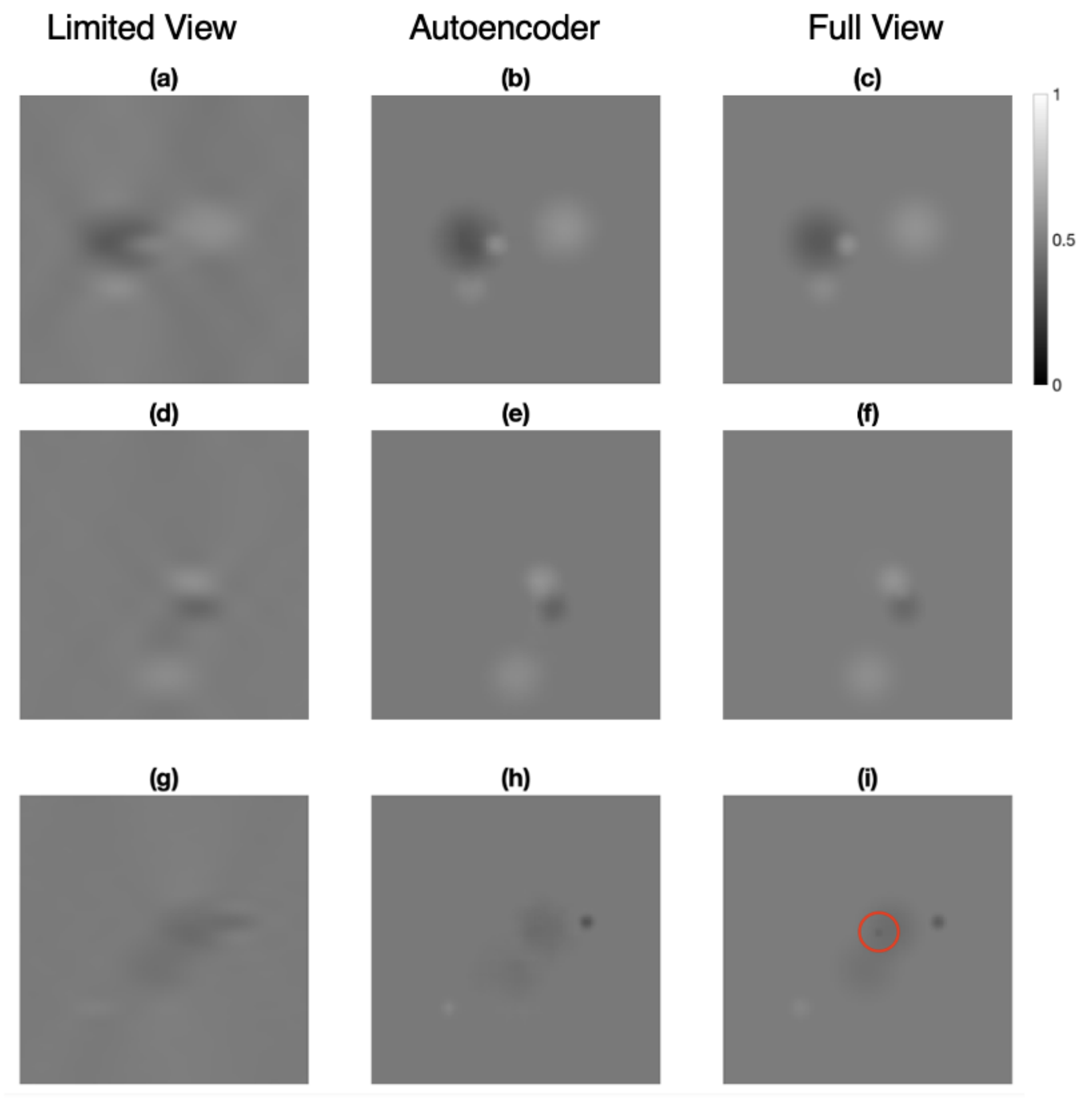


Figure 80. Artificial testing examples. Three thickness maps (a-c, d-f, g-i) are presented. For each, the limited view registration (input to the autoencoder) is shown at the left, the output of the autoencoder at the centre and the ground truth full view registration at the right. In subfigure (i), the plot is annotated with a red circle highlighting a feature missed by the autoencoder.

7.4.2 Performance on the Artificial Test Cases

The limited view transducer configuration causes some regions of the area of interest to be unmeasurable in the frequency domain (see Figure 76). These regions are known as null space. Thus, if a modification is made in the null space of the thickness map, such information should not be available via the limited view, no matter the processing method.

Clearly, the sensitivity of the configuration to spatial frequency components will vary depending on the direction; transmitting primarily vertically as in the example of Figure 76(a) will result in an insensitivity to horizontal variations, and these horizontal variations will be in the null space. Figure 81 shows three test cases that investigates this. The ground truth samples are horizontal stripes (c), vertical stripes (f) and the combination of the two (i). As expected, the limited view registration of the horizontal stripes completely misses such features (a), while that of the vertical stripes does a good job in their reconstruction (d). When the two stripe directions are combined, the limited view is still highly sensitive to the vertical ones, and, interestingly, also gives a hint on the presence of the horizontal ones (g). It is interesting to assess whether the trained autoencoder of Section 7.4 is somehow able to synthesise the missing data for such images.

The limited views shown in Figure 81(a, d, g) were then given as input to the trained autoencoder of Section 7.4, and the outputs are shown in (b, e, h), respectively. Unsurprisingly, the output in (b) is a pure hallucination and has nothing in common with the ground truth of (c). Notably, the autoencoder outputs in the two other instances are farther from the ground truth than the limited view registration. This is essentially due to the autoencoder's tendency to sharpen the image into the shape of blobs, which in these cases distort the outputs. These results serve as a reminder that an ML algorithm can only be as good as the data it has been trained on. Given that no sharp-edged straight geometries were present in the training set, the resultant algorithm does not perform too well when presented with such a problem. Two solutions can be implemented to improve the resilience of an ML algorithm to unseen data. One involves training on a significantly broader random training set, such as web-scraped images, while the other or systematically analysing the possible types of inputs and matching the training set. The first option has the disadvantage of losing the domain-specificity and requiring an increase in the network size, training duration and computational complexity at inference. The second is extremely work-intensive and reliant on the designer's ability to identify all the real-world cases and effectively simulate them. The second approach, pioneered in the NDT context by the work of Richard Pyle [144], involves implementing techniques such as ensemble learning to identify whether the input data comes from a known distribution. Such information can be used to reject the output of the ML algorithm as untrustworthy.

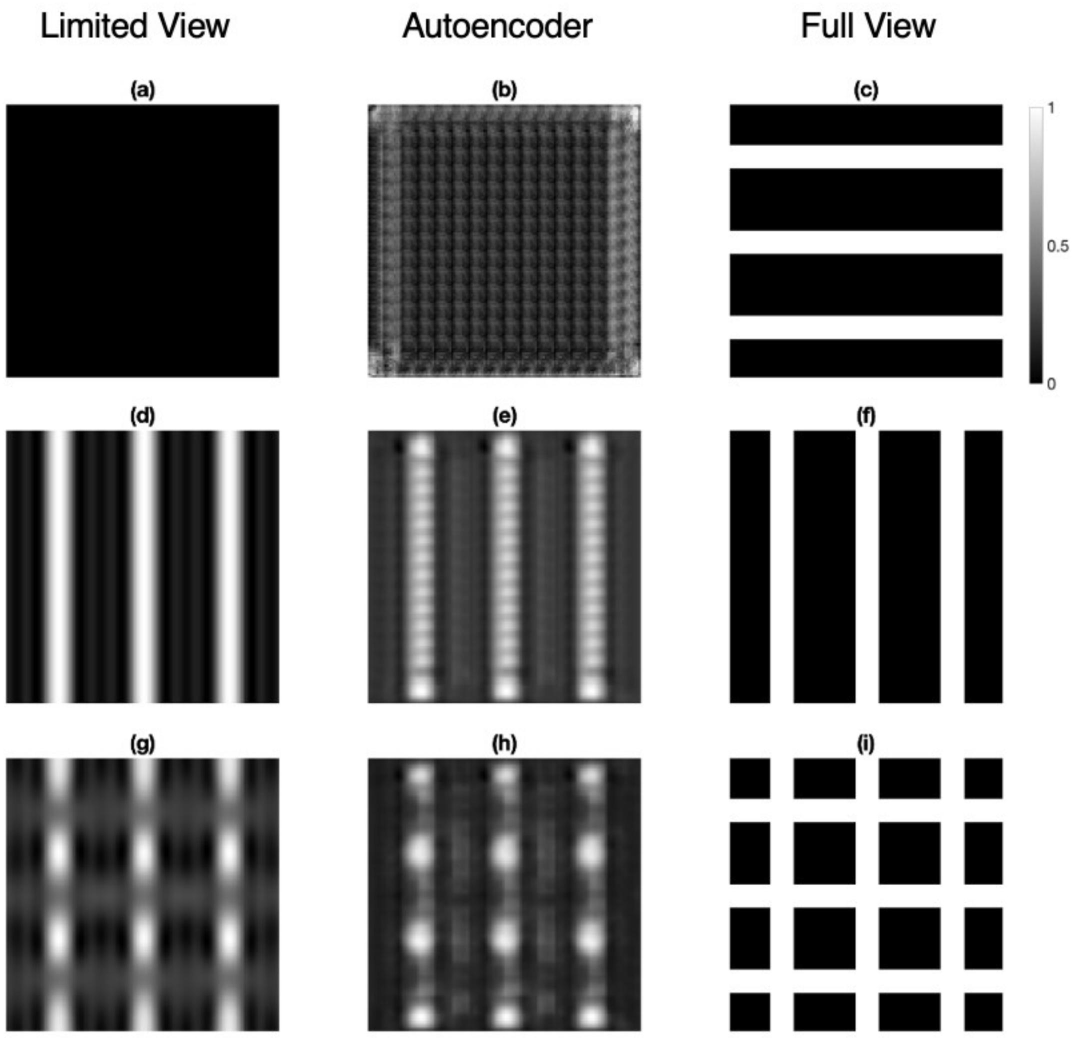


Figure 81. Limited view registrations (a, d, g) compared to the autoencoder output (b, e, h) and full view ground truth (c, f, i) of artificial geometric thickness maps including horizontal stripes (a-c), vertical stripes (d-f), and a combination of the two (g-i).

7.4.3 Performance on the real test cases

This section assesses the performance of the autoencoder trained on the artificial dataset when applied to the set of ten experimentally acquired corrosion patches described in Section 7.3.1.2. In this case, the results are also compared to those obtained using the state-of-the-art positivity regularisation method, described in Section 7.1, on the limited view images.

Figure 82 and Figure 83 display all the results. A visual inspection of the images reveals that the autoencoder significantly improves the contrast and focus of all test cases when compared to either the original limited view registrations or to their positivity regularized version. The improvement in focus is especially evident in Figure 82(e-h, q-t) and Figure 83(a-p), corresponding to elongated corrosion patches, which are common in linear contact scenarios (for example, a pipe on a simple support).

Table 9 lists the RMSE and MAXAE metrics for all test cases. Expectedly, the autoencoder errors are significantly higher than those obtained on the artificial testing dataset reported in the previous section. However, when compared to the limited view registrations, the autoencoder performs better in 18 out of 20 test scenarios, lowering the RMSE and the MAXAE by an average of 35 and 40%, respectively. The two cases where the autoencoder performed worse than the limited view registration are a 14% drop in the MAXAE of the third sample (Figure 82 (i-l)), for which, however, the RMSE improves by 42%, and a 5% drop in the RMSE of the fifth sample ((Figure 82 (q-t)), for which there is a 28% improvement in the MAXAE. In both cases the visual inspection of the plots shows a significantly enhanced sharpness of the image output by the autoencoder.

Compared to the positivity regularisation method, the ML algorithm typically performs better in terms of the MAXAE metric, with an average improvement of 11%. However, it tends to perform worse on the RMSE, with an average drop of 47%. The latter result is, however, strongly influenced by the downsides of RMSE as a metric for this type of tasks, since it can be significantly impacted by not-so-critical background mismatches. As a matter of example, the visual inspection of the fifth sample (Figure 82 (q-t)), where the autoencoder performs 163% worse than positivity regularisation on RMSE, shows that the ML model provides the overall best reconstruction of the ground truth image, though is strongly penalised by the background mismatch in terms of RMSE.

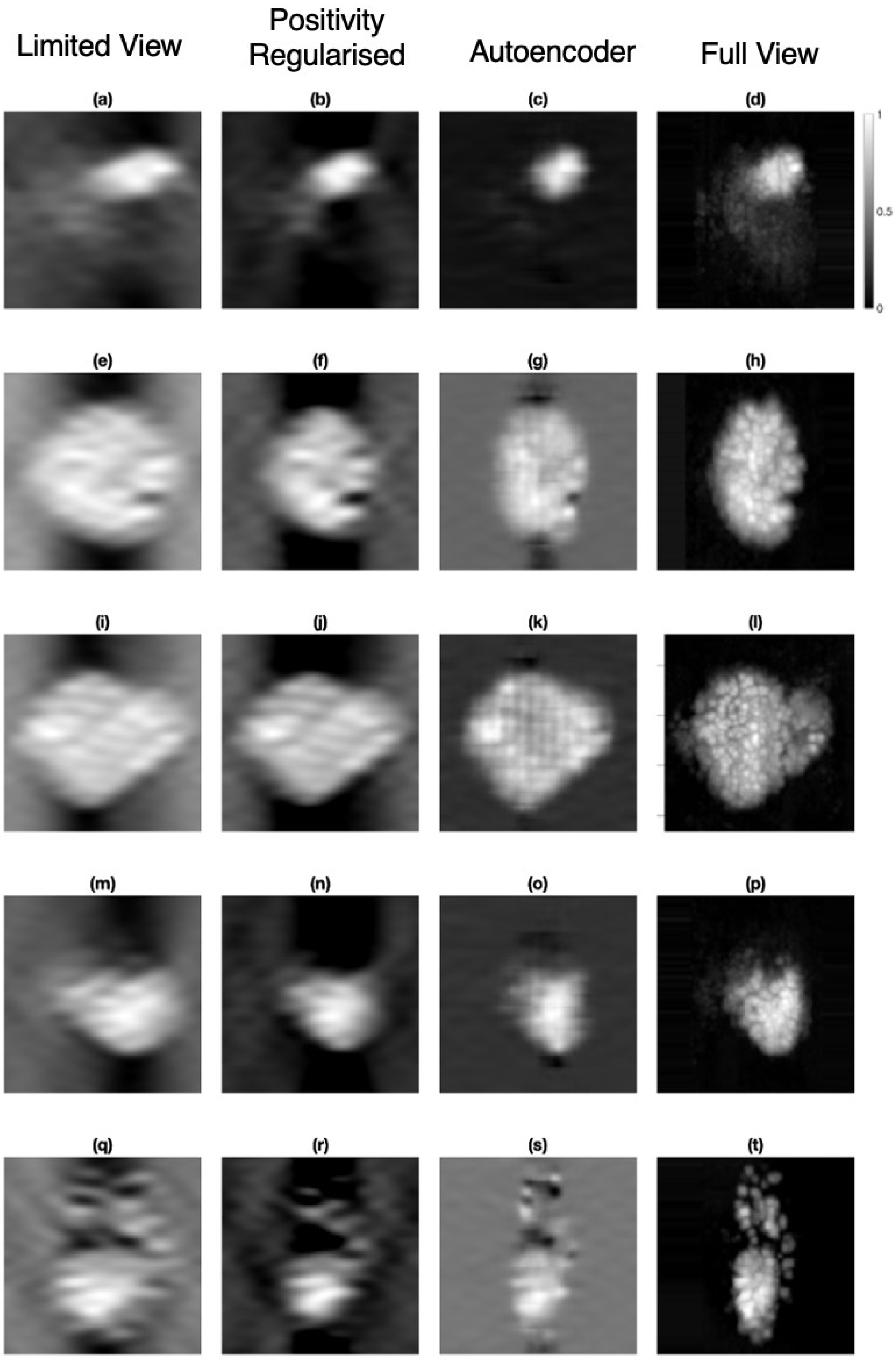


Figure 82. Experimental corrosion patches 1 to 5. Original limited view registration (a, e, i, m, q), image processed using the positivity regularisation method (b, f, j, n, r), output of the autoencoder (c, g, k, o, s) and full view ground truth (d, h, l, p, t).

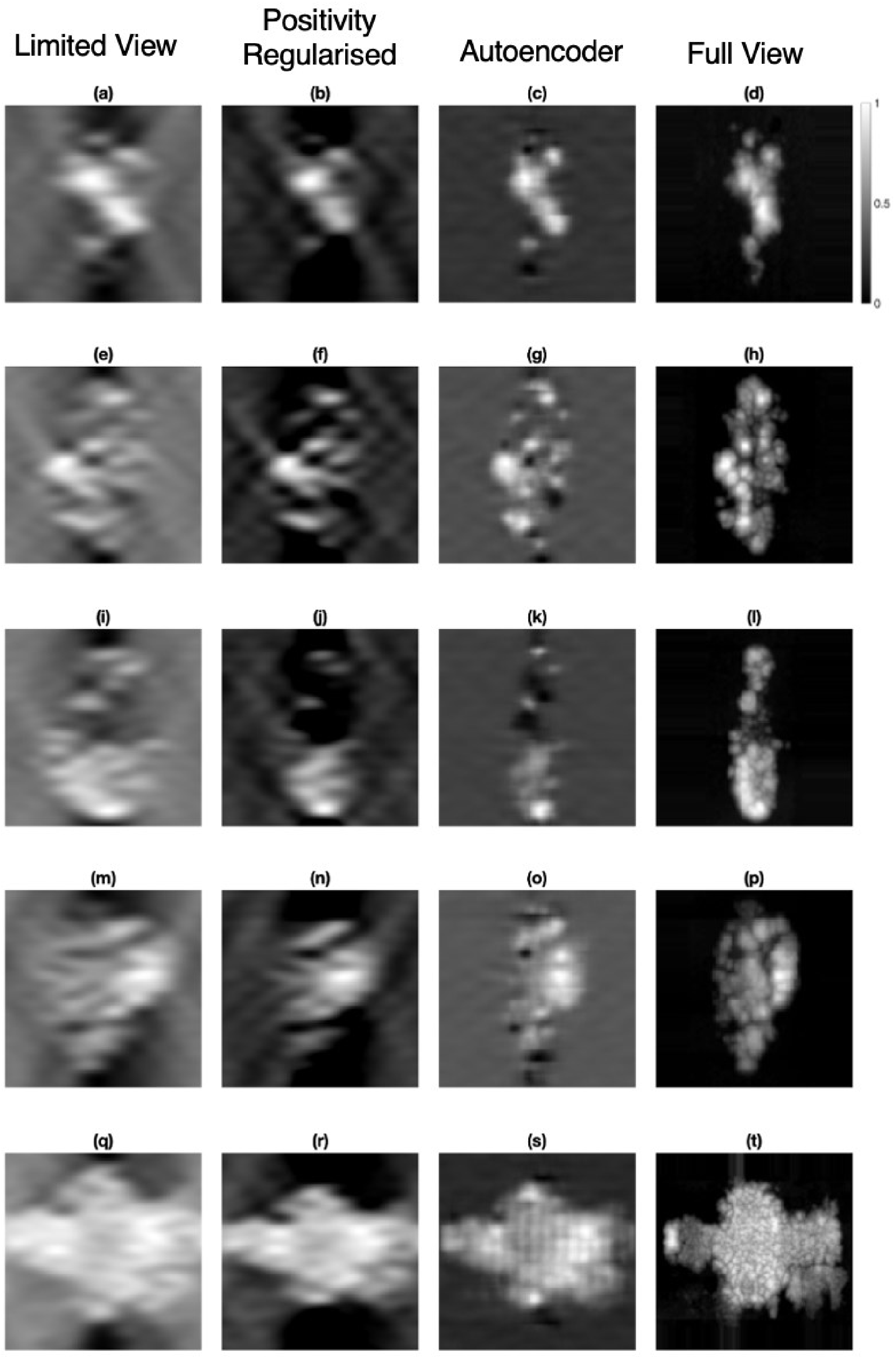


Figure 83. Experimental corrosion patches 6 to 10. Original limited view registration (a, e, i, m, q), image processed using the positivity regularisation method (b, f, j, n, r), output of the autoencoder (c, g, k, o, s) and full view ground truth (d, h, l, p, t).

Table 9. RMSE and MAXAE for the limited view (LV) registration of experimentally acquired corrosion thickness maps compared to the positivity regularised (PR) registration and the output of the autoencoder (AE). The "Diff." columns report the differences between PR and AE, with the cells in pink background highlighting the instances where the metric of PR surpassed that of AE. The two instances where the metric of LV surpassed that of AE are indicated in pink.

	RMSE (proportion of max wall loss)				MAXAE (proportion of max wall loss)			
	LV	PR	AE	Diff.	LV	PR	AE	Diff.
Figure 82(a-d)	0.20	0.09	0.06	0.03	0.98	0.42	0.28	0.14
Figure 82(e-h)	0.39	0.20	0.29	-0.08	0.98	0.61	0.49	0.13
Figure 82(i-l)	0.33	0.20	0.19	0.01	0.76	0.73	0.87	-0.13
Figure 82(m-p)	0.26	0.10	0.15	-0.05	0.78	0.39	0.35	0.04
Figure 82(q-t)	0.37	0.15	0.39	-0.24	0.93	0.66	0.67	-0.02
Figure 83(a-d)	0.29	0.11	0.16	-0.05	0.83	0.50	0.43	0.06
Figure 83(e-h)	0.37	0.14	0.25	-0.11	0.97	0.73	0.65	0.08
Figure 83(i-l)	0.37	0.14	0.21	-0.07	0.82	0.55	0.52	0.03
Figure 83(m-p)	0.32	0.15	0.27	-0.13	0.96	0.61	0.54	0.07
Figure 83(q-t)	0.37	0.16	0.15	0.01	0.91	0.82	0.58	0.24

It is notable that the corrosion patches considered in this section are more complicated than the training dataset used, which is likely to have some impact on the performance. It would be ideal to train with real corrosion patches, however, as with many problems, particularly in non-destructive testing, data scarcity is a significant challenge. It may be possible, however, to synthesise more representative corrosion patches and/or utilise data augmentation approaches, which may improve this.

7.5 Conclusion

This work proposes a novel approach to a limited view registration compensation algorithm for use in the NDT context. This is based on the implementation of a fully convolutional autoencoder machine learning architecture. The experimental data scarcity problem was circumvented by the usage of a fully artificial training dataset corrupted with a background uniform noise. The resultant trained model was tested both on an artificial testing dataset drawn from the same distribution as the training set, and, more importantly, on a set of ten real thickness maps of corrosion patches.

The trained autoencoder was first shown to perform very well in terms of both RMSE and MAXAE when tested on the artificial testing dataset. However, the truly important finding is the fact that the algorithm generalises well to real corrosion patches. The visual inspection of the results on all available experimental samples reveals that the proposed ML model improves the contrast and focus of the images when compared to both the original limited view registrations and to their processing via a state-of-the-art positivity regularisation routine. The inadequacy of the numeric metric is best presented by Figure 82(q-t). This example by MaxAE metric is the worst of the examples with positivity regularisation significantly outperforming AE. However, upon inspection it is clear that AE was able to horizontally localise the corrosion patch significantly better than regularisation. The improvements are also generally confirmed when looking at the chosen metrics, with the autoencoder's outputs outperforming the limited view registrations in 18 out of 20 cases, with an average error reduction of 35 and 40% in terms of RMSE and MAXAE, respectively. When the same metrics are used to compare the results of the autoencoder against those of the positivity regulariser, a general superiority of the former in terms of MAXAE is noted, though the opposite is true in terms of RMSE. Nevertheless, a further visual inspection of the results indicates that this is essentially due to a not-so-critical tendency of the autoencoder to achieve a higher background mismatch than the positivity regularisation approach, while the more relevant parts of the image including the reductions in thickness rendered by the autoencoder are much sharper and more accurate than those given by the regulariser.

Finally, the algorithm was tested on artificial geometric thickness maps designed to test the limitations of the limited view algorithm. These examples were very far from the training data distribution. The test shows that this is the only application in which the AE processing yielded adverse results, with the original limited view registration closer to the ground truth. This result serves as a reminder, that the proposed autoencoder, similarly to most ML applications, is very dependent on the quality of the input data. This work proposes two approaches for limiting the impact of this issue. The first involves an expert-led process of improving the training dataset to successfully simulate every input type a limited view UT could encounter. This faces its own set of issues, such as the work intensiveness and the lack of guarantee that all scenarios are indeed covered. The second approach involves building on the existing work to detect out-of-distribution samples and flag them before passing through the autoencoder.

8 Conclusion

This chapter is comprised of the recap of the project described in the thesis, a summary of the main findings and the future work recommended for the development of the project.

8.1.1 Summary

This project had a broad remit in exploring the methods in which machine learning can be utilised to enhance the guided wave testing of pipelines. The industrial motivation was that the guided wave testing is very dependent on the performance of the human inspector, introducing the uncertainty into the results of the method. Machine Learning offers potential solutions to the issue by offering a second-marking capacity to inspectors.

Chapter 1 introduced guided wave testing, its industrial use cases, the inspection protocol, and the motivation for the project. Additionally, it introduced the industrial partner of this project, Guided Ultrasonics Limited, who are the source of the in-service inspection data.

Chapter 2 provides the theoretical background necessary to fully understand the thesis. As the project has two distinct constituent parts, the theoretical background is similarly split into sections. At first it introduces the mechanics of wave propagation as the necessary step to explain the finite element modelling, being a source of most of the data used in this work. It follows with the artificial focusing methods which were trialled as an attempt to inject physics-based knowledge into the machine learning paradigm. It also briefly introduces guided wave transducers, as the hardware dictates the ultimate capability of the testing method. In the second part, machine learning is broadly introduced. While this work used relatively simple ML approaches, the section is meant to provide the reader with the knowledge necessary to critique the approaches. It provides the historical outline of ML development, the common ML paradigms of supervised and unsupervised learning as well as introduces the recent advances in generative ML and the social and ethical issues with AI fairness. Furthermore, it derives the operations of the deep learning architectures and provides the mathematical background for the understanding of the constituent parts of the ML model.

Chapter 3 introduces the datasets used for the purposes of the work. It describes the data pipelines for both the simulated and the real data. Importantly, it discusses the different input formats dependent on the source of data, decides on the common output format and describes the transformations necessary. Furthermore, it deals with the problems encountered when working with the data, both real and simulated, most importantly, the inherent differences between the real and the simulated data, stemming from the lack of ability to fully simulate the real world. The chapter concludes with the description and parameters of the datasets used in the rest of the work.

Chapter 4 describes the design of the major part of this project – a decision-making ML model designed to assist the inspectors. It starts by introducing the metrics used for the assessment of ML approaches, from the basic ones, through their derivatives like AUROC and finishing by developing a bespoke metric of FPR@1TPR designed specifically for NDT. It goes through the descriptions of the architectures used in the work, their specific implementations and adjustments made due to the characteristics of the problem or the data.

Subsequently, Chapter 5 it presents the results of the experiments conducted to select the best parameters for the ML training. The final part of this chapter presents the results of a major study testing the performance of ML on GWT data. The first experiments were conducted on the simulations assessing the impact of the data processing. The testing on the in-service data followed, initially assessing the performance in defect indication detection. With the results on this problem were unsatisfactory, the question was broadened to feature detection, where the performance improved to industry-standard level. This path was therefore investigated further, with a full-scale study designed to select the best architecture and investigate the impact of increasing the number of training data points on the performance of the model. Finally, the samples consistently misclassified by the models are investigated to inform the future development of the ML approaches.

Chapter 6 deals with the modern ML topics, which have been briefly investigated during this project. It begins with the presentation of the generative adversarial network designed to produce realistic-looking GWT samples. It describes the considerations when building such a network and follows with the presentation of the generated results in both time and frequency domain. The second part of this chapter touches on the issues outside of the technical development of the ML tools, but relevant to the broader issue of widespread utilisation of ML methods in the context of NDT. Most importantly it deals with the problems of trust and reliability and the various method they can be assured. Finally, a development enabling any advanced machine learning research is the broadening of access to good quality training data. This issue has been discussed at length in chapter 3.4 concerning the data issues faced in this work, but new and advanced machine learning architectures are more data-hungry than ever. The costs of monitoring decreasing compared to the costs of failure will make the data significantly more abundant. It is a clear interest of the research community to make data accessible. However, this is generally acquired by the service providers, hardware manufacturers and end users. Those actors have an interest in keeping the data private, making it necessary to develop a sharing scheme allowing the owners to open their data to NDT machine learning research without revealing trade secrets. Open data facilitates the stated aim of NDT: preventing as many in-service failures as possible. Thus, a public data regime should be developed by a broad confederation of industry actors to ensure that everyone has a stake and participates in the benefits.

Chapter 7 introduces the standalone work conducted to assess the usefulness of ML in tackling other issues in ultrasonic NDT. This work has introduced a content-agnostic, fast, and lightweight algorithm compensating for the limited view transducer configuration. The architecture is based on a fully convolutional autoencoder. The network was trained on fully artificial data and tested on a small, but illustrative dataset of laser-scanned corrosion patches to assess its generalisation capacity. It improves the quality of the images when measured as maximum absolute error and RMSE. The work compared the performance of the ML architecture to that of a conventional non-ML algorithm – positivity regularisation. While the ML approach comparatively underperformed on RMSE, the visual inspection of the results shows that it leads to a significantly closer-to-truth images.

8.1.2 Key Contributions

- An ML algorithm has been developed that assesses the standard guided wave inspection A-scan and decides whether the trace contains a feature of interest or not. This algorithm performs at the true positive rate (probability of detection) of 99.7% with the false positive rate (probability of false alarm) between 1.5 and 4.1%. This is presented in Figure 68 Section 5.4.
- VGG-Net has been proven to reach better performance than U-Net both measured using AUROC and FPR@1TPR. Additionally, the training of VGG-Net is quicker and more stable than of a U-Net. This points the NDT ML research towards simpler architectures. This is presented in Figure 67 and Figure 68 Section 5.4.

- Pre-training the ML algorithm using FEM simulated data followed by fine-tuning on the real in-service data makes the model perform better than using just the real data. This points to the potential gains from developing better finite element models. This is presented in Figure 65 Section 5.2.1 and Figure 67 Section 5.4.
- An ML algorithm for the analysis of GWT traces can be trained on as few as 1000 real traces, including 250 examples of the positive class (Figure 68 Section 5.4.). This has been proven for feature detection, but the follow-up study showed that defect indications are not overrepresented in the misclassified set (Table 7, Figure 69, and Figure 70 Section 5.5).
- The generative adversarial network is able to generate traces containing just the backwall signal which appear realistic, but simulating smaller defect indication signals was not as successful. This behaviour is clear both in time and frequency domains (Figure 73, and Figure 74 Section 6.1.2). It is likely that a significantly larger dataset would be needed for a well performing algorithm in that category. Thus, it should be concluded that generative learning should be used in more restricted ways, such as defect indication injection.
- ML based limited view compensation algorithm performs better than current state of the art on meaningful metrics (Figure 82, Figure 83, and Table 9 Section 7.4.3). This can be trained on purely simulated data and generalise well to unseen real data.
- Machine learning model performance on the simulated data does not necessarily translate to the real data, thus any developed models need to be tested on data gathered in-service to ensure their suitability (Figure 63, Figure 64, and Figure 65 Section 5.2).
- The main issue faced by this work was the difficulty in the data processing. This points to the directions of further research in the establishment of a data repository that could be used both to develop and to validate automated decision-making methods. This is showcased by the complexity of the data processing described in Section 3.2.

8.1.3 Future Work

This work has developed a proof-of-concept algorithm for automated defect indication detection using guided wave testing. The model in this work was tested using the data gathered in operational conditions but manually processed and bespoke designed in research conditions, making it a TRL 5/6 technology; thus, the most important follow-up should be raising the TRL to levels 8/9. This type of work is typically undertaken outside of the academic setting. In order to raise the TRL the technology needs to be included into a robust pipeline:

1. Design data format for inspection data.
2. Collect and process the inspection data into a known and constant format.
3. Receive the model predictions. Save the prediction in a way the clearly links it to the original data.
4. Design a continuous learning pipeline that allows for model re-training based on field inspection results.
5. Design a robust periodic validation procedure to guard against the model drift (changing predictions based on the new training data).

Overarchingly ensure robust inspector training in the use of the new tool to facilitate adoption.

Secondly, the model itself can be refined and tested to assess the potential gains in the performance. Some of the parameters that can be changed and hold some promise for the improvement in the performance are:

- Changing size of the input data or using multiple input data sizes – this would allow the capture and classification of larger features.
- Experimenting with a variety of loss types, especially hinge or focal loss.
- Running a large-scale grid search over the parameters such as the receptive field size, the number of layers, type of pooling.
- Experimenting with other modern ML architectures such as EfficientNet.
- Experiment with pre-trained ML architectures, promising candidates would be environmental sound classification or keyword recognition networks.

All the mentioned developments are less suited to academic research and more to the commercial environment.

From the research perspective, the clear next stage is extending the ML capability to defect indication, rather than feature, detection. This could be pursued in a couple of ways:

- Develop a real experimental dataset of 250+ defect indications and apply the lessons learnt in this research.

- Improve the simulation capacity so the finite element models capture behaviour within the real data better, thus reducing the need for the in-service data.
- Implement a two-stage ML process, first finding the features and then classifying between a benign one and a defect indication. This should use engineered features and explainable architectures, as it is clearly a safety-critical stage.
- Design a multi-agent ML pipeline with separate models trained to automate or assist in specific inspector tasks (weld identification, DAC setting, frequency sweep etc.)

The generative learning approaches investigated were too data-hungry to be deployed directly as a method for acquiring realistic data. These could, however, be used to enhance the simulated data or to transform and inject the defects between samples. Furthermore, the modern GPT technology should be explored, especially its ability to generate sound waves, for the possibility of fine-tuning it for ultrasonic NDT area.

The limited view compensation generally performed very well and can be considered state-of-the art, however, it underperformed on RMSE compared to the conventional approach. This can be remedied with relatively little additional work. Additionally, the performance of the algorithm can probably be improved by employing a similar transfer learning approach as in the main defect indication detection work. This would only necessitate gathering a significantly larger dataset (at least a couple hundred examples) of corrosion thickness maps and implementing a two-stage training. This work would be suitable to either academic or industrial setting.

Finally, the problem of explainability and trust must be addressed. A full review of the explainability methods must be undertaken, with the candidates assessed for the suitability in NDT. This work needs to be performed in collaboration with industry bodies, as purely academic research does not have the profile to become a standard.

Bibliography

[1] BINDT, "BINDT," 21 3 2023. [Online]. Available: https://www.bindt.org/What-is-NDT/NDT-Method-Selector/.

[2] Wikipedia, "Guided Wave Testing," 2023. [Online]. Available: https://en.wikipedia.org/wiki/Guided_wave_testing. [Accessed 4 10 2023].

[3] W. P. &. S. S. U. David Russell, "Pigging in Pipeline Pre-Comissioning," Pigging Products and Services Association, 2005.

[4] M. Lowe and P. Cawley, "Long Range Guided Wave Inspection Usage – Current Commercial Capabilities and Research Directions," Guided Ultrasonics Ltd., 2006.

[5] globaldata.com, "Global Oil and Gas Pipelines Market Outlook, 2021-2025 – Capacity and Capital Expenditure Outlook with Details of All Operating and Planned Pipelines," 2022.

[6] D. Alleyne and P. Cawley, "The interaction of Lamb waves with defects," *IEEE Transactionson on Ultrasonics, Ferroelectrics, and Frequency Controlv,* vol. 39, no. 3, pp. 381-397, 1992.

[7] R. Pyle, R. Bevan, R. Hughes, R. Rachev, A. Ali and P. Wilcox, "Deep learning for ultrasonic crack characterization in NDE," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control,* vol. 68, no. 5, pp. 1854-1865, 2020.

[8] Guided Ultrasonics Ltd., "Guided Ultrasonics GUL Products, Screening," 2023. [Online]. Available: https://www.guided-ultrasonics.com/products/#!. [Accessed 8 Aug 2023].

[9] Guided Ultrasonics Ltd., "WavePro4 Software," 2023. [Online]. Available: https://www.guided-ultrasonics.com/product/wavepro4-software/. [Accessed 18 July 2023].

[10] Rayleigh, "On the free vibrations of an infinite plate of homogeneous isotropic elastic matter.," *Proceedings London Mathematical Society,* pp. 225-237, 1888-1889.

[11] H. Lamb, "On waves in an elastic plate," *Proceedings Royal Society,* pp. 114-128, 1916-1917.

[12] R. Stoneley, "Elastic waves at the surface of separation of two solids," *Royal Society Proceedings London,* pp. 416-428, 1924.

[13] Institute of Sound and Vibration Research, "Dispersive waves," 25 July 2023. [Online]. Available: https://resource.isvr.soton.ac.uk/spcg/tutorial/tutorial/Tutorial_files/Web-further-dispersive.htm.

[14] D. C. Gazis, "Three dimensional investigation of the propagation of waves in hollow circular cilinders.," *Journal of the Acoustical Society of America,* pp. 568-578, 1959.

[15] J. D. Achenbach, Wave Propagation in Elastic Solids, New York: North-Holland, 1984.

[16] B. N. Pavlakovic, M. J. Lowe, D. N. Alleyne and P. Cawley, "DISPERSE: A General Purpose Program for Creating Dispersion Curves.," in *Review of Progress in Quantitative Nondestructive Evaluation*, 1999.

[17] J. Rose, in *Ultrasonic Waves in Solid Media*, Cambridge, Cambridge University Press, 1999, pp. 103-107.

[18] M. G. Silk and K. P. Bainton, "The propagation in metal tubing of ultrasonic wave modes equivalent to Lamb waves," *Ultrasonics,* pp. 11-19, 1979.

[19] D. N. Alleyne, T. Vogt and P. Cawley, "The choice of torsional or longitudinal excitation in guided wave pipe inspection," in *Proceedings of 5th Iranian International NDT Conference*, Tehran, 2018.

[20] P. Huthwaite, "Accelerated finite element elastodynamic simulations using the GPU," in *Journal of Computational Physics*, 2014.

[21] S. Marburg, "Discretization requirements: How many elements per wavelength are necessary?," in *Computational Acoustics of Noise Propagation in Fluids-Finite and Boundary Element Methods* , Berlin, Heidelberg, Springer Berlin Heidelberg, 2008, pp. 309-332.

[22] M. B. Drozdz, Efficient finite element modelling of ultrasound waves in elastic media (Thesis)., London: Imperial College London (University of London), 2008.

[23] J. R. Pettit, A. Walker, P. Cawley and M. J. S. Lowe, "A Stiffness Reduction Method for efficient absorbtion of waves at boundaries for use in commercial Finite Element codes," *Ultrasonics,* pp. 1868-1879, 2014.

[24] K. J. Langenberger, M. Berger, T. Kreutter, K. Mayer and V. Schmitz, "Synthetic aperture focusing technique signal processing," *NDT International,* pp. 188-189, 1986.

[25] C. Holmes, B. W. Drinkwater and P. D. Wilcox, "Post-processing of the full matrix of ultrasonic transmit-receive array for non-destructive evaluation," *NDT&E International,* pp. 701-711, 2008.

[26] R. K. Rachev, P. D. Wilcox, A. Velichko and K. L. McAughey, "Plane Wave Imaging Techniques for Immersion Testing of Components With Nonplanar Surfaces," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control,* vol. 67, no. 7, pp. 1303-1316, 2020.

[27] J. Davies, F. Simonetti, M. Lowe and P. Cawley, "Review of Synthetically Focused Guided Wave Imaging Techniques With Application to Defect Sizing," in *AIP Conference Proceedings 820*, 2006.

[28] J. Davies and P. Cawley, "The Application of Synthetic Focusing for Imaging Crack-Like Defects in Pipelines Using Guided Waves," *IEEE Transacitions on Ultrasonics Ferroelectrics and Frequency Control,* pp. 759-771, 2009.

[29] A. Velichko and P. D. Wilcox, "Excitation and scattering of guided waves: Relationships between solutions for plates and pipes," *The Journal of the Acoustical Society of America,* vol. 125, no. 6, pp. 3623-3631, 2009.

[30] J. Davies and P. Cawley, "The Application of Synthetically Focused Imaging Techniques for High Resolution Guided Wave Pipe Inspection," 2007.

[31] H. Geesink and D. Meijer, "Mathematical Structure for Electromagnetic Frequencies that May Reflect Pilot Waves of Bohm's Implicate Order," *Journal of Modern Physics,* pp. 851-897, 2018.

[32] D. N. Alleyne and P. Cawley, "The excitation of Lamb waves in pipes using dry-coupled piezoelectric transducers," *Journal of Nondestructive Evaluation,* vol. 15, pp. 11-20, 1996.

[33] H. Kwun and A. Holt, "Feasibility of under-lagging corrosion detection in steel pipe using the magnetostrictive sensor technique," *NDT&E International,* vol. 28, no. 4, pp. 211-214, 1995.

[34] J. J. Niederhouser, M. Jaeger and M. Frenz, "Comparision of laser-induced and classical ultasound," San Jose, 2003.

[35] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review,* vol. 65, no. 6, p. 386, 1958.

[36] R. Bush and F. Mosteller, "A mathematical model for simple learning," *Psychological Review,* vol. 58, no. 5, p. 313, 1951.

[37] V. Vapnik and A. Y. Chervonenkis, "A class of algorithms for pattern recognition learning," *Avtomatika i Telemekhanika,* vol. 25, no. 6, pp. 937-945, 1964.

[38] K. Fukushima and S. Miyake, "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognition,* vol. 15, no. 6, pp. 455-469, 1982.

[39] D. Rumelhart, G. Hinton and R. Williams, "Learning representations by back-propagating errors," *nature,* vol. 323, no. 6088, pp. 533-536, 1986.

[40] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Google Inc., 2004.

[41] P. Vingelmann and F. Fitzek, "CUDA," NVIDIA, 2007.

[42] Y. LeCun, Y. Bengio and G. Hinton, "Deep Learning," *nature,* vol. 521, no. 7553, pp. 436-444, 2015.

[43] I. Goodfellow, J. Abadie-Pouget, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative Adversarial Nets," 2014.

[44] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel and Y. Bengio, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention," 2015.

[45] V. Ashish, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser and I. Polosukhin, "Attention is all you need," 2017.

[46] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskver, "Language Models are Unsupervised Multitask Learners," OpenAI, San Francisco, 2008.

[47] OpenAI, "ChatGPT," 2023. [Online]. Available: https://chat.openai.com. [Accessed 22 Aug 2023].

[48] L. P. Kaelbling, M. L. Littman and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research,* vol. 4, pp. 237-285, 1996.

[49] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglu, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach and Kavucuogl, "Mastering the game of Go with deep neural networks and tree search," *nature,* vol. 529, pp. 484-489, 2016.

[50] D. Silver, J. Shrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicarp, F. Hui, L. Sifre, G. van den Driessche and T. Graepel, "Mastering the game of Go without human knowledge," *nature,* no. 550, pp. 354-359, 2017.

[51] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicarp, K. Simonyan and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science,* vol. 362, no. 6419, pp. 1140-1144, 2018.

[52] M. W. Berry, A. Mohamed and B. W. Yap, Supervised and unsupervised learning for data science, Springer Nature, 2019.

[53] Nvidia, "SuperVize Me: What’s the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning?," 2018. [Online]. Available: https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/. [Accessed 5 10 2023].

[54] Google Inc., "reCaptcha," 2023. [Online]. Available: https://www.google.com/recaptcha/about/. [Accessed 5 10 2023].

[55] Kaggle, "House Prices - Advanced Regression Techniques," 2024. [Online]. Available: https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques. [Accessed 14 July 2024].

[56] IBM, "What is Unsupervised Learning?," 2024. [Online]. Available: https://www.ibm.com/topics/unsupervised-learning. [Accessed 14 July 2024].

[57] Coursera, "Machine Learning Specialization," 2024. [Online]. Available: https://www.coursera.org/specializations/machine-learning-introduction. [Accessed 13 July 2024].

[58] Coursera, "Deep Learning Specialisation," 2024. [Online]. Available: https://www.coursera.org/specializations/deep-learning. [Accessed 13 July 2024].

[59] K. Cooper, "What does weight mean in terms of neural network.," 2020. [Online]. Available: https://www.quora.com/What-does-weight-mean-in-terms-of-neural-networks. [Accessed 5 10 2023].

[60] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift.," in *International Conference on Machine Learning*, 2015.

[61] K. Palczynski, "Application of convolutional neuron network for image processing and interpretation," *Telecommunication and Electronics 23,* no. 23, pp. 5-21, 2019.

[62] A. Anwar, "What is transposed convolutional layer?," 2020. [Online]. Available: https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11. [Accessed 5 10 2023].

[63] O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *MICCAI 2015,* pp. 234-241, 2015.

[64] Computer Science Wiki, "Maxpool Sample," 2018. [Online]. Available: https://computersciencewiki.org/index.php/File:MaxpoolSample2.png. [Accessed 5 10 2023].

[65] Dive Into Deep Learning, "Multilayer Perceptrons," 2023. [Online]. Available: https://d2l.ai/chapter_multilayer-perceptrons/mlp.html. [Accessed 5 10 2023].

[66] T. Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollar, "Focal loss for dense object detection," 2017.

[67] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," 2015.

[68] N. Qian, “On the Momentum term in gradient descent learning algorithms,” *Neural Networks,* pp. 145-151, 1999.

[69] Papers with Code, "SGD with Momentum," 2023. [Online]. Available: https://paperswithcode.com/method/sgd-with-momentum. [Accessed 5 10 2023].

[70] D. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimisation," in *ICLR*, 2015.

[71] S. Ruder, "Review of gradient descent optimization algorithms," arXiv:1609.04747 [cs.LG] , 2016.

[72] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative Adversarial Nets," in *NeurIPS*, 2014.

[73] D. P. Kingma and M. Welling, "An Introduction to Variational Autoencoders," *Foundations and Trends in Machine Learning,* pp. 307-392, 2019.

[74] L. A. Gatys, A. S. Ecker and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," in *CVPR*, 2016.

[75] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation,* pp. 1735-1780, 1997.

[76] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomes, L. Kaiser and I. Poposukhin, "Attention Is All You Need," in *NeurIPS*, 2017.

[77] K. Weiss, T. M. Khoshgoftaar and D. Wang, "A survey of transfer learning," *Journal of Bog Data,* vol. 3, 2016.

[78] M. T. Ribeiro, S. Singh and C. Guestrin, ""Why should I trust You?" Explaining the predictions of any classifier.," in *Proceedings of the 22nd ACM SIGKKD*, 2016.

[79] S. M. Lundberg and S. Lee, "A Unified Approach to interpreting model predictions," in *NeurIPS*, 2017.

[80] B. H. Zhang, B. Lemoine and M. Mitchell, "Mitigating Unwanted Biases with Adversarial Learning," in *AIES*, New Orleans, 2018.

[81] T. Le Quy, A. Roy, V. Iosifidis, W. Zhang and E. Ntousi, "A survey on datasets for fairness-aware machine learning," *WIREs Data Mining and Knowledge Discovery,* 2022.

[82] W. Cukierski, "Dogs vs. Cats. Kaggle.," 2013. [Online]. Available: https://kaggle.com/competitions/dogs-vs-cats.

[83] S. Mariani, S. Heinlein and P. Cawley, "Location Specific Temperature Compensation of Guided Wave Signals in Structural Health Monitoring," *EEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control,* pp. 146-157, 2020.

[84] ASTM International, "ASTM E2339-21: Standard Practice for Digital Imaging and Communication in Nondestructive Evaluation (DICONDE)," ASTM International, West Conshohocken, PA, 2022.

[85] S. Haykin, Neural Networks: a comprehensive foundation., Prentice Hall PTR, 1994.

[86] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2015.

[87] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg and L. Fei-Feu, "ImageNet large scale visual recognition challenge.," 2014.

[88] L. Yang, H. Wang, B. Huo, F. Li and L. Yanhong, "An automatic welding defect location algorithm based on deep learning," *NDT & E International,* vol. 120, 2021.

[89] B. C. F. Oliveira, A. A. Seibert, V. K. Borges, A. Albertazzi and R. H. Schmitt, "Employing a U-net convolutional neural network for segmenting impact damages in optical lock-in thermography images of CFRP plates," *Nondestructive Testing and Evaluation,* vol. 36, no. 4, pp. 440-458, 2021.

[90] Q. Luo, B. Gao, W. L. Woo and Y. Yang, "Temporal and spatial deep learning network for infrared thermal defect detection," *NDT & E International,* vol. 108, 2019.

[91] C. Wu, Y. Zou and Z. Yang, "U-GAN: Generative Adversarial Networks with U-Net for Retinal Vessel Segmentation," *2019 14th International Conference on Computer Science & Education (ICCSE),* pp. 642-646, 2019.

[92] E. Schonfeld, B. Schiele and A. Khoreva, "A U-Net Based Discriminator for Generative Adversarial Networks," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 8207-8216, 2020.

[93] R. Komatsu and T. Gonsalves, "Comparing U-Net Based Models for Denoising Color Images," *AI,* vol. 1, no. 4, pp. 465-486, 2020.

[94] R. Giri, U. Isik and A. Krishnaswamy, "Attention Wave-U-Net for Speech Enhancement," *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA),* pp. 249-253, 2019.

[95] yolov8.org, "Yolov8 Architecture," 2024. [Online]. Available: https://yolov8.org/yolov8-architecture/. [Accessed 14 July 2024].

[96] T. G. Dietterich, "Ensemble Methods in machine learning.," Berlin, Heidelbeg, 2000.

[97] G. Vilone and L. Longo, "Notions of explainability and evaluation approaches for explainable artificial intelligence," *Information Fusion,* vol. 76, pp. 89-106, 2001.

[98] Google Inc., "Welcome to Colab," Google Inc., [Online]. Available: https://colab.research.google.com/. [Accessed 4 Jan 2023].

[99] R. J. Pyle, R. R. Hughes and P. D. Wilcox, "Interpretable and Explainable Machine Learning for Ultrasonic Defect Sizing," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control,* vol. 70, no. 4, pp. 277-290, 2023.

[100] The American Society for Nondestructive Testing, "ASNT Certification Services LLT," 2020. [Online]. Available: https://certification.asnt.org. [Accessed 4 Jan 2023].

[101] The British Institute of Non-Destructive Testing, 2013. [Online]. Available: https://www.bindt.org/Certification/. [Accessed 4 Jan 2023].

[102] European Network for Inspection and Qualification, "ENIQ Report No. 65, Technical Area 8," 2021.

[103] A. Devaney, "Geophysical diffraction tomography," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 1, pp. 3-13, 1984.

[104] A. Andersen, "Algebraic reconstruction in CT from limited views," *IEEE Transactions on Medical Imaging,* vol. 8, no. 1, pp. 50-55, 1989.

[105] A. Delaney and Y. Bresler, "Globally convergent edge-preserving regularized reconstruction: An application to limited-angle tomography"," *IEEE Transactions on Image Processing,* vol. 7, no. 2, pp. 204-221, 1998.

[106] R. Lewitt, "Processing of incomplete measurement data in computed tomography," *Medical Physics,* vol. 6, no. 5, pp. 412-417, 1979.

[107] A. Andersen, "Algebraic reconstruction in CT from limited views," *IEEE Transactions on Medical Imaging,* vol. 8, no. 1, pp. 50-55, 1989.

[108] P. Huthwaite, A. A. Zwiebel and F. Simonetti, "A new regularization technique for limited-view sound-speed imaging," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control,* vol. 60, no. 3, pp. 603-613, 2013.

[109] G. Wang, C. Y. Jong and B. De Man, "Deep learning for tomographic image reconstruction," *Nature Machine Intelligence,* no. 2, pp. 737-748, 2020.

[110] Z. Zhou, Y. Wang, J. Yu, Y. Guo, W. Guo and Y. Qi, "High spatial–temporal resolution reconstruction of plane-wave ultrasound images with a multichannel multiscale convolutional neural network," *IEEE transactions on ultrasonics, ferroelectrics, and frequency control,* vol. 65, no. 11, pp. 1983-1996, 2018.

[111] J. Sun, H. Li and Z. Xu, "Deep ADMM-Net for compressive sensing MRI," in *Advances in Neural Information Processing Systems*, 2016.

[112] B. Zhu, J. Liu, S. F. Cauley, B. R. Rosen and M. S. Rosen, "Image reconstruction by domain-transform manifold learning," *Nature,* vol. 555, pp. 487-492, 2018.

[113] M. Born and E. Wolf, Principles of Optics 7th ed., Cambridge: Cambridge University Press, 1999.

[114] Y. M. Wang and W. C. Chew, "An iterative solution of the two-dimensional electromagnetic inverse scattering problem," *International Journal of Imaging Systems and Technology,* vol. 1, no. 1, pp. 100-108, 1989.

[115] W. C. Chew and Y. M. Wang, "Reconstruction of two-dimensional permittivity distribution using the distorted Born iterative method," *IEEE Transactions on Medical Imaging,* vol. 9, no. 2, pp. 218-225, 1990.

[116] P. Huthwaite and F. Simonetti, "High-resolution imaging without iteration: a fast and robust method for breast ultrasound tomography," *The Journal of the Acoustical Society of America,* vol. 130, no. 3, pp. 1721-1734, 2011.

[117] A. J. Devaney, "A Filtered Backpropagation Algorithm for Diffraction Tomography," *Ultrasonic Imaging,* vol. 4, no. 4, 1982.

[118] F. Simonetti and L. Huang, "From beamforming to diffraction tomography," *Journal of Applied Physics,* vol. 103, no. 10, 2008.

[119] M. Slaney and A. Kak, Principles of Computerized Tomographic Imaging, New York: IEEE Press, 1999.

[120] G. Wang, C. Y. Jong and B. De Man, "Deep learning for tomographic reconstruction," *Nature Machine Intelligence,* vol. 2, pp. 737-748, 2020.

[121] R. Yang and Y. Yu, "Artificial convolutional neural network in object detection and semantic segmentation for medical imaging analysis," *Frontiers in oncology,* vol. 11, no. 638182, 2021.

[122] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

[123] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[124] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *EEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, 2016.

[125] O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, 2015.

[126] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, 2019.

[127] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[128] N. Davoudi, X. Dean-Ben and D. Razansky,, "Deep learning optoacoustic tomography with sparse data," *Nature Machine Intelligence,* vol. 1, pp. 453-460, 2019.

[129] S. Guan, A. A. Khan, S. Sikdar and P. V. Chitnis, "Limited-View and Sparse Photoacoustic Tomography for Neuroimaging with Deep Learning," *Scientific Reports,* vol. 10, p. 8510, 2020.

[130] E. Nehme, L. E. Weiss, T. Michaeli and Y. Shechtman, "Deep-STORM: super-resolution single-molecule microscopy by deep learning," *Optica,* vol. 5, no. 4, pp. 458-464, 2018.

[131] Y. Rivenson, Z. Gorocs, H. Gunaydin, Y. Zhang, H. Wang and A. Ozcan, "Deep learning microscopy," *Optica,* vol. 4, no. 11, pp. 1437-1443, 2017.

[132] K. H. Jin, M. T. McCann, E. Froustey and M. Unser, "Deep convolutional neural network for inverse problems in imaging," *IEEE Trans. Image Process.,* vol. 26, pp. 4509-4522, 2017.

[133] K. Kwon, D. Kim and H. Park, "A parallel MR imaging method using multilayer perceptron.," *Madical Physics,* vol. 44, pp. 6209-6224, 2017.

[134] Y. H. Yoon, S. Khan, J. Huh and J. C. Ye, "Efficient B-mode ultrasound image reconstruction from sub-sampled RF data using deep learning.," *IEEE Transactions on Tedical Imaging,* no. 38, pp. 325-336, 2018.

[135] Z.-Y. Zhu, T. Park, P. Isola and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks.," in *Proceedings of IEEE International Conference on Computer Vision*, 2017.

[136] E. Kang, H. J. Koo, D. H. Yang, J. B. Seo and J. C. Ye, "Cycle-consistend adversarial denoising network for multiphase coronary CT angiography.," *Medical Physics,* vol. 46, pp. 550-562, 2019.

[137] C.-Y. Liou, W.-C. Cheng, J.-W. Liou and D.-R. Liou, "Autoencoder for words," *Neurocomputing,* vol. 139, pp. 84-96, 2014.

[138] Mathworks Inc., "imresize," 22 Jan 2023. [Online]. Available: https://www.mathworks.com/help/matlab/ref/imresize.html. [Accessed 22 Jan 2023].

[139] G. Aurelien, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, O'Reilly Media, 2019.

[140] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," San Diego, 2015.

[141] Meta AI, "SGD with Momentum," 2023. [Online]. Available: https://paperswithcode.com/method/sgd-with-momentum. [Accessed 1 November 2023].

[142] Google Inc. , "Keras Tuner," 22 Jan 2023. [Online]. Available: https://www.tensorflow.org/tutorials/keras/keras_tuner. [Accessed 22 Jan 2023].

[143] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh and A. Talwalkar, "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization," *Journal of Machine Learning Research,* no. 18, pp. 1-52, 2018.

[144] R. J. Pyle, R. R. Hughes and P. D. Wilcox, "Interpretable and Explainable Machine Learning for Ultrasonic Defect Sizing," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control,* vol. 70, no. 4, pp. 277-290, 2023.

[145] A. Galvagni and P. Cawley, "GUIDED WAVE PERMANENTLY INSTALLED PIPELINE MONITORING SYSTEM, 1430," in *AIP Conference Proceedings*, 2012.

[146] A. Croxford, J. Moll, P. D. Wilcox and J. E. Michaels, "Efficient temperature compensation strategies for guided wave structural health monitoring," *Ultrosonics,* pp. 517-528, 2010.

[147] A. C. S. Douglass and J. B. Harley, "Dynamic Time Warping Temperature Compensation for Guided Wave Structural Health Monitoring," *EEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control,* pp. 851-861, 2018.

[148] Y. Gavrilova, "Convolutional Neural Networks for Beginners," 2021. [Online]. Available: https://serokell.io/blog/introduction-to-convolutional-neural-networks. [Accessed 5 10 2023].

List of Publications

(1) M. Mroszczak, R.E. Jones, P. Huthwaite, S. Mariani, “Transfer Learning in Guided Wave Testing of Pipes” (In review)

(2) M. Mroszczak, P. Huthwaite, S. Mariani, “Improved limited view ultrasound tomography via machine learning” (In review)