DEEP LEARNING IN ULTRASONIC WAVE INVERSION FOR THIN COATINGS

A Master's Thesis
Presented to
The Academic Faculty

By

Maximilian Schmitz

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Engineering Science and Mechanics
in the
School of Civil and Environmental Engineering
College of Engineering

Georgia Institute of Technology

May 2022

DEEP LEARNING IN ULTRASONIC WAVE INVERSION FOR THIN COATINGS

Thesis committee:

Prof. Laurence J. Jacobs
Advisor
School of Civil and Environmental Engineering
Georgia Institute of Technology

Prof. Christine Valle
George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Dr. Jin-Yeon Kim
School of Civil and Environmental Engineering
Georgia Institute of Technology

Date approved: January 14, 2022

ACKNOWLEDGMENTS

The first person I want to thank is my advisor Prof. Laurence J. Jacobs. My hardly expressible gratitude goes to him for all the help and support he gave me. Dr. Jacobs has been an the best advisor I could think of as well as the best friend possible, whose help reached far beyond his function as an advisor in all situations. I cannot say enough about what we have been through together.

My biggest thanks go to Dr. Jin-Yeon Kim who was a great advisor and friend. I am very grateful for all his suggestions and all the interesting discussions. I would like to thank Prof. Christine Valle for agreeing and serving on my thesis committee. I also would like to thank Remi Dingreville and Ryan Alberdi from Sandia National Laboratories for our fruitful conversations. I have special thanks to Prof. Michael Hanss and Dominik Hose from the University of Stuttgart for giving me the opportunity to study at the Georgia Institute of Technology.

Last but not least, I thank my family and friends who made this work possible for supporting me throughout my time at Georgia Tech.

The author gratefully acknowledges the support of the Baden-Württemberg-Stipendium which partially funded this program. This research was supported in part through research cyberinfrastructure resources and services provided by the Partnership for an Advanced Computing Environment (PACE) at the Georgia Institute of Technology, Atlanta, Georgia, USA.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This research focuses on the use of machine and deep learning to non-destructively characterize the quality of a coating in a layered system in terms of thickness and uniformness. The coating's parameters (thickness and uniformness) are evaluated using dispersion curves.

To obtain these dispersion curves in the first step, this work uses a computational finite element analysis (FEA) model to obtain dispersion data for coated specimen with different thicknesses. The simulated FEA data is then processed via a two-dimensional Fourier transform to obtain a frequency - wave number relation. This representation is a dispersion curve. Dispersion curves are characteristic and depend on the coating thickness and uniformness.

Simulated dispersion curves for various coatings are obtained. They are then further processed and a feature representation for each dispersion curve is extracted. Those extracted features are then fed into machine learning classifiers which allow a thickness classification.

The above-described machine learning procedure is limited to classifying the thickness of a uniform coating. To classify if a non-uniformess is present, deep learning steps in. To obtain information about if a coating is uniform, two convolutional neural network architectures are used. Both networks are evaluated and tested and recommendations on their use are given.

CHAPTER 1
INTRODUCTION

Measuring the unknown thickness of a coating of a specimen using ultrasonic waves via an inversion scheme has been a part of nondestructive evaluation for a long time [1]. Until now, a profound understanding of the underlying physics and a physics-based model of the underlying problem was necessary for those approaches. Previous research use simulated or experimentally measured data and compare this data to a theoretical model like [2], or use a scheme based on the Global Matrix Method as done by [3, 4].

This research proposes an approach to invert to the thickness of a coated specimen requiring only sufficient training data of a given system. This way, the physics-based model is surrogated by a data-driven machine learning and deep learning model, respectively.

Deep learning methods for wave inversion have been around in geophysics and geoscience, but to the knowledge of the author, have not yet been applied in a similar context to nondestructive evaluation. [5] describes different use cases for machine learning in seismology, while [6] developed a deep network consisting of encoder and decoder to obtain subsurface velocity structures for subsurface characterization in geoscience. Integrating known physical relationships into the training process has been conducted by [7, 8].

After this introduction, chapter 2 of this work will be dedicated to understanding the fundamental concepts of the given forward and inverse problem. This includes wave propagation in solids, finite element analysis, and signal processing for the forward problem as well as deep learning for the inverse problem. Chapter 3 then describes in-depth the setup for the finite element analysis simulations conducted and

discusses results of the forward problem, such that chapter 4 can build upon that such that the feature extraction and the machine learning-based thickness inversion can be described. Chapter 5 describes the uniformness inversion based on deep learning and chapter 6 concludes this work.

CHAPTER 2
INTRODUCTION AND THEORETICAL BACKGROUND

In this research, two major problems of material science are discussed: the forward problem and the inverse problem. While the forward problem assumes that the material properties like Young's modulus and density are known such that the state of the solid, e.g. displacement, stress, strain, can be calculated, the inverse problem starts the state of the solid and tries to obtain information about the causing material properties of the material.

The first section of this chapter is devoted to an introduction to wave propagation, followed by a description of the foundations of finite element analysis in the second section. The third section gives an intuition about the signal processing used in this work, i.e. the Fourier transform, and the last section of this chapter explains some fundamental concepts of deep learning.

2.1 Wave Propagation

This section is intended to provide the reader with a brief introduction to wave propagation in solid media. It provides a background to understand the topics covered in this work - but is not intended to give a deep and complete understanding of wave propagation in general. The reader is encouraged to have a look at [9, 10, 11] for a more detailed explanation of the subject.

2.1.1 Governing Equations in Linear Elasticity

To derive the one-dimensional scalar wave equation, consider the one-dimensional wave propagation in an elastic rod with Young's modulus E, density ρ, cross-section

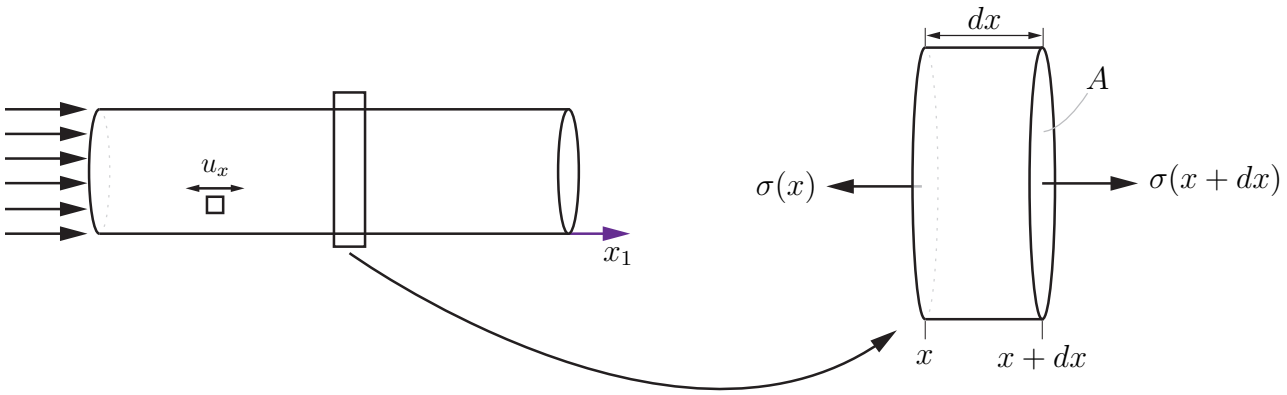


Figure 2.1: One-dimensional propagation in an elastic rod

A along the rod and the two points x and $x + dx$ along the x_1-axis as shown in Figure 2.1. After inserting the relation for stress $\sigma = \frac{F}{A}$, Newton's second law provides

$$\sum F_{ext} = m\,a \iff -\sigma(x,t)A + \sigma(x+dx,t)A = \rho dx A \,\frac{\partial^2 u_x}{\partial t^2}(x,t). \tag{2.1}$$

Expanding Equation 2.1 with the Taylor expansion and reshaping the equation leads to

$$\frac{\partial \sigma}{\partial x}(x,t) = \rho \,\frac{\partial^2 u_x}{\partial t^2}(x,t). \tag{2.2}$$

The media utilized in this work are considered to be continuous, homogeneous, isotropic, and linear elastic. For an elastic, linear material and small displacements with the strain ε it holds that

$$\sigma(x,t) = E\varepsilon(x,t) \qquad \text{and} \qquad \varepsilon(x,t) = \frac{\partial u_x}{\partial x}(x,t). \tag{2.3}$$

From Equation 2.2 and Equation 2.3, the one-dimensional scalar wave equation can be obtained to

$$\frac{\partial^2 u_x}{\partial x^2}(x,t) = \frac{\rho}{E}\frac{\partial^2 u_x}{\partial t^2}(x,t). \tag{2.4}$$

with a solution given in exponential form

$$u_x(x,t) = \left(Ae^{(ikx)} + Be^{(-ikx)}\right) e^{(i\omega t)} \tag{2.5}$$

using the wavelength λ, the wave frequency f, the phase velocity c_p, the angular velocity ω, and the angular wavenumber k. The relationships between those entities are shown below in Equation 2.6.

$$\begin{aligned} \lambda = c_p T \quad c_p = f\lambda \qquad\quad \omega = 2\pi f \\ T = \frac{1}{f} \qquad k = \frac{2\pi}{\lambda} = \frac{\omega}{c_p} \quad \omega = kc_p \end{aligned} \tag{2.6}$$

The one-dimensional wave equation from Equation 2.4 can be generalized to higher dimensions. Using the Lamé parameters λ and μ that connect the Young's modulus E and the Poisson's ratio ν according to

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \tag{2.7}$$

$$\mu = \frac{E}{2(1+\nu)}, \tag{2.8}$$

Hook's law (constitutive stress-strain relationship) for the stress tensor $\tau_{i,j}$ is

$$\sigma_{ij} = \lambda\varepsilon_{kk}\delta_{ij} + 2\mu\epsilon_{ij} \tag{2.9}$$

with the Kronecker delta $\delta_{i,j}$ and the strain tensor $\varepsilon_{i,j}$ relating the displacement to the strain according to

$$\varepsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}). \tag{2.10}$$

Neglecting the body forces, the balance of momentum leads to

$$\sigma_{ij,j} = \rho \ddot{u}_i. \tag{2.11}$$

Substituting sequentially the strain relation from Equation 2.10 into the constitutive stress-strain relationship from Equation 2.9 and then into the balance of momentum from Equation 2.11, the three-dimensional displacement equation of motion for isotropic materials can be obtained to

$$(\lambda + \mu)u_{j,ij} + \mu u_{i,jj} = \rho \ddot{u}_i. \tag{2.12}$$

This equation can be rewritten into its vector representation

$$(\lambda + \mu)\nabla\nabla \cdot \mathbf{u} + mu\nabla^2\mathbf{u} = \rho\ddot{\mathbf{u}}. \tag{2.13}$$

The highly complex nature of the displacement equation of motion in Equation 2.12 or Equation 2.13, respectively, should be noted. Since it is comparably hard to solve for the displacement $\mathbf{u}$ directly, a simpler set of equations can be obtained by the introduction of the scalar potential φ and the vector potential $\mathbf{\Psi}$. According to a theorem given by Helmholtz and [12], a vector field can be decomposed into the gradient of a scalar and the curl of an irrotational vector, i.e.

$$\mathbf{u} = \nabla\varphi + \nabla \times \mathbf{\Psi} \qquad \text{with} \qquad \nabla \cdot \mathbf{\Psi} = 0. \tag{2.14}$$

The condition $\nabla \cdot \mathbf{\Psi} = 0$ guarantees the uniqueness of the solution for the three components of $\mathbf{u}$ from ϕ, ψ_1, ψ_2 and ψ_3.

Substituting Equation 2.14 into Equation 2.13 leads to

$$\nabla^2 \varphi = \frac{1}{c_L^2} \frac{\partial^2 \varphi}{\partial t^2} \qquad \text{and} \qquad \nabla^2 \mathbf{\Psi} = \frac{1}{c_S^2} \frac{\partial^2 \mathbf{\Psi}}{\partial t^2} \tag{2.15}$$

with the wave velocities for the longitudinal or P-wave and shear or S-wave c_L and c_S, respectively. They are given by

$$c_L = \sqrt{\frac{\lambda + 2\mu}{\rho}} \qquad \text{and} \qquad c_S = \sqrt{\frac{\mu}{\rho}}. \tag{2.16}$$

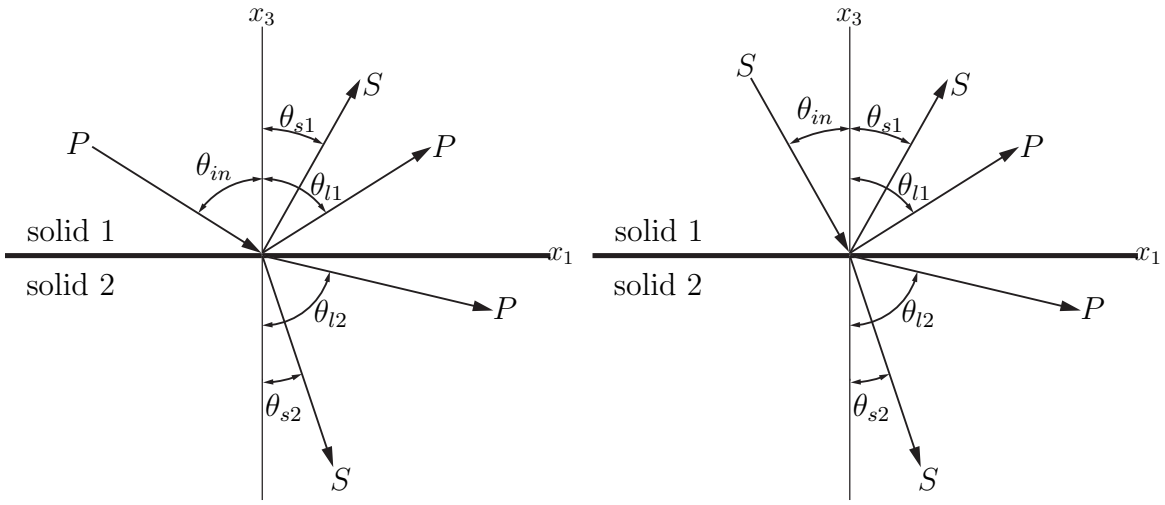
Each of the two basic types of waves is associated with one of each potential from Equation 2.14. P- and S-waves s are furthermore referred to as body waves.

2.1.2 Reflection and Transmission of Body Waves

Reflection and transmission are phenomena that occur when waves interact with boundaries and arise in nearly all applications of ultrasonics. Besides the fundamental physics around wave reflection and transmission which are described in this section, these phenomena lead to so-called *guided waves* which are discussed in the following section.[13]

For this subsection, consider two half-spaces with different, solid material and assume the most general case in which either a longitudinal or a shear incident wave with oblique incidence interacts with the solid-solid interface as shown in Figure 2.2. Regardless of the type of the wave, the incident wave will be reflected and refracted into both P- and S-waves[1]. The effect of a single incidence wave-type producing both P- and S- waves after reflection and refraction is called *mode conversion*.

[1]Exceptions occur for the case of total reflection which is discussed later on.



(a) P-wave incidence. **(b)** P-wave incidence.

Figure 2.2: General acoustic bulk wave interaction (incidence, reflection & refraction) at the interface between two solid half-spaces.

With this setup, all field quantities will depend only on x_1 and x_3. The boundary conditions at the solid interfaces require continuity at $x_3 = 0$. This continuity means that normal and shear stresses, as well as normal and horizontal displacements, need to be equal for both solids at the interface. The boundary conditions can be written as

$$\begin{matrix} \text{vertical displacement:} \\ \text{horizontal displacement:} \\ \text{normal stress:} \\ \text{shear stress:} \end{matrix} \begin{bmatrix} u_1^{(2)} - u_1^{(1)} \\ u_3^{(2)} - u_3^{(1)} \\ \sigma_{33}^{(2)} - \sigma_{33}^{(1)} \\ \sigma_{13}^{(2)} - \sigma_{13}^{(1)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.17}$$

The angles in Figure 2.2 of the incident, reflected, and refracted waves must satisfy Snell's law

$$\frac{\sin\theta_j}{c_j} = const \qquad \text{for} \qquad j = in, s1, s2, l1, l2. \tag{2.18}$$

For an incident angle above a certain value θ_{crit} which is called *critical angle* and which is given by

$$\theta_{crit} = \arcsin \frac{c_S}{c_L} \tag{2.19}$$

the reflected P-wave vanishes and an evanescent surface wave is established. It travels along the x_1 axis and decays in x_3 direction. Further information about wave transmission and reflection can be found in [14].

2.1.3 Rayleigh Waves

In elastic media, two and only two types of waves, i.e. P- and S-waves, can be propagated. In the preceding subsection, the interaction of these waves with a boundary was investigated, and mode conversion was described. Nevertheless, a third type of wave may exist whose effects are confined close to the surface when there is a boundary. According to [10], this type of wave was demonstrated theoretically by the English scientist Lord Rayleigh in 1885, who showed that the amplitude of these surface waves decays rapidly with depth, i.e. the vibration is limited to a shallow layer of approx. one wavelength below the surface, and that their velocity of propagation is smaller than that of body waves. This type of wave is called *Rayleigh wave*. A schematic sketch of a Rayleigh wave can be found in Figure 2.3 where the gray ellipses describe the trajectory of the particles in the surface layer.

Recap that the Helmholtz-decomposition and wave equations from Equation 2.14 and Equation 2.15, respectively, hold for a Rayleigh wave as well

$$\mathbf{u} = \nabla\phi + \nabla \times \mathbf{\Psi} \qquad \text{with} \qquad \nabla \cdot \mathbf{\Psi} = 0. \tag{1.14}$$

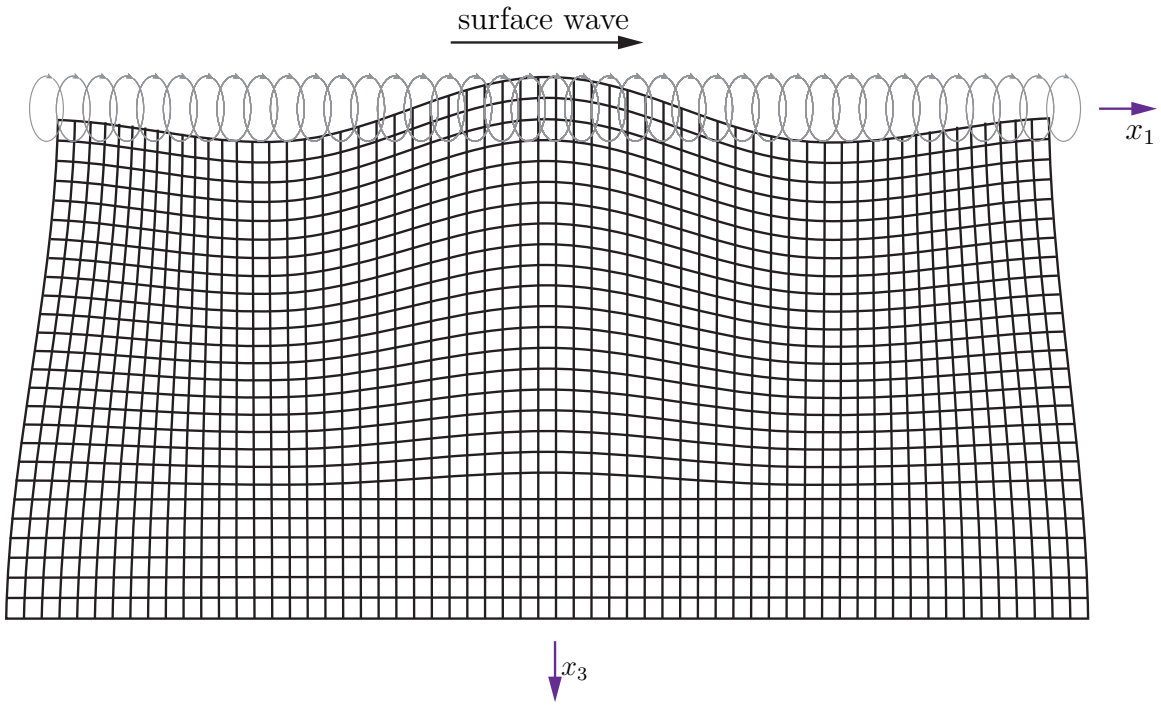


Figure 2.3: Rayleigh surface wave in elastic half-space.

Suppose that the Rayleigh wave propagates in x_1 direction, such that $u_2 = 0$, $\psi_1 = \psi_3 = 0$ and $\psi_2 = \psi$, then the simplified wave equation transfers to

$$\frac{\partial^2 \varphi}{\partial x_1^2} + \frac{\partial^2 \varphi}{\partial x_3^2} + k_L^2 \varphi = 0 \qquad \text{with} \qquad k_L = \frac{\omega}{c_L} \tag{2.20}$$

$$\frac{\partial^2 \psi}{\partial x_1^2} + \frac{\partial^2 \psi}{\partial x_3^2} + k_S^2 \psi = 0 \qquad \text{with} \qquad k_S = \frac{\omega}{c_S}. \tag{2.21}$$

The solutions to the second-order Equation 2.20 and Equation 2.21 allow the potential functions to be rewritten as

$$\varphi = A e^{-\kappa_L x_3} e^{i(k_R x_1 - \omega t)} \qquad \text{with} \qquad \kappa_L = \sqrt{k_R^2 - k_L^2}, \tag{2.22}$$

$$\psi = B e^{-\kappa_S x_3} e^{i(k_R x_1 - \omega t)} \qquad \text{with} \qquad \kappa_S = \sqrt{k_R^2 - k_S^2}, \tag{2.23}$$

with the Rayleigh wavenumber $k_R = \frac{\omega}{c_R}$. It can be seen that the surface wave solution is a combination of coupled longitudinal and shear waves and that both partial waves are evanescent ($k_R > k_S > k_L$).

The normal stress parallel to the surface and the shear stress is

$$\sigma_{33} = \lambda \left(\frac{\partial^2 \varphi}{\partial x_1^2} + \frac{\partial^2 \varphi}{\partial x_3^2} \right) + 2\mu \left(\frac{\partial^2 \varphi}{\partial x_1^2} + \frac{\partial^2 \psi}{\partial x_1 \partial x_3} \right), \tag{2.24}$$

$$\sigma_{31} = \mu \left(2\frac{\partial^2 \varphi}{\partial x_1 \partial x_3} + \frac{\partial^2 \psi}{\partial x_1^2} - \frac{\partial^2 \psi}{\partial x_3^2} \right). \tag{2.25}$$

which are required to be zero at the surface at $x_3 = 0$. Writing this in matrix form and substituting φ and ψ leads to the resulting characteristic equation

$$\left(\kappa_S^2 + k_R^2\right)^2 - 4\kappa_S \kappa_L k_R^2 = 0. \tag{2.26}$$

Resubstituting the expressions from above into Equation 2.26 gives the exact Rayleigh equation

$$\eta^6 - 8\eta^4 + 8(3 - 2\xi^2)\eta^2 - 16(1 - \xi^2) = 0 \quad \text{with} \quad \xi = \frac{c_S}{c_L}, \qquad \eta = \frac{c_R}{c_S} \tag{2.27}$$

which can be approximately solved to

$$\eta \approx \frac{0.87 + 1.12\nu}{1 + \nu}. \tag{2.28}$$

For most cases, it can be assumed that $\eta \approx 0.9$ holds.

Using these findings allows substituting the Rayleigh wavenumber $k_R = \frac{\omega}{c_R}$ into Equation 2.22 and Equation 2.23 which can be substituted into Equation 2.14 then. This is omitted in this work. Doing this shows that there is a 90° phase difference between the normal and tangential displacements which indicate that the particle motion is elliptical in nature and retrogrades with respect to the direction of propa-

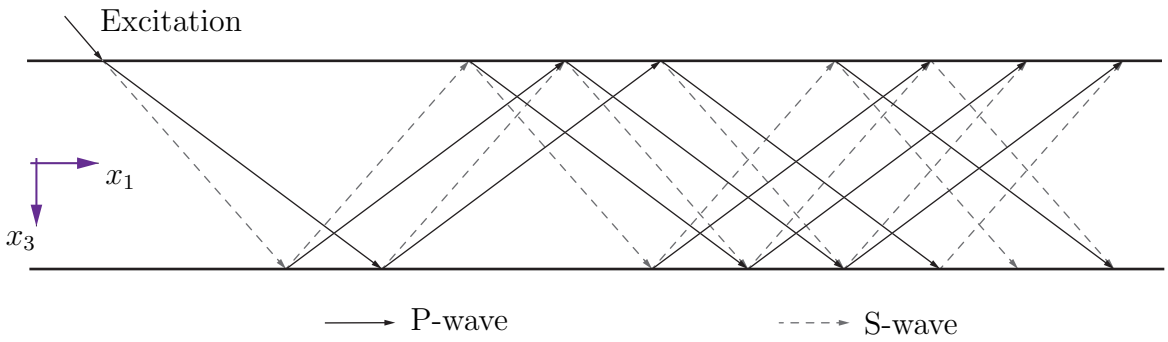


Figure 2.4: Reflection of partial waves in a wave guide.

gation (this means the motion is counter-clockwise for a wave travelling to the right). At the surface, the vertical component of displacement is larger than the horizontal component. The particle motion trajectories are shown as grey ellipses in Figure 2.3.

2.1.4 Wave Guides

In section subsection 2.1.2, reflection phenomena in half-spaces were discussed, where an interface or bounding between two different solids introduced a coupling between P-waves and S-waves, which both can be generated from the other wave type after reflection and refraction. When there is an infinite body with finite dimensions in one direction, like an infinitely long plate, P- and S-waves will be reflected back and forth between the two boundaries. This is shown in Figure 2.4 and causes a pattern of constructive and destructive interference within the body. The interference of the partial waves generate a new kind of wave propagating in the direction of the boundaries. This pattern can be interpreted as a standing wave in x_3 direction while the wave is traveling in x_1 direction. Since these waves are guided by the boundaries, they are called *guided waves* and the body in which this phenomenon happens is called *waveguide*.[15]

To analyze the motion in the elastic layer, consider a solution for the wave equations Equation 2.14 which are two-dimensional for plain strain. For investigation of the wave motion in the elastic layer, consider a solution of the form

$$\varphi = \Phi(x_3)\exp[i(kx_1 - \omega t)], \tag{2.29}$$

$$\psi = \Psi(x_3)\exp[i(kx_1 - \omega t)]. \tag{2.30}$$

Substitution of Equation 2.29 and Equation 2.30 into Equation 2.14 leads to a solution of the resulting equation as

$$\Phi(x_3) = A_1\sin(px_3) + A_2\cos(px_3) \tag{2.31}$$

$$\Psi(x_3) = B_1\sin(px_3) + B_2\cos(px_3) \tag{2.32}$$

with

$$p^2 = \frac{\omega^2}{c_L^2} - k^2, \qquad q^2 = \frac{\omega^2}{c_S^2} - k^2. \tag{2.33}$$

Substituting these solutions into the displacement equations shows that the modes of a wave propagating in the elastic layer can be split up into two systems of symmetric and anti-symmetric modes. Symmetric modes contain mostly longitudinal motion, while anti-symmetric modes contain mostly vertical motion. The expression relating ω tp the wavenumber k can now be obtained from the boundary conditions. Assuming a plate with stress-free boundaries at $x_3 = \pm h$ with plain strain, the Rayleigh-Lamb frequency equations are given by [11] as

$$\text{symmetric modes:} \qquad \frac{\tan(qh)}{\tan(ph)} = -\frac{4k^2pq}{(q^2-k^2)^2}, \tag{2.34}$$

$$\text{asymmetric modes:} \qquad \frac{\tan(qh)}{\tan(ph)} = -\frac{(q^2-k^2)^2}{4k^2pq}. \tag{2.35}$$

Likewise, the guided waves are referred to as *Lamb waves* and are determined by their wavenumber k and their frequency f. At a given frequency, more than one wave with the corresponding wavenumber can exist in the waveguide. The possible combinations of the frequency with their corresponding wavenumber form modes, which differ for different geometries and materials. These branches can be visualized in a frequency-wavenumber diagram like Figure 2.5 for a 1mm thick plate of Zirconium-4, where each of the branches describes one propagating mode. It should be noted that the solution of all modes should be continuous for all wavenumbers starting at zero. Even though the equations look rather simple, solving them is comparably hard. The modes should be continuous and start at $k = 0\frac{1}{\mathrm{m}}$. The reason why this is not the case in Figure 2.5 is purely computationally induced. Solving for the modes was first done by [16], and is additionally described by [17]. Today, both free and commercial software for finding the roots of Equation 2.34 and Equation 2.35 is available. For this work, [18] was used. Note that Lamb modes are dispersive, hence the propagation velocity depends on its frequency. A different visualization of this can be found in Figure 2.6 which contains the same underlying data from a 1mm thick Zirconium-4 plate as in Figure 2.5 and only uses a different representation to demonstrate the different mode speeds at different frequencies.

Symmetric and asymmetric modes are not only appearing in single material systems, but exist in layered media too, even though there they cannot be clearly distinguished as symmetric and asymmetric anymore. [19] gives an introduction to Lamb wave propagation in layered media.

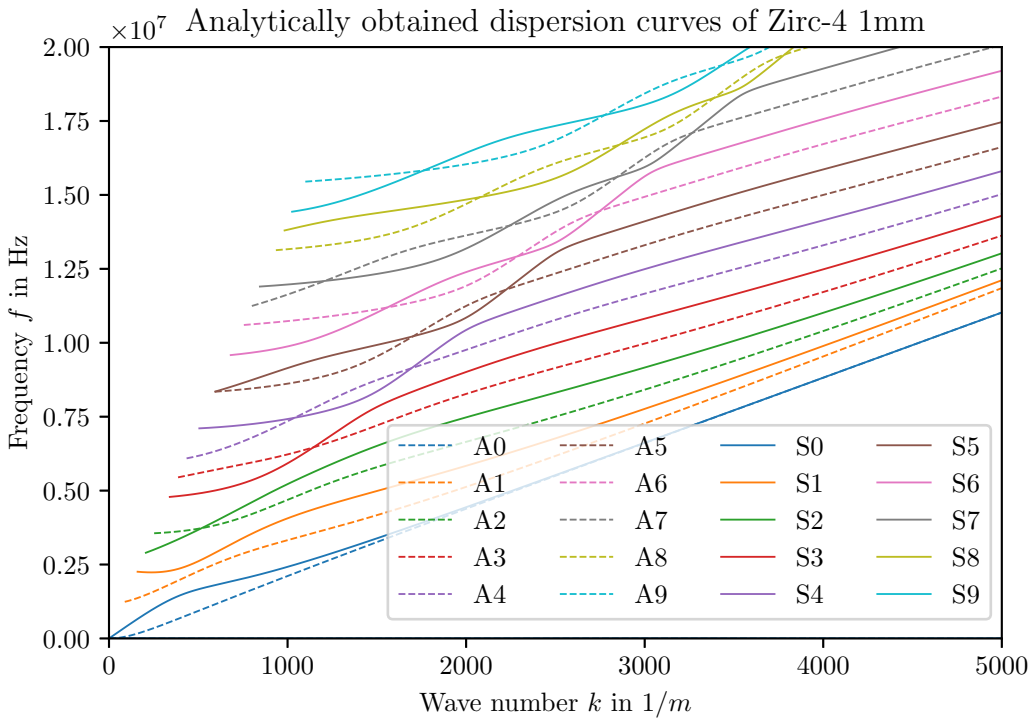


Figure 2.5: Analytically obtained dispersion relation for a 1mm thick Zirconium plate in the frequency-wavenumber representation.

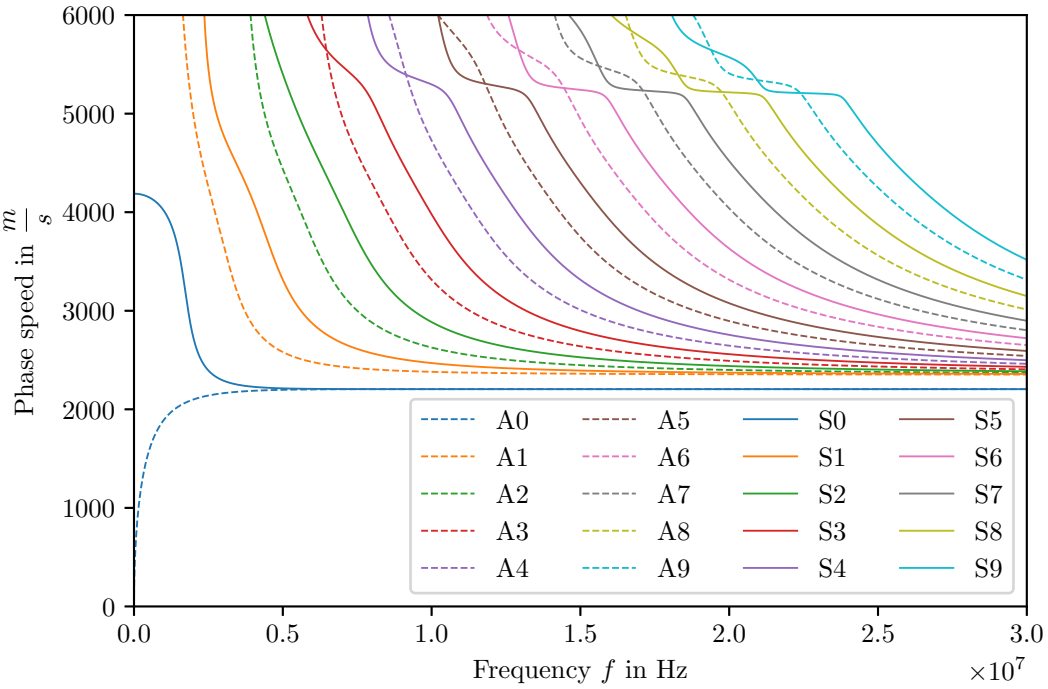


Figure 2.6: Dispersion relation for a 1mm thick Zirconium plate in phase speed-frequency representation.

2.2 Finite Element Analysis Foundations

In finite element analysis (FEA), a body is approximated with discrete finite elements which are connected at N finite element nodal points on the element boundaries. Therefore, the displacement at element m can be written as

$$\mathbf{u}^{(m)}(x_1, x_2, x_3) = \mathbf{H}^{(m)}(x_1, x_2, x_3)\hat{\mathbf{U}} \tag{2.36}$$

with the displacement interpolation matrix $\mathbf{H}^{(m)}$ and $\hat{\mathbf{U}}$ as a vector of the three global displacement components X_1, X_2, X_3 at all nodal point, such that

$$\hat{\mathbf{U}} = [X_{1,1}, X_{2,1}, X_{3,1}, \ ..., \ X_{N,3}, X_{N,3}, X_{N,3}] \tag{2.37}$$

The choice of $\mathbf{H}^{(m)}$ is one of the basic steps of a finite element solution.

The corresponding strain elements are given by

$$\boldsymbol{\varepsilon}^{(m)}(x_1, x_2, x_3) = \mathbf{B}^{(m)}(x_1, x_2, x_3)\hat{\mathbf{U}} \tag{2.38}$$

with the strain-displacement matrix $\mathbf{B}$ whose rows are obtained by combining and appropriately differentiating rows of $\mathbf{H}^{(m)}$.

The stresses are related to the strains and initial stresses σ^i according to

$$\sigma^m = \mathbf{C}^m\boldsymbol{\varepsilon}^{(m)} + \sigma^{i(m)} \tag{2.39}$$

with the elasticity matrix $\mathbf{C}^{(m)}$ of element m. $\mathbf{C}^{(m)}$ is specified by the material law and can differ from element to element.

With this, the general equation of motion can be derived. This research assumes linear elasticity. The general equation of motion and can be stated in matrix form in

terms of the nodal displacements **u** and their derivative for the equilibrium are given as

$$\mathbf{M\ddot{u}} + \mathbf{C\dot{u}} + \mathbf{Ku} = \mathbf{R} \tag{2.40}$$

with the load vector **R**, and where **M** is the mass matrix given by

$$M = \sum_m \int_{V^m} \rho^{(m)} \mathbf{H}^{(m)T} \mathbf{H}^{(m)} dV^{(m)},, \tag{2.41}$$

with the mass density of element m which is $\rho^{(m)}$. **C** is the damping matrix of the formally structure

$$C = \sum_m \int_{V^m} \kappa^{(m)} \mathbf{H}^{(m)T} \mathbf{H}^{(m)} dV^{(m)}, \tag{2.42}$$

with $\kappa^{(m)}$ as the damping property parameter of element m, and **K** is the stiffness matrix described by

$$K = \sum_m \int_{V^m} \mathbf{B}^{(m)T} \mathbf{C}^{(m)} \mathbf{B}^{(m)} dV^{(m)}. \tag{2.43}$$

For further reading the author recommends [20], [21], and [22].

2.3 Signal Processing and the Fourier Transform

This section presents the fundamental signal processing methods used in this work. As described earlier, a wave propagating in solids usually consists of several frequencies. Hence, a physical process like wave propagation can be described in the time domain, where some physical quantity depends on the time, e.g. $s(t)$, or in the frequency domain where the process is defined by its amplitude S and as a function of frequency, i.e. $S(f)$. It is useful to think of both $s(t)$ and $S(f)$ as different representations of

the same function. Since an analysis in the time domain makes solving the problems in wave propagation harder, the analysis in this work takes place primarily in the frequency domain. The transformation from the time to the frequency domain is conducted by the Fourier transformation.

In the following, first the Fourier transformation for continuous signals and secondly for discretely sampled data is described. In the last step the Fourier transformation for real data in two dimensions is discussed as this work heavily relies on this method. Other methods like the short-time Fourier transform or the Wavelet transform, as described in [23], are not further investigated. For a deeper understanding of the Fourier transform, the author recommends [24], [25], and [26].

2.3.1 Fourier Transform for Continuous Signals

Classically, the Fourier transformation was developed for continuous functions. For a function $s(t)$ which is absolutely integrable, i.e. $||s(t)||_1 = \int_{-\infty}^{\infty} |s(t)|dx < \infty$, the continuous Fourier transformation $\mathcal{F}$ and its inverse $\mathcal{F}^{-1}$ are defined as

$$\begin{aligned} \mathcal{F}\{s(t)\} = S(f) &= \int_{-\infty}^{\infty} s(t)e^{i2\pi ft}dt, \\ \mathcal{F}^{-1}\{S(f)\} = s(t) &= \int_{-\infty}^{\infty} S(f)e^{-i2\pi ft}df. \end{aligned} \tag{2.44}$$

If $s(t)$ is measured in seconds, then $S(f)$ from Equation 2.44 has the unit cycles per second or Hertz $\left(\frac{1}{s}\right)$. If $s(t)$ is measured in meters, then $S(f)$ is a function of the inverse wavelength $\left(\frac{1}{m}\right)$.

The same Fourier transformation can be written with the use of the angular frequency $\omega = 2\pi f$ as

$$\begin{aligned} \mathcal{F}\{s(t)\} = S(\omega) &= \int_{-\infty}^{\infty} s(t)e^{i\omega t}dt, \\ \mathcal{F}^{-1}\{S(\omega)\} = s(t) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} S(\omega)e^{-i\omega t}d\omega. \end{aligned} \tag{2.45}$$

For further information consulting signal processing literature like [25] and [27] is recommended.

2.3.2 Fourier Transform of Discretely Sampled Data

Practically, most signals are captured digitally in a discrete way. For the discrete Fourier transformation a function is estimated from a finite number of sampled points. Assume that there are N successive sampled points with the sampling interval Δ

$$s_k \equiv s(t_k), \qquad t_k \equiv k\Delta, \qquad k = 0, 1, 2, ..., N-1 \tag{2.46}$$

and that the sampled points are at least *typical* of how $s(t)$ looks like if the analyzed time interval does not lay completely within the sampled points. Since not more than N independent numbers of output can be produced from the N sample points, the Fourier transform $S(f)$ cannot be estimated at all values of $f \in [-f_c, f_c]$, where f_c is the Nyquist critical frequency given by

$$f_c \equiv \frac{1}{2\Delta}. \tag{2.47}$$

Hence, instead of computing the values for all frequencies f, only the values at discrete frequencies are estimated at

$$f_n \equiv \frac{n}{N\Delta} \qquad \text{for} \qquad n = -\frac{N}{2}, ..., \frac{N}{2}. \tag{2.48}$$

Note that the extreme values for f_n correspond to the lower and upper limits of the Nyquist critical frequency range.

To calculate the amplitude values at the above mentioned discrete frequencies, the final step is to approximate the integral from Equation 2.44 by a discrete sum to

$$S(f_n) = \int_{-\infty}^{\infty} s(t)e^{-i2\pi ft} \approx \sum_{k=0}^{N-1} se^{2\pi i f_n t_k}\Delta = \Delta \sum_{k=0}^{N-1} s_k e^{2\pi ikn/N} \tag{2.49}$$

where Equation 2.46 and Equation 2.48 have been used for the final transformation. The discrete Fourier transform of N points is therefore defined by

$$S_N \equiv \sum_{k=0}^{N-1} s_k e^{2\pi ikn/N}. \tag{2.50}$$

2.3.3 Fourier Transforms of Real Data in Two Dimensions

The two-dimensional Fourier transform was originally developed for analyzing experimental results in the form of multimode time signals as described by [28]. Since it allows obtaining a frequency-wavenumber representation from discrete measurements from various sources, the 2D-Fourier transformation is used heavily in this research.

Assume a complex function $s(k_1, k_2)$ is given over the two-dimensional grid $0 \leq k_1 \leq N_1 - 1$, $0 \leq k_2 \leq N_2 - 1$. Then, its two-dimensional discrete Fourier transform $S(n_1, n_2)$ can be defined by

$$S(n_1, n_2) = \sum_{k_2=0}^{N_2-1} \sum_{k_1=0}^{N_1-1} s(k_1, k_2) e^{2\pi ik_2 n_2/N_2} e^{2\pi ik_1 n_1/N_1}. \tag{2.51}$$

Usually, n_1 and n_2 will correspond to one spatial direction and time or to two spatial directions. In this work, the data is sampled spatially and temporally. The advantage of this method is that different modes can be easily distinguished because of each mode's distinct frequency-wavenumber relation as described in subsection 2.1.4. In the following, the two-dimensional Fast-Fourier transform (2D-FFT) will be used

instead of the standard 2D-Fourier transform since it is computationally optimized but offers the same results.

2.4 Deep Learning

In the inversion part of this work, a deep convolutional neural network (CNN) is used. Hence, this section discusses the theoretical basics of deep neural networks. Even though deep learning is traditionally a part of machine learning, its complexity goes way beyond simple supervised machine learning classifiers. Because of this, the entire following section is devoted to deep learning.

First, an introduction into the simplest network structure, so-called feedforward networks or multilayer perceptrons is given. It is discussed how data is passed through a network and how deep networks learn. Lastly, convolutional neural networks as a special kind of feedforward networks are described.

2.4.1 Deep Feedforward Networks

The fundamental idea of a deep feedforward network is to approximate a function f^*. In the given case of classification, this can be $y = f^*(\mathbf{x})$ mapping an input $\mathbf{x}$ to a category y. To be precise, the network defines the mapping $y = f(\mathbf{x}, \mathbf{W})$ and learns the parameters $\mathbf{W}$ that result in the best function approximation. Information only flows from the input $\mathbf{x}$ through the intermediate computations used to define f to the output with no feedback connections. Feedforward networks typically compose together many different functions. Assume there are the three functions $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$ connected in chain, i.e. $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$. This structure is used often in neural networks, where $f^{(1)}$ is called the first layer, $f^{(2)}$ is called the second layer, and $f^{(3)}$ is called the third layer. The first layer of a network is called input layer and the final layer is called the output layer, while the layers in between are referred to as hidden layers. The number of layers defines the depth of the network. During

training, the network tries to match the learned function $f(\mathbf{x})$ as close as possible to the ground truth function $f^*(\mathbf{x})$. The training data consists of noisy samples from $f^*(\mathbf{x})$ and is accompanied by a label $y \approx f^*(\mathbf{x})$. Since the training data is labeled, each training data sample specifies exactly what the desired output if the output layer should be, i.e. a value that is sufficiently close to the ground truth label y. All other layers are not specified directly by the training samples and the learning algorithm must decide by itself how to use the input and hidden layers to obtain the best approximation of f^*.

In general, each layer in a linear feedforward neural network represents a simple matrix operation in layer i as

$$\mathbf{h}^{(j)} = \mathbf{W}^{(i)} h^{(i)} + b^{(i)} \qquad \text{with} \qquad j = i + 1, \tag{2.52}$$

with the layer output $h^{(j)} \in \mathbb{R}^{N_j}$ ($\triangleq$ input layer of the next jth layer), the weights or parameters $\mathbf{W}^{(i)} \in \mathbb{R}^{N_j \times N_i}$, layer input $h^{(i)} \in \mathbb{R}^{N_i}$ and bias vector $b^{(i) \in \mathbb{R}^{N_j}}$.

Since a linear mapping as in Equation 2.52 does not allow to learn nonlinear relationships, so-called *activation functions* or *non-linearities* are used. A non-linearity is a nonlinear function applied to the output of the linear matrix operation, such that in practice usually

$$\mathbf{h}^{(j)} = g^{(i)} \left(\mathbf{W}^{(i)} h^{(i)} + b^{(i)} \right) \qquad \text{with} \qquad j = i + 1, \tag{2.53}$$

holds with the activation function $g^{(i)}$. In machine learning practice, most networks use the Rectified Linear Unit (ReLU).

Forward Pass

In Figure 2.7 a schematic visualization of a neural network with three layers during the forward pass is shown. The input to the network is $\mathbf{x} \in \mathbb{R}^D$ and the output is

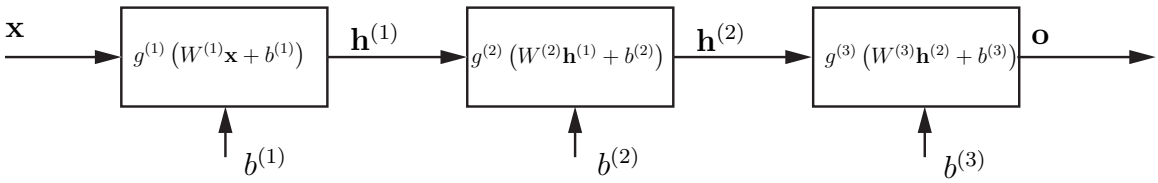


Figure 2.7: Schematic forward propagation through three layered feedforward neural network.

$\mathbf{o} \in \mathbb{R}^{N_3}$. The input of the network is fed into the first layer where it is multiplied with the weights of the neurons of the first layer $W^{(1)} \in \mathbb{R}^{N_1 \times D}$ and added to the biases of the first layer $b^{(1)} \in \mathbb{R}^{N_1}$. To make the network suitable to learn non-linear functions, this sum is then fed into a non-linear activation function of layer one, i.e. $g^{(1)}$. In this example, each unit takes as input all the units from the previous layer which is called *fully connected*. The output of layer one is analogously fed into layer two, such that $h^{(1)} \in \mathbb{R}^{N_1}$, $W^{(2)} \in \mathbb{R}^{N_2 \times N_1}$, $b^{(2)} \in \mathbb{R}^{N_2}$, and $h^{(2)} \in \mathbb{R}^{N_2}$.

Optimization for Deep Models

To optimize a network, a measure that quantifies how *good* the network is, is vital. In most machine learning scenarios, P is chosen to be the real performance measure. P is then d defined with respect to the test set but may be indissoluble in most cases. Since P cannot be optimized directly generally, a different loss or cost function $J(\mathbf{W})$ which can be optimized, is optimized with the hope that optimizing $J(\mathbf{W})$ also improves P. Therefore, learning in a neural network consists of minimizing a cost function $J(\mathbf{W})$ with respect to the parameters $\mathbf{W}$ over the training set. The cost function can be usually written as an average over the training set like

$$J(\mathbf{W}) = \mathbb{E}_{(\mathbf{x},y)\sim\hat{p}_{data}} L\left(f(\mathbf{x},\mathbf{W}),y\right), \tag{2.54}$$

with the loss function L, the predicted output $f(\mathbf{x},\mathbf{W})$, the ground truth target y, and the empirical data distribution $\hat{p}_{data}$. In general, it would be optimal to minimize

the cost function across the unknown data-generating distribution p_{data} rather than just over the known finite training set.

The optimal parameter set $\mathbf{W}^*$ can be obtained by solving the minimization problem

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \sum_{i=1}^{N} L\left(f(\mathbf{x}^{(i)}, \mathbf{W}), y^{(i)}\right). \tag{2.55}$$

with the n-th input $\mathbf{x}^{(n)}$ within N data points.

Stochastic Gradient Descent

After discussing suitable performance measures for a neural network, the question arises on how to minimize a complicated function of parameters. The answer to this is backpropagation, which is technically repeated application of the chain rule [29]. The gradient of the loss is given by

$$\nabla_{\mathbf{W}} L(\mathbf{W}) = \sum_{i=1}^{N} \nabla_{\mathbf{W}} L_i(\mathbf{x}_i, y_i, \mathbf{W}). \tag{2.56}$$

and is pointing in the direction where the loss increases the most. This gradient is then used to update the network parameters according to

$$\mathbf{W} \leftarrow \mathbf{W} - \epsilon \nabla_{\mathbf{W}} L(\mathbf{W}), \tag{2.57}$$

with the learning rate $\epsilon \in [0, 1]$ as a hyperparameter. Since a full summation over the entire dataset in Equation 2.56 is expensive for large N, it is common practice to use a minibatch of examples instead of the whole summation over the entire data set. When the examples are shuffled and randomly selected, the gradient descent algorithm gets stochastic. This gives the algorithm its name: *Stochastic Gradient Descent (SGD).*

In practice, mostly computationally optimized enhancements of SGD like Adam [30] are used.

2.4.2 Convolutional Neural Networks

One major drawback of fully connected networks is that their number of parameters rises quickly fast. A huge number of parameters, in turn, requires more training samples that might not be accessible, as well as more computational resources like storage and memory, are need. Additionally, for a lot of computer vision problems, a spatial correlation is mostly local anyway.

At this point, convolutional neural networks (CNNs) [31, 32] step in. CNNs can be used to process data with a known grid-like topology. They are simply neural networks that use the convolution operator instead of general matrix multiplication in at least one of their layers.[33]

In engineering and science, convolution is usually defined as

$$s(t) = (x * w)(t) = \int x(a)w(t-a)da, \tag{2.58}$$

with the input x, the kernel w, and the output $s(t)$ which is sometimes called feature map.

In machine learning, the data to use is usually in a form of a multidimensional array and the kernel is a multidimensional finite array of parameters adapted by the learning algorithm. This allows to discretizing Equation 2.58. For a two-dimensional image I as an input with the two-dimensional kernel k the convolution operation for the (i, j) element is

$$S(i,j) = (I * k)(i,j) = \sum_m \sum_n I(m,n)k(i-m, j-n). \tag{2.59}$$

At this point, it should be noted that most modern machine learning frameworks do not implement the convolution from Equation 2.59 and instead use a related function called cross-correlation which is given by

$$S(i,j) = (I * k)(i,j) = \sum_m \sum_n I(i+m, j+n)k(m,n), \tag{2.60}$$

but call it convolution still. An example of 2D-convolution can be found in Figure 2.8. It can be seen that the same parameters α, β, γ, and δ are used for all convolution operations. This is called *parameter sharing* and reduces the number of learned parameters at large.

A typical convolutional layer in a CNN includes three steps. In the first step, the convolution is performed. In the second step, the output feature map is run through a non-linearity, e.g. ReLU. In the third and last step, a pooling function is applied to modify the layer further. A pooling function uses the nearby outputs of a certain location and replaces the output with a summary statistic of those. For further readings about convolutional neural networks, [33] and [34] are recommended.

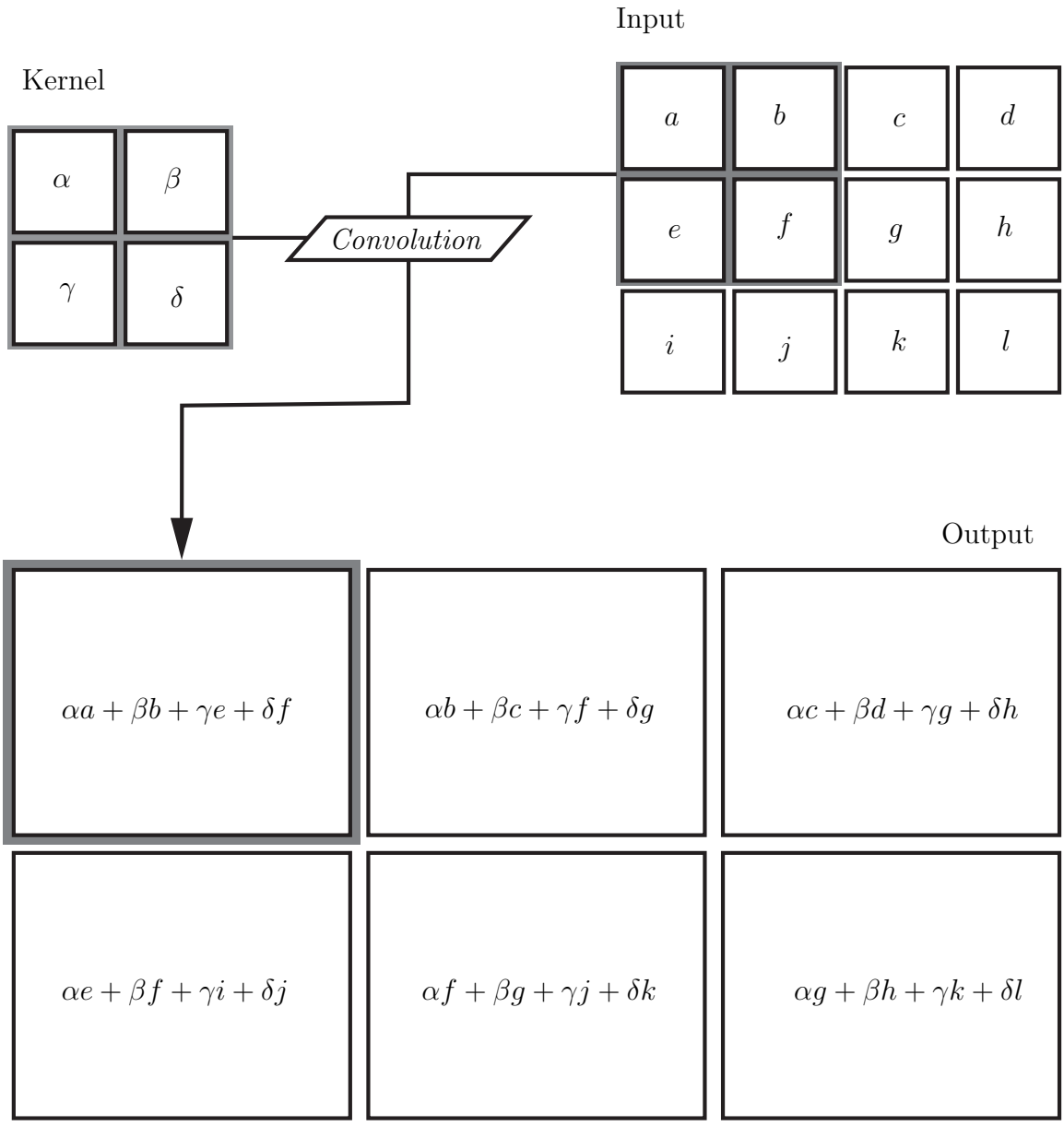


Figure 2.8: Example of a two-dimensional convolution with a 2×2 kernel and restriction to output values where the kernel lays completely in the input tensor (no padding needed in this case).

CHAPTER 3
FORWARD PROBLEM

In this chapter, the forward problem is discussed. During the forward problem, the material properties are known and the state, i.e. displacements, of the analyzed solid are calculated. For the forward problem the finite element analysis software Abaqus CAE is used. All postprocessing is conducted in Python 3. Abaqus itself was automated with Python 2 to upscale and parallelize simulations.

In the following, the used materials are described first. Then, an in-depth explanation into the FEA model is given. Then, the 2D-FFT postprocessing which creates dispersion plots from the simulated data is described. An introduction into max-pooling and function fitting is given which is used to extract features for the standard machine learning inversion part.

3.1 Material Description

To increase the performance of accident tolerant fuels, coated zirconium alloys are a promising solution [35]. Especially, a combination of a zirconium alloy with a chromium coating is auspicious [36, 37]. Therefore, in this research, the two-layer bonded system uses a 1mm thick Zirconium-4 plate with a Chromium coating of varying thickness. The material properties of both materials can be found in Table 3.1.

Material	Young's Modulus	Poisson's Ratio	Density
Zircaloy-4	$E_p = 99.3\text{GPa}$	$\nu_p = 0.37$	$\rho_p = 6.56\frac{\text{g}}{\text{cm}^3}$
Chromium	$E_c = 279\text{GPa}$	$\nu_c = 0.21$	$\rho_c = 7.19\frac{\text{g}}{\text{cm}^3}$

Table 3.1: Material properties for coating and plate

3.2 FEA Basics in Abaqus

Abaqus is a software suite that offers five software products for FEA:

- Abaqus/CAE,
- Abaqus/Standard,
- Abaqus/Explicit,
- Abaqus Multiphysics,
- Add-on Tools.

For this research, primarily Abaqus/Standard and Abaqus/Explicit are of interest, since those two packages can be used to solve dynamic problems like Equation 2.40. For Abaqus/Standard, an unconditionally stable implicit algorithm is used. The implicit algorithm takes more time to compute and needs significantly more computational ressources than Abaqus/Explicit. Abaqus/Explicit uses an explicit solver which makes it computationally efficient for the analysis of large models with relatively short dynamic response times. This makes it optimal for high-frequency simulations. In the following, the fundamental properties for an FEA analysis are described since a smart parameter selection has a huge influence on the computation's cost and accuracy.

3.2.1 Integration Step Time

Abaqus/Explicit integrates the state of the model using many small time increments Δt based on the state of the model and the start of the increment at time t. The central difference operator is only conditionally stable with the stability limit being approximately equal to the time for an elastic wave to cross the smallest element

dimension in the model. Per default, Abaqus/Explicit approximates this stability limit across any of the elements in the mesh via

$$\Delta t \approx \frac{l_{e,min}}{c_L} \tag{3.1}$$

where $l_{e,min}$ is the smallest element length in the mesh and c_L is the wave speed of a longitudinal wave. Since equation Equation 3.1 provides the approximate stability limit, practically, Abaqus/Explicit uses a more conservative and safe estimate for the maximum integration step size which is given by

$$\Delta t \approx \frac{l_{e,min}}{\sqrt{2}c_L}. \tag{3.2}$$

Additionally, Abaqus uses a variable step size to speed up the simulations per default. The automatically calculated step size for Abaqus/Explicit seems to produce good results. However, the software allows the user to manually specify the time step too. When manually choosing a step size, two cases need to be taken into consideration: if the time step is too short, a lot of calculation time and resources are wasted, but if the time step is too long, high-frequency vibrations are not resolved accurately and the solution process might become numerically unstable. A rule of thumb for the integration step time is given by [38] as

$$\Delta t = \frac{1}{20 f_{max}}. \tag{3.3}$$

3.2.2 Meshing

Meshing a model is a fundamental step in FEA. After meshing, a discretized representation of the model is obtained. Typically, the smaller the mesh size, the more accurate is the solution since the mesh elements are better sampled across the physical domains. The trade-off is that with higher accuracy the simulation gets larger,

i.e. more computational resources are needed and the simulation time increases. To estimate the optimal element size l_e, the Rayleigh wave speed needs to be obtained. From the material properties and according to Equation 2.7 and Equation 2.8, the Rayleigh speed for Chrome is given by

$$c_{R,Cr} = \eta_{Cr}\sqrt{\frac{\mu_{Cr}}{\rho_{Cr}}} = 3657.5\frac{\mathrm{m}}{\mathrm{s}}. \tag{3.4}$$

The size of the mesh elements needs to be chosen in a way that the propagating waves are captured accurately. It is recommended to use n_{mesh} elements per wavelength. [28] suggests to choose $n_{\mathrm{mesh}} = 10$, but a smaller number might be feasible to speed up the simulation. The maximal wavenumber k_{max} is given by

$$k_{max} = \frac{1}{\lambda_{min}} \tag{3.5}$$

so the element size can be calculated from

$$l_e = \frac{1}{n_{\mathrm{mesh}} k_{max}} \tag{3.6}$$

$$= \frac{\lambda_{min}}{n_{\mathrm{mesh}} k_{max}} \approx \frac{c_R}{n_{\mathrm{mesh}} f_{max}} \tag{3.7}$$

$$\tag{3.8}$$

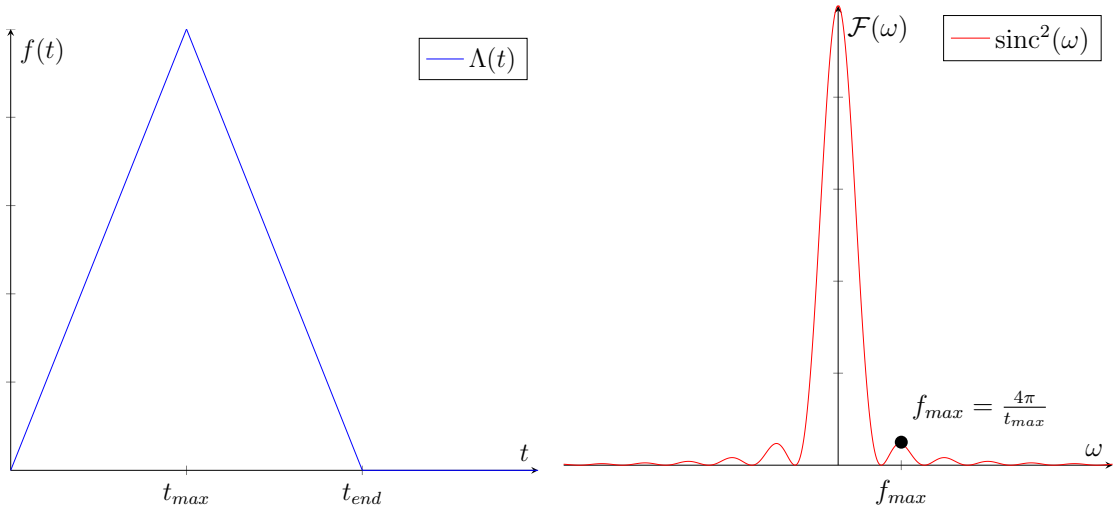
Remark: in this work, the same mesh size is used for both, the coating and the base plate. The reason for this is that a difference in mesh size can cause reflection artifacts at the boundary between the two materials. Since the Rayleigh speed for the coating is higher than the Rayleigh speed of the plate, it is feasible to use the coatings Rayleigh speed to determine the overall mesh size.

The element size is the most critical parameter since it has a huge influence on the computational cost as well as on the accuracy of the overall simulation.

3.2.3 Excitation

Since the objective of this research is to obtain a dispersion frequency-wavenumber representation for multiple frequencies, the excitation needs to be an impulse that contains those frequencies. In this research, a triangular pulse as in Figure 3.1a is used. The Fourier transform of a triangular pulse is the squared Sinus cardinalis (Sinc) function as shown in Figure 3.1b. The point in time where the pulse reaches its maximum is denoted by t_{max}. The maximum frequency obtained from the excitation of the Sinc function will be defined as the second maximum besides 0. This maximum frequency is given by

$$f_{max} = \frac{4\pi}{t_{max}}. \tag{3.9}$$



(a) Triangular pulse function $\Lambda(t)$ in the time domain.

(b) The Fourier transformed triangular pulse is the $\text{sinc}^2(\omega)$ function in the frequency domain.

Figure 3.1: Excitation signal in time and frequency domain.

The excitation takes place at the upper right corner of the coating on the first node at the edge. Using the first node at the edge has the advantage, that reflection effects soon after the excitation are bypassed. A visualization of the excitation is shown in

Figure 3.2. In this work, the triangular function is designed with $t_{max} = 20ns$ which allows excitation of frequencies up to $f_{max} = 628\text{MHz}$ according to Equation 3.9.

3.2.4 Sampling

Sampling in conducted both in the time as well as in the spatial domain. Both domains are subject to the Nyquist criterion. For the time domain, the Nyquist criterion states the upper frequency limit for capturing a continuous signal to avoid aliasing. The continuous signal needs to be sampled at least twice within the period of the highest frequency according to

$$\Delta T_{sampling} = \frac{1}{2f_{max}}. \tag{3.10}$$

In this work, a sampling time $\Delta T_{sampling} = 2 \cdot 10^{-8}\text{s} = 20\text{ns}$ is used to avoid aliasing up to a frequency of $f_{max} = 628\text{MHz}$. The practically usable maximum frequency is way lower though. In Abaqus, the sampling time corresponds to the sampling time for the history output requests.

A similar relation can be established for the maximum sampling distance, i.e. the maximum distance between two sample locations, as

$$\Delta X = \frac{1}{2k_{max}} \tag{3.11}$$

with the maximum wavenumber to be captured $k_{max} = 18,000\frac{1}{\text{m}}$. This gives a maximum sampling distance $\Delta X = 27\mu\text{m}$.

For spatial sampling, all nodes between $d_{start} = 3\text{cm}$ and $d_{end} = 7\text{cm}$ on the upper surface of the coating are used.

3.2.5 Abaqus FEA Model

In this research, two types of models are simulated. First, a coated plate with a uniform coating is simulated (see Figure 3.2), then a coated plate with a non-uniformness in the coating is simulated (see Figure 3.3). Though the basic decisions regarding accuracy and computational cost are the same for both models, constructing the models within Abaqus differs. The simulation duration for both models is $t_{end} = 92\mu$s.

Both FEA models use four-node plane stress elements (CPS4R) with two degrees of freedom at each node (translation in x_1 and x_3 direction). On one side of the plate, the elements are constrained by a boundary condition which fixes their x_1-coordinates. The excitation is conducted via a time-depending displacement boundary condition applied to the top-left node as shown in Figure 3.2 and Figure 3.3. Italic numbers are parameters that are changed between simulations, and only an example value is used in the drawings. The nodes of the coating and plate are merged at the intersection for both models. The uniform model's coating is modeled from one part. The non-uniform model's coating is constructed from three parts: the boundary, the non-uniformness, and the sampling nodes part. All three parts are merged at the respective interfaces before the coating nodes are merged with the plate nodes.

The non-uniformness is always placed between the excitation node and the sampling nodes with a certain distance to both. This way it is assured that the waves generating the displacements at the sampling nodes passed the non-uniformness part before. To avoid effects caused by the near field after the excitation, the distance between the excitation node and the first sampling node is always at least 3cm. Both models are meshed according to Equation 3.8 with an element length of $l_e = 8.8\mu$m.

Simulating one of the models on the Georgia Tech high-performance cluster PACE [39] takes between 15 and 25 hours depending on the thickness and geometry of the coating.

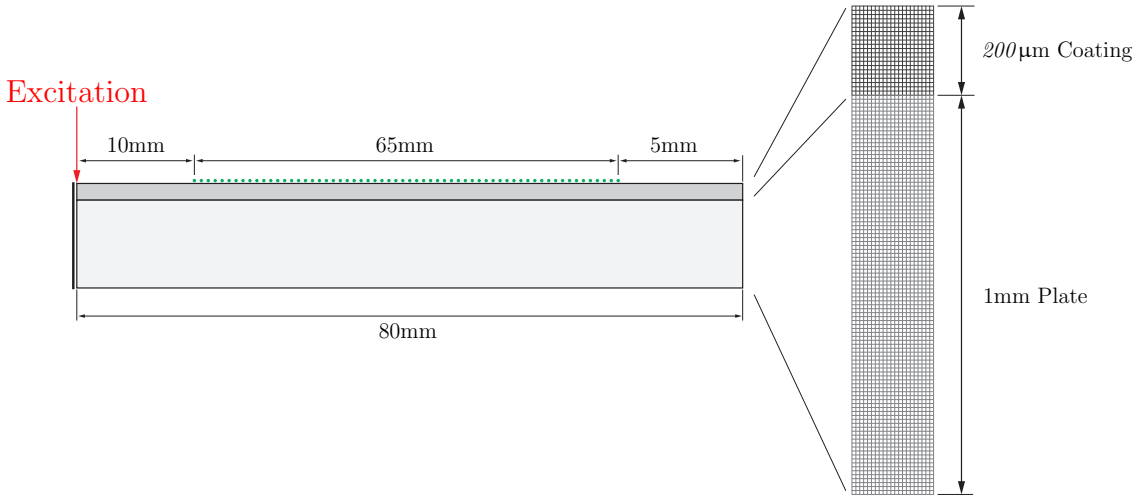


Figure 3.2: Sketch of FEA model with uniform coating (not drawn to scale).

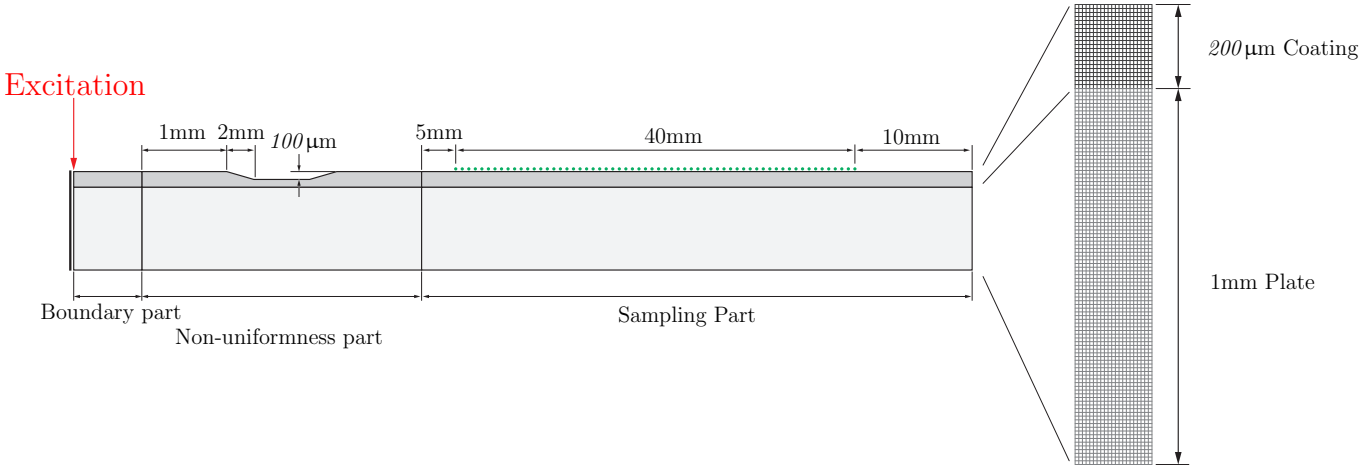


Figure 3.3: Sketch of FEA model with non-uniform coating (not drawn to scale).

3.3 Postprocessing

In the postprocessing step, only the out of plane displacement over time in x_3-direction is used. This information is extracted from the Abaqus simulation output database file and is stored in a two-dimensional matrix. The two dimensional discrete Fourier transform from Equation 2.51 transforms the data in a two-dimensional matrix in the frequency-wavenumber domain for further analysis and processing.

3.4 FEA Outcomes

For this work, various coating thicknesses in the range between $10\mu m$ and $600\mu m$ were simulated in Abaqus/Explicit. In the following analysis, only the uniform case is analyzed. First, the theoretical idea and its complexity behind the analysis is explained. Secondly, simulation results for different thicknesses are described and evaluated.

3.4.1 Concept and Complexity Behind the Analysis

Setting up the FEA model and obtaining usable simulation results turns out to be highly complex for the given material combination. The problem is challenging to analyze mainly for two reasons:

1. The material properties, and especially the stiffnesses, of coating and plate are comparably similar. While a related problem was discussed in [3, 4] with a layered specimen of Aluminum and Tape, Young's moduli differed by a factor of 70. This research analyzes the case when coating's and plate's Young's modulus are more similar and differ only by a factor smaller than 3.

2. The coating thickness is examined in a wide range from 10µm up to 600µm, i.e. down to a hundredth of the plate's thickness, making real physical experimental

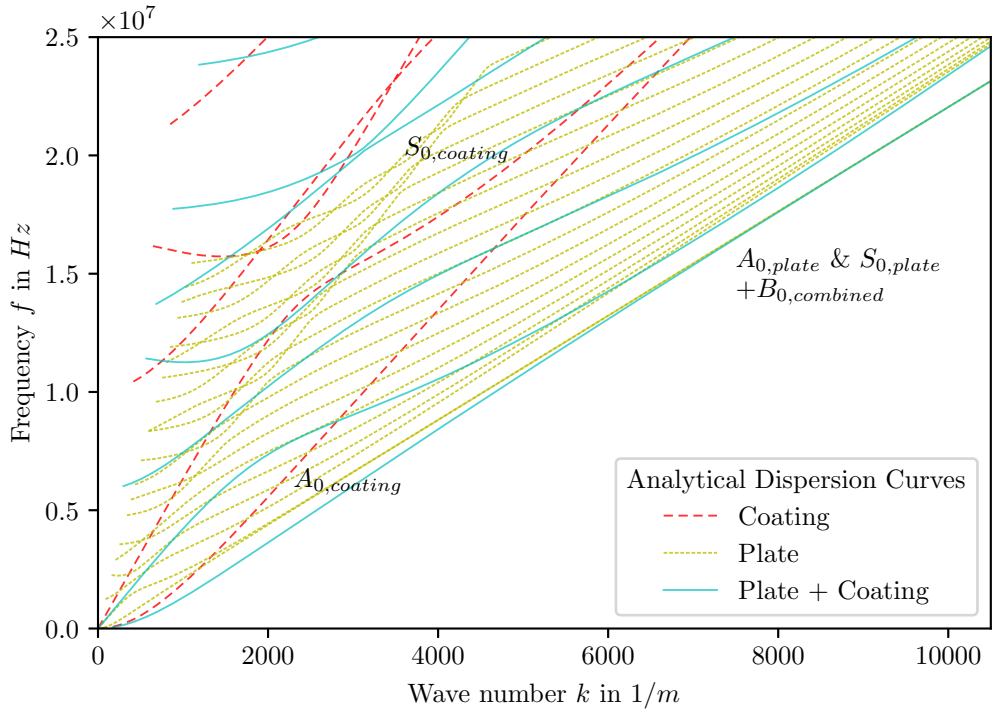


Figure 3.4: Analytical dispersion curves for a 200µm coating (red dashed), 1mm plate (green dashed), and the layered combined system (solid blue).

measurement almost impossible and increases the demands on the simulation accuracy.

In Figure 3.4 the analytically obtained dispersion curves for the coating, the place, and the combined system are shown. The purpose of this research is to obtain information about the coating, and not about the underlying plate. For this, the dominant physics of the coating needs to be extracted without the influence of the coating. This is done by analyzing the dispersive behavior of the specimen and extracting the coatings modes, which change depending on the coatings thickness or other parameter changes, as well as possible.

Since the coating thickness can be very small, the idea is to analyze the change in dispersion curves for high frequencies f with $f \in [10, 25]$MHz and wavenumbers within the range of $k \in [500, 7000]\frac{1}{\mathrm{m}}$. The reason for this is that for these frequencies

and wavenumbers the distance between the lower order modes of the coating (red), i.e. $A_{0,coating}$ and $S_{0,coating}$, and the lower order modes of the plate (green), i.e. $A_{0,plate}$ and $S_{0,plate}$, is maximized. Note that with increasing mode order, less energy is present in the respective mode. The idea is to be able to analyze the lower order modes of the coating without or only with little influence of the modes from the plate. It can be seen that in the proposed frequency and wavenumber range the zero-modes of the coating are present while only higher order modes of the plate show up. Another thought should be noted: moving to higher frequencies and wavenumbers also means approaching the domain where Lamb waves are having less influence and Rayleigh waves, which concentrate their energy close to the boundary/surface, are getting more predominant.

Those proposed phenomena can be seen in the dispersion graphs after simulation like in Figure 3.5. There, high-intensity values are registered close to the origin, because the lower order modes of the coating and of the plate are close and because the most energy is contained for low frequencies and wavenumbers. It can be seen that along the coating's mode $A_{0,coating}$ the intensity values are high for higher frequencies and wavenumbers, and that the modes of the plate shine out only around this area. The location of where the coating's A_0 mode and therefore the higher-order modes of the plate the show up is sensitive to the coating thickness. The area for wavenumbers below $k = 2000\,1/\mathrm{m}$ is not sensitive to the coating's thickness and therefore, is not further analyzed in this work, even though there might be still useful information contained.

3.4.2 Analysis of Simulation Results

In this section, the simulated dispersion curves for different coating thicknesses are discussed. When comparing these dispersion plots from different coating thicknesses there are three main entities to look at: the gradient of the dominant lower order coat-

ing modes, the con-/ divergence of the lower-order coating modes, and the appearance of lower- and higher-order modes of the plate.

In the considered coating thickness range between 10 and 600 microns, coated plates can be assigned to either one of three thickness sets according to their different underlying phenomena. Even though there are discrete values given for each set border, the borders should be considered as smooth and the effect of the respective adjacent set shows up for simulations at the border of the given set limits.

⇒ 10µm - 60µm: Both plate and coating show dominant modes in this area as shown in Figure 3.6. The plate's lower- and higher-order modes are visible almost everywhere above the coatings S_0, which can be seen especially well in Figure 3.6a. The coating is not dominant though but increases its influence for thicker coatings. It can be seen that the A_0 and S_0 modes of the coating are getting detached, diverge, and can be separated at an increasingly higher frequency and wavenumber for decreasing coating thickness.

⇒ 60µm - 300µm: Between 60 and 100 microns the A_0 and S_0 modes of the coating converge stronger (cf. Figure 3.7). This is specifically true for higher frequencies where both modes merge and share the same locations in the frequency - wavenumber domain. For frequencies higher than $f = 10$MHz, many of the plate's higher modes remain visible around the area where the coatings A_0 and S_0 modes are. The gradient of the zero-modes of the coating is increasing for increasing coating thickness.

⇒ 300µm - 600µm: As Figure 3.8 shows, for thicker coatings, the coating modes are getting predominant in the simulated dispersion curves and the A_0 and S_0 modes of the coating are merging together even for small frequencies and wavenumbers, such that only the superposition of both can be seen. Additionally, intensity

is shifted toward the origin with increasing coating thickness. The gradients of the curves are not changing remarkably anymore.

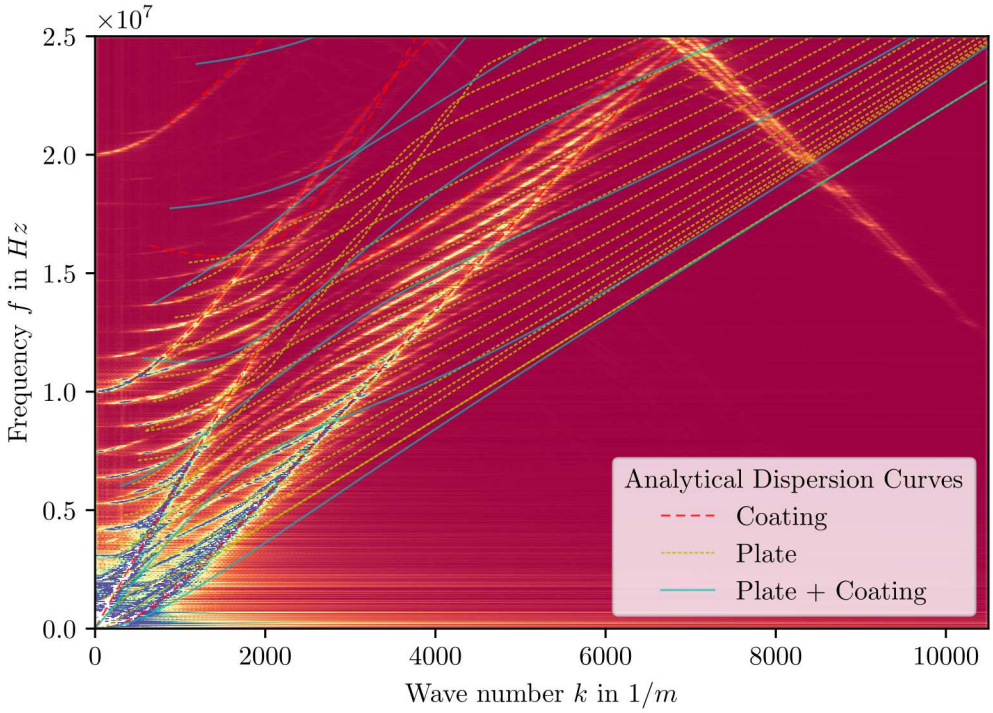
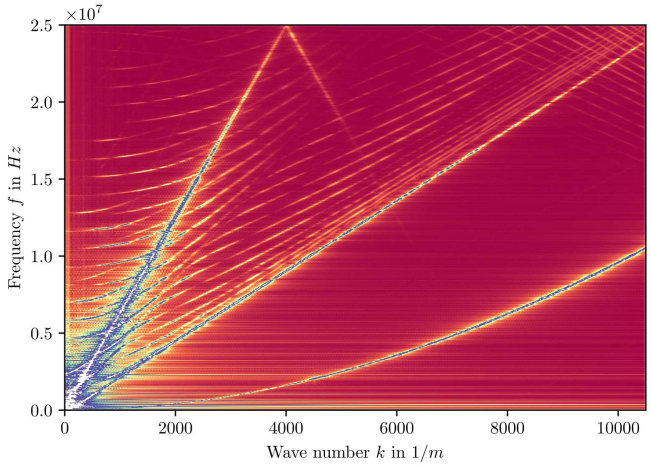
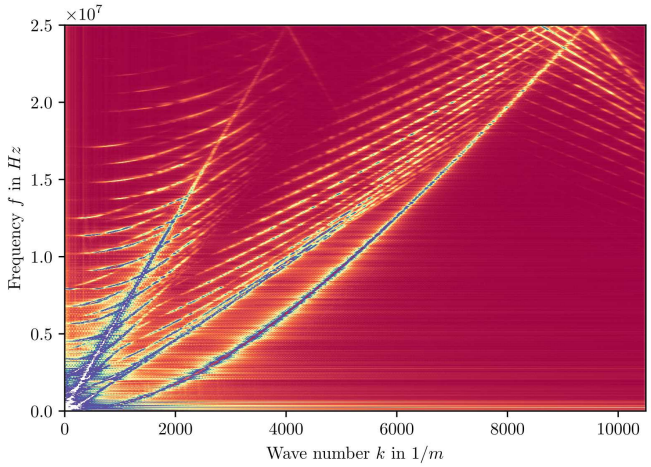


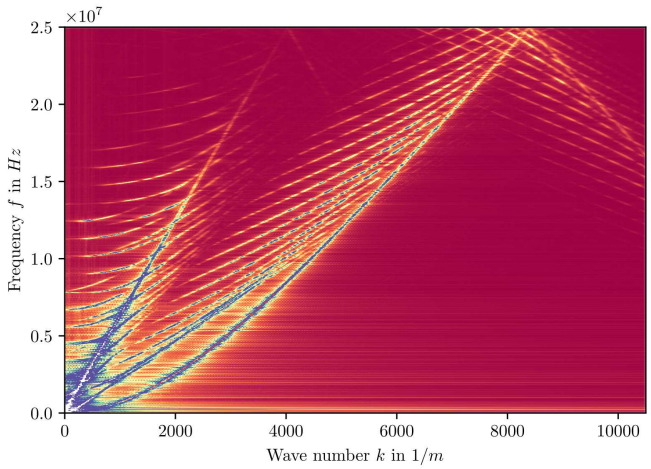
Figure 3.5: Analytical and simulated dispersion curves for a coating of 200μm thickness. Blue values indicate a high intensity while red indicate a low intensity. The yellow line with negative gradient in the top right corner is connected to reflections and is not part of this discussion.



(a) Coating thickness of $h_{coating} = 10\mu m$

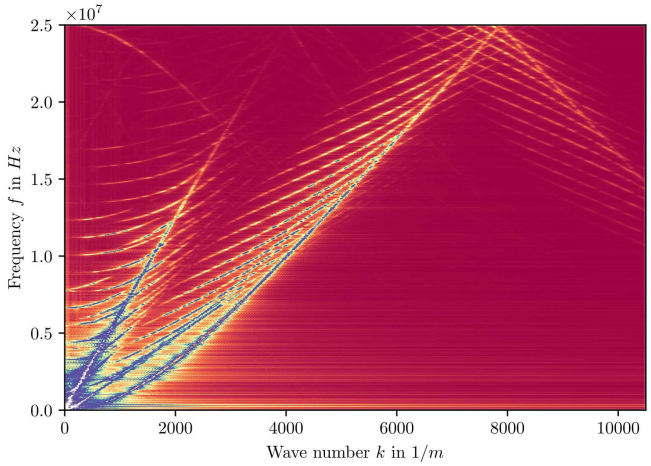


(b) Coating thickness of $h_{coating} = 40\mu m$

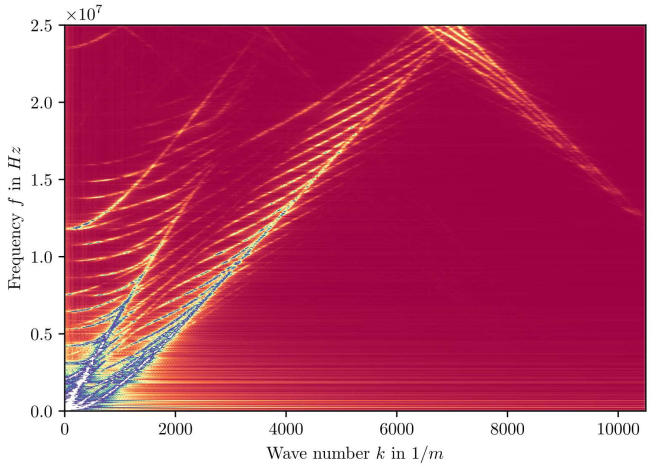


(c) Coating thickness of $h_{coating} = 60\mu m$

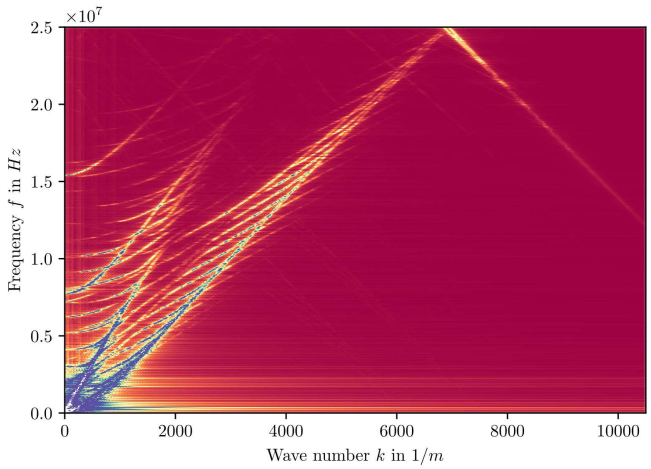
Figure 3.6: Selected simulated dispersion curves for coating thicknesses between $10\mu m$ - $60\mu m$.



(a) Coating thickness of $h_{coating} = 80\mu m$

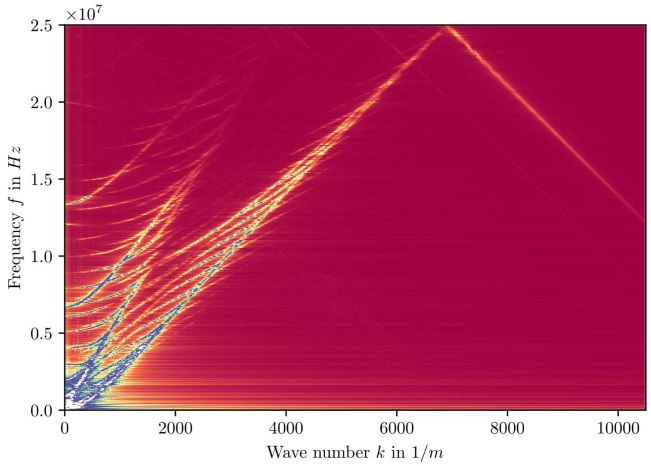


(b) Coating thickness of $h_{coating} = 170\mu m$

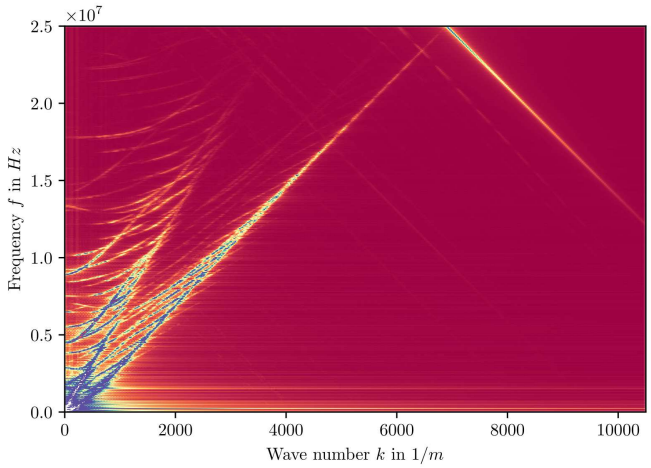


(c) Coating thickness of $h_{coating} = 260\mu m$

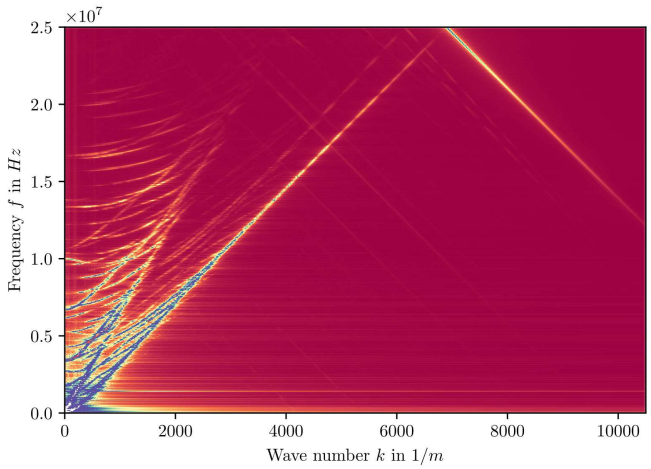
Figure 3.7: Selected simulated dispersion curves for coating thicknesses between 60µm - 300µm.



(a) Coating thickness of $h_{coating} = 300\mu m$



(b) Coating thickness of $h_{coating} = 450\mu m$



(c) Coating thickness of $h_{coating} = 600\mu m$

Figure 3.8: Selected simulated dispersion curves for coating thicknesses between 300μm - 600μm.

CHAPTER 4
MACHINE LEARNING BASED THICKNESS INVERSION

The simulated dispersion curves described in the previous chapter can now be used for a machine learning-based inversion procedure. The goal of this procedure is to classify if a previously unseen dispersion curve belongs to a layered system whose coating is satisfying the condition 'thick enough' or not.

In the following, first, the feature extraction procedure, consisting of non-maximum suppression and function fitting is described. In the second part of this chapter, the extracted features are used and fed into a binary machine learning classifier. Various classifiers are compared and evaluated

4.1 Feature Engineering

The feature extraction consists of two steps: non-maximum suppression to extract the coordinates of high-intensity values around the modes and function fitting to fit a linear function to these points to obtain features from this. Both methods are discussed in the given order the following.

4.1.1 Non-Maximum Suppression

The 2D-FFT serves a two-dimensional array with intensity values in the frequency - wavenumber domain for a given specimen. The modes of the respective specimen are encoded in the 2D-FFT spectrum as coherent points of high intensity. These points need to be extracted. For that, non-maximum suppression (NMS) is used. NMS is a method from computer science and computer vision that presents a solution to extracting the coordinates of these local intensity maxima.

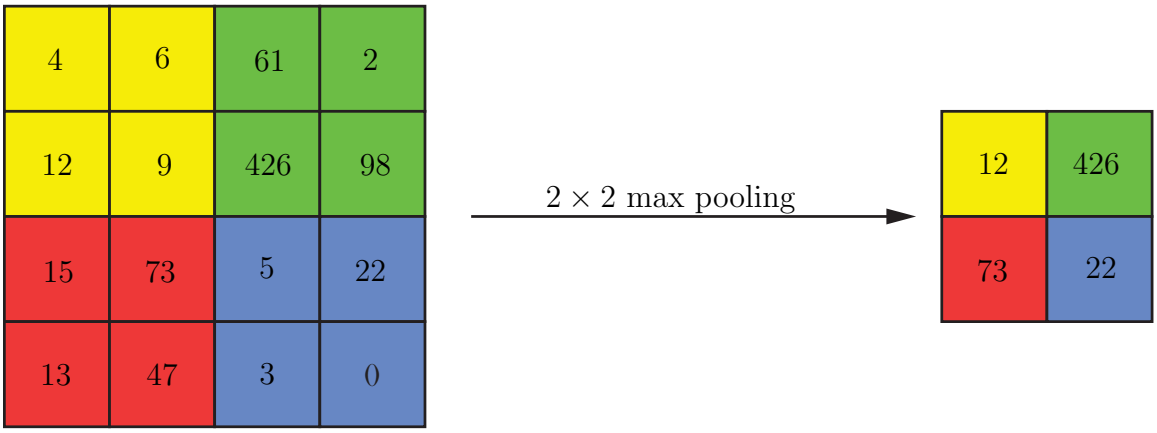


Figure 4.1: Schematic visualization of max-pooling with a kernel of size $\kappa = 2$ and `stride` $= 2$.

The first step of non-maximum suppression is the use of the max-pooling operation. The max-pooling operation reports the maximum output within a rectangular neighborhood which is shown in Figure 4.1. The size of this neighborhood is a parameter which needs to be tuned according to the problem. This research proposes a kernel size of $\kappa = 15$ discrete elements, where the physical size of one element differs between frequency and wavenumber axis and can be calculated by Equation 2.48.

The next step is that a blank copy with the same input dimensions[1] as the dispersion plot is created. The via max-pooling reported maximum value within the rectangular neighborhood is then copied into all element cells of the blank copy which are within the same neighborhood as the max-pooling operator in the original image as shown in Figure 4.2.

In the following, the max-pooling operator is shifted element by element[2] over the entire input image and the two steps above are repeated, such that the copied image looks like a down-sampled version of the original input only with the maximum values within the respective neighborhoods.

[1] Assume that the input is padded accordingly. In this work, zero-padding on each side with length `padding`$= \kappa//2$ is utilized.

[2] Assume that stride is equal to one.

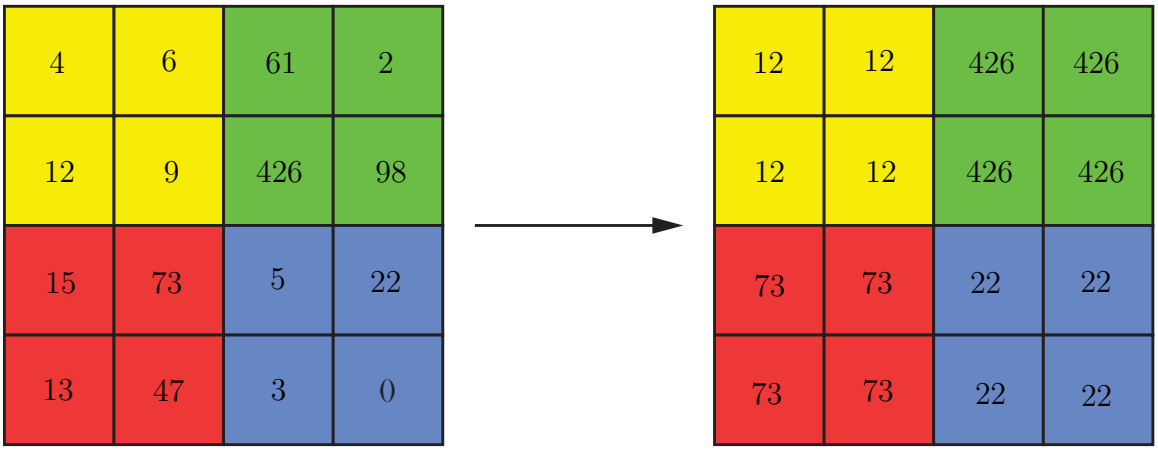


Figure 4.2: Schematic visualization of the copy step during non-maximum suppression with a kernel of size $\kappa = 2$ and `stride` $= 2$.

In the last step, the input image and its copy are element-wise compared. This way, the coordinates of the maxima can be extracted. The extracted maxima from a simulated dispersion curve from a system with a 200µm coating thickness is shown in Figure 4.3. To exclude the extraction of maxima for higher-order modes for small wavenumbers, as well as noise in the overall dispersion graph, non-maximum suppression takes place in a certain wavenumber band-with and within a cone (pink lines) only. Non-maximum suppression is conducted with Numpy [40] and Pytorch [41].

4.1.2 Function Fitting and Feature Extraction

The extracted maximum intensity values from the dispersion curves from the previous section can now be used as an input to a simple linear fitting algorithm. For this, Scipy's [42] optimization package is used. The function to fit is a simple linear function

$$f(k) = a\,k + b, \tag{4.1}$$

with the gradient a and the y-intercept b. The fitting function uses a non-linear least squares algorithm as described in [43]. An example of a fit can be seen in Figure 4.4.

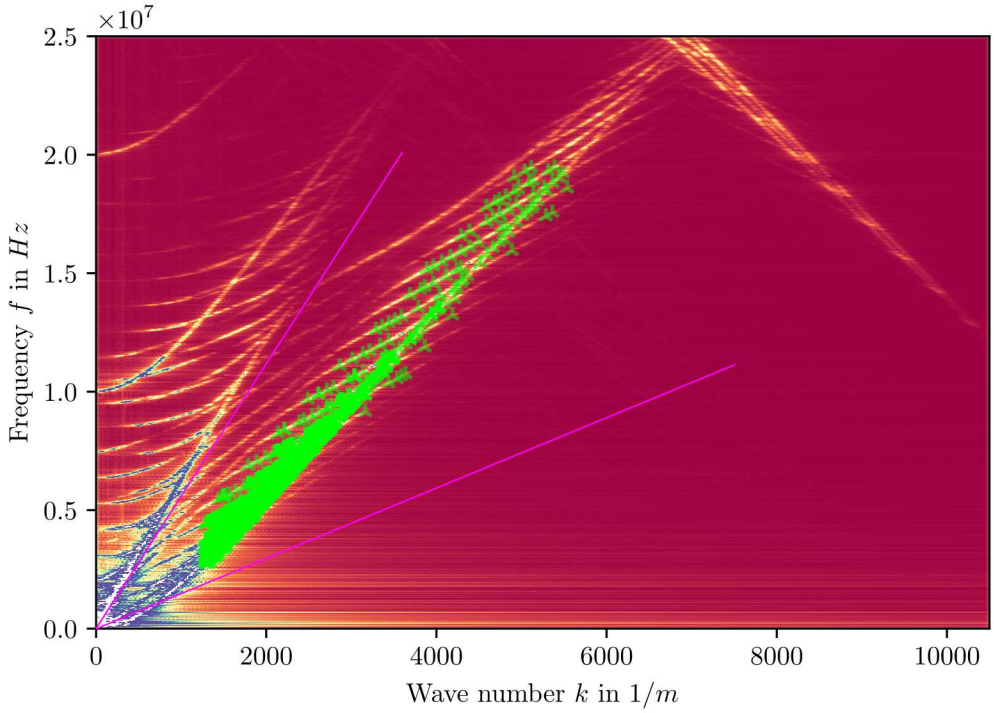
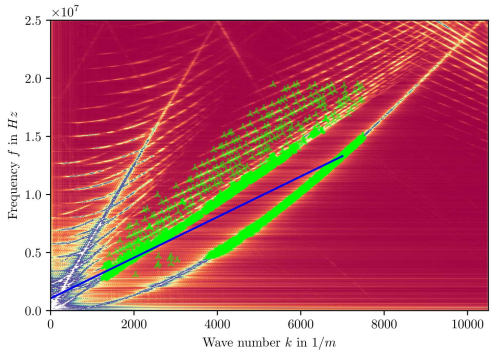
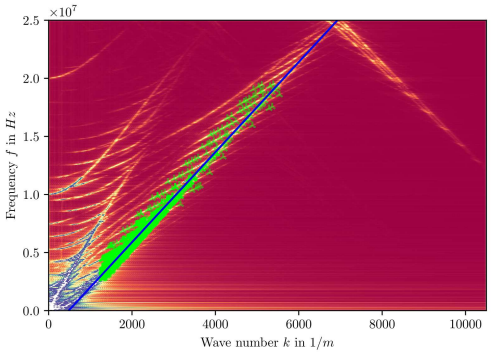


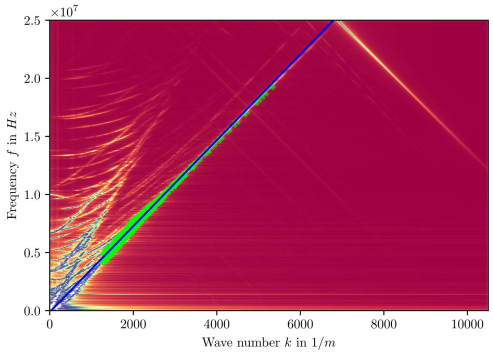
Figure 4.3: Non-maximum suppression for a dispersion curve from a system with a coating thickness of $h_{coating} = 200\mu\text{m}$.



(a) Coating thickness of $h_{coating} = 30\mu m$ with NMS suppression and fitted function.



(b) Coating thickness of $h_{coating} = 200\mu m$ with NMS suppression and fitted function.



(c) Coating thickness of $h_{coating} = 600\mu m$ with NMS suppression and fitted function.

Figure 4.4: Selected simulated dispersion curves with non-maximum suppression and corresponding fit for coating thicknesses between 30µm - 600µm.

This approach works well in the considered thickness range from 10µm up to 600µm and was not tested for thicknesses outside of that range. Physically, the understanding of this approach is that especially the gradient is sensitive to a change in the coating's thickness. The reason for this is that the coating's A_0 mode diverges more and more from its S_0 mode with decreasing coating thickness, 'pulling' the overall gradient to a smaller value. Only for really small coatings below 30µm, the gradient might increase again. This is caused by the coating's A_0 mode leaving the analyzed cone area such that it has no or limited weight on the fit and only

the S_0, which is naturally having a higher gradient, is analyzed. For thick coatings above 300µm the gradient is not changing severely anymore since the zero-modes are converged already.

One thing to note: since Equation 2.6 provides that $k = \frac{2\pi}{c_p} f$, with $c_p \approx c_R$ for high frequencies and wavenumbers, the gradient could be used to estimate the Rayleigh frequency. This is not further investigated within this research.

4.2 Applying Machine Learning Classifier

As already stated earlier, one goal of this research is to develop a procedure that is capable to classify if the uniform coating of a layered specimen has a certain thickness or not. A coated specimen is classified as, i.e. the labels are, *thick enough* or *not thick enough*, respectively. Exemplary and without loss of generality, a coating thickness of $h_{coating} = 200\mu m$ is chosen to be the threshold between *thick enough* and *not thick enough*. The scheme was tested with other thresholds and worked analogously. With this, it is

$$\begin{array}{ccc} \text{not thick enough} & \Longleftrightarrow & \text{thick enough} \\ h_{coating} < & 200\mu m & \leq h_{coating} \end{array}$$

The extracted gradient and y-intercept from the previous section are now used as features for thickness classification. Since only two features are used, a visualization of the complete feature space is possible and can be seen in Figure 4.5. Each data point corresponds to the simulation of one coated specimen with respective coating thickness and is color-coded according to its thickness classification. Even before applying a machine learning classifier, it can be seen that the data points seem to be linearly separable.

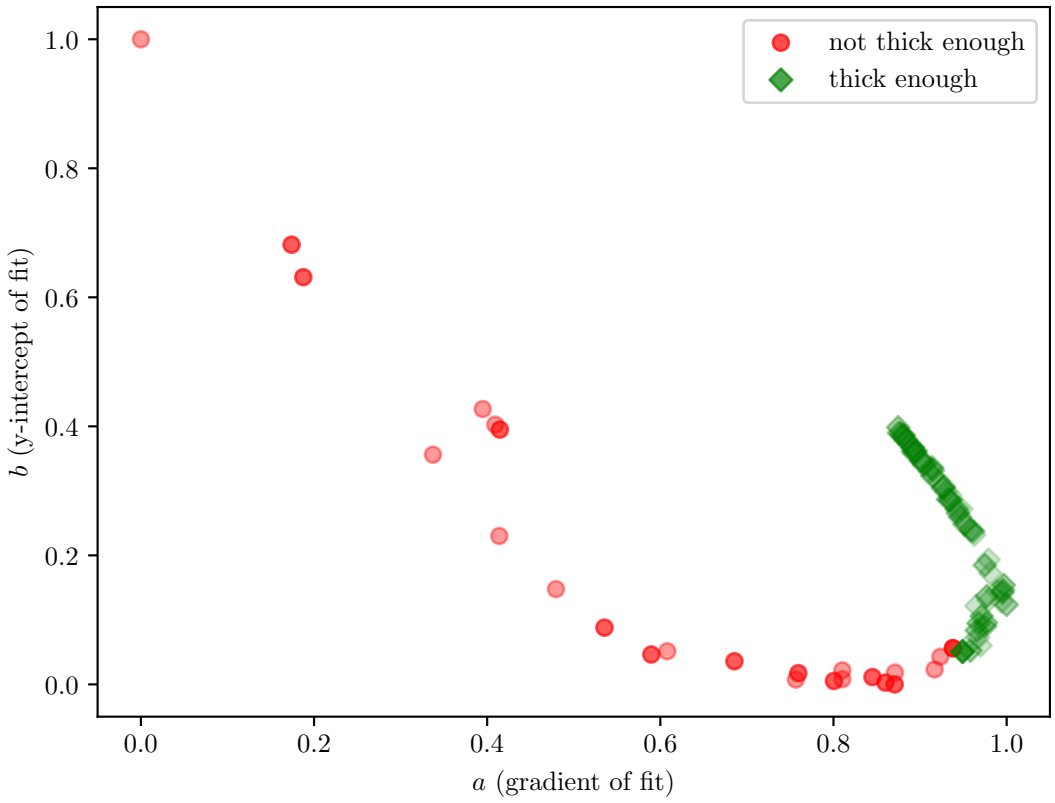


Figure 4.5: Feature plot for dispersion inversion with fitted function of shape $f(k) = a\,k + b$ and labeling according to threshold $h_{coating} = 200\mu m$.

The first step of applying a machine learning classifier is to normalize or standardize the input data. The goal of this is to make sure that the variance of each feature is in the same order of magnitude as the other features-. This ensures that no feature is dominating just because of its input distribution. Each data point $x_{in,i}$ of the input dataset $x_{in,i} \in X_{in}$ is scaled according to

$$x_{scaled,i} = \frac{x_{in,i} - \min(X_{in})}{\max(X_{in}) - \min(X_{in})}. \tag{4.2}$$

With this step, the feature values are normalized to the range between zero and one.

The obtained data can now be fed into various classifiers. The analyzed classifiers are k-Nearest Neighbor (kNN) [44], single layer perceptron [45], Support Vector machines (SVM) with and without Radial Basis Function kernel [46, 47], Gaussian processes (GP) [48] and feedforward networks (MLP) [33, 49]. The algorithms are then evaluated with k-fold cross-validation for $k = 5$.[50] The results after hyperparameter tuning are displayed in Table 4.1.

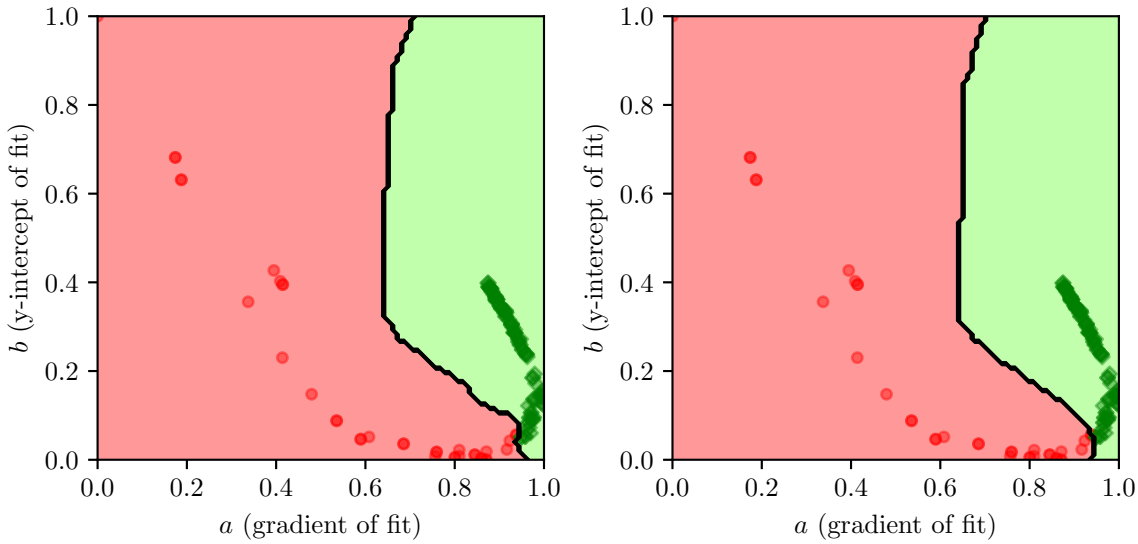
Classifier	Accuracy score	Standard deviation after cross-validation
1-Nearest Neighbor	0.903	0.123
5-Nearest Neighbor	0.917	0.107
Perceptron	0.94	0.074
Linear Support Vector Machines	0.947	0.066
Radial Basis Function Support Vector Machines	0.954	0.062
Gaussian Processes	0.903	0.132
Feedforward Network	0.917	0.107

Table 4.1: Comparison of accuracy and standard deviation after 5-fold cross-validation and hyperparameter tuning.

It can be seen that all classifiers perform well on the given dataset of 134 simulated uniformly coated plates and achieve high accuracy of over 90%. Support Vector

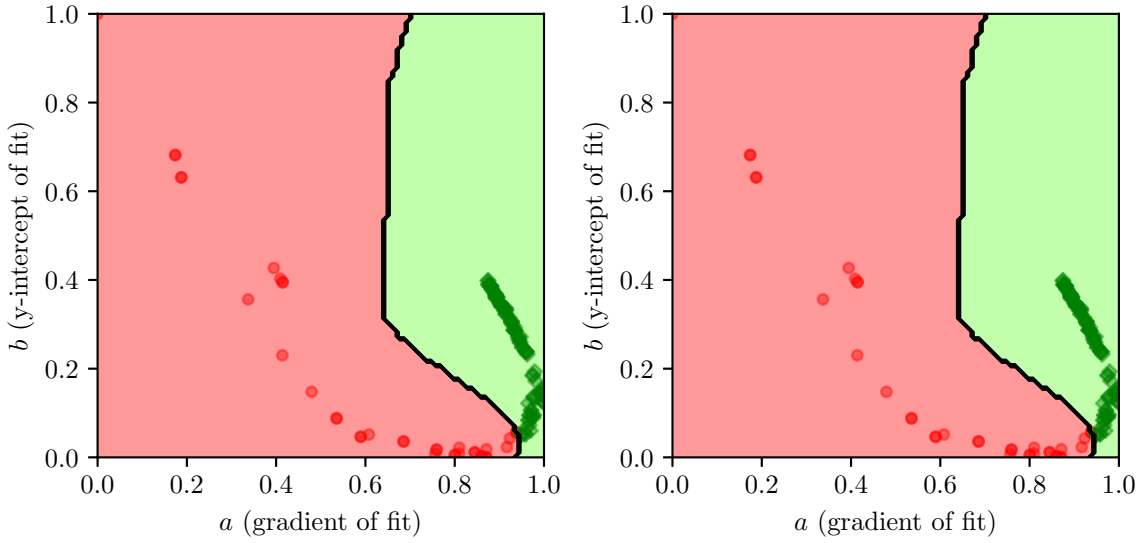
machines, especially Support Vector machines with a Radial Basis Function kernel (RBFSVM), tend to have the highest accuracy with the smallest standard deviation after cross-validation, while one-Nearest Neighbor (1NN) and Gaussian Processes perform not as well.

Figure 4.6 provides a visualization of four selected classifiers with the lowest (1NN in Figure 4.6a) and highest (RBFSVM in Figure 4.6c) accuracy, where the red area describes where a feature combination would be labeled as *not thick enough*, and the green area where a feature combination would be classified as *thick enough*. In this research, scikit-learn [51] was used for the implementation of the machine learning algorithms and their evaluation.



(a) kNN classifier with $k = 1$.

(b) kNN classifier with $k = 5$.



(c) SVC classifier with RBF kernel and regularization parameter `C` $= 4$ and kernel coefficient `gamma` $= 2$.

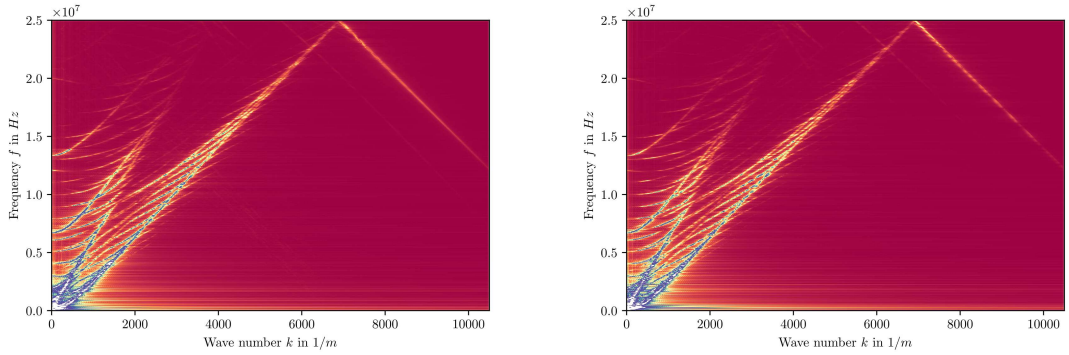
(d) MLP classifier with three hidden layers of size $(20, 50, 30)$ and ReLU as activation function.

Figure 4.6: Selected machine learning classifiers for a two-dimensional thickness classification problem.

CHAPTER 5

DEEP LEARNING BASED UNIFORMNESS INVERSION

The machine learning approach for inversion from the previous section works well for a uniform thickness but reaches its limits when a non-uniformness is incorporated in the coating of a simulated specimen. A non-uniform simulation model was described in Figure 3.3. Comparing the dispersion curves of a non-uniform coating with its uniform counterpart, shown in Figure 5.1 exemplary for a coating thickness of $h_{coating}$ = 200µm, a depth of the gap as the non-uniformness of h_{gap} = 100µm (which is a reduction of the coating thickness by half) over a length of $l_{gap=}$1.2cm, shows, that there is a difference in the dispersion figures, but neither a fitted gradient or y-intercept would change largely. This is not different for other coating thicknesses or gap depths. Hence, the features used in the machine learning-based inversion process are not sensitive to a non-uniformness in the coating.



(a) Coating thickness of $h_{coating}$ = 300µm without non-uniformness.

(b) Coating thickness of $h_{coating}$ = 300µm with a gap of depth h_{gap} = 200µm as a non-uniformness.

Figure 5.1: Comparison of dispersion curves from a specimen with *uniform* and *non-uniform* coating thickness

Not merely the manually crafted features from the previous chapter do no work for in this problem. In general, the changes in a dispersion graph caused by a non-

uniformness are rather small. Even a tedious manual analysis of comparing various coating thicknesses and gap depths did not result in the recognition of any quantifiable pattern, such that manual feature engineering could have taken place. Thence, the idea is to let a convolutional neural network (CNN) learn the features by itself.

In the following sections the data acquisition pipeline and used dataset for the CNN is described. Then an overview about the developed and used network architectures is given. Last, the network's performance is evaluated.

5.1 Data Acquisition and Dataset Statistics

Since the features for classification should be learned by the CNN itself, it requires significantly more data to train than the simple machine learning classifiers from the previous section. To obtain the data for CNN training, 670 simulations with an average simulation and post-processing CPU time of 15 hours were conducted on the Georgia Tech PACE cluster [39]. An overview of the simulation space can be found in Figure 5.2. In this figure, each dot represents one conducted simulation of a specimen with given coating thickness $h_{coating}$ and depth of the gap h_{gap} as a measure of the extend of the non-uniformness. Simulations with a non-uniformness, i.e. a gap depth bigger than zero, are coded with red circles while uniform simulations are coded with a green diamond.

5.1.1 Data Tightening

With these simulations, a problem arises: there are a lot of different simulations possible with non-uniformness for a given thickness since there are different variations for a non-uniformness, but only one version with uniform thickness per thickness is possible. This leads to a distribution of the input data as shown in Figure 5.2 with almost 80% of the simulations belonging to non-uniform coatings. Additionally, most simulations are falling into a coating thickness between 200µm and 300µm. A non-

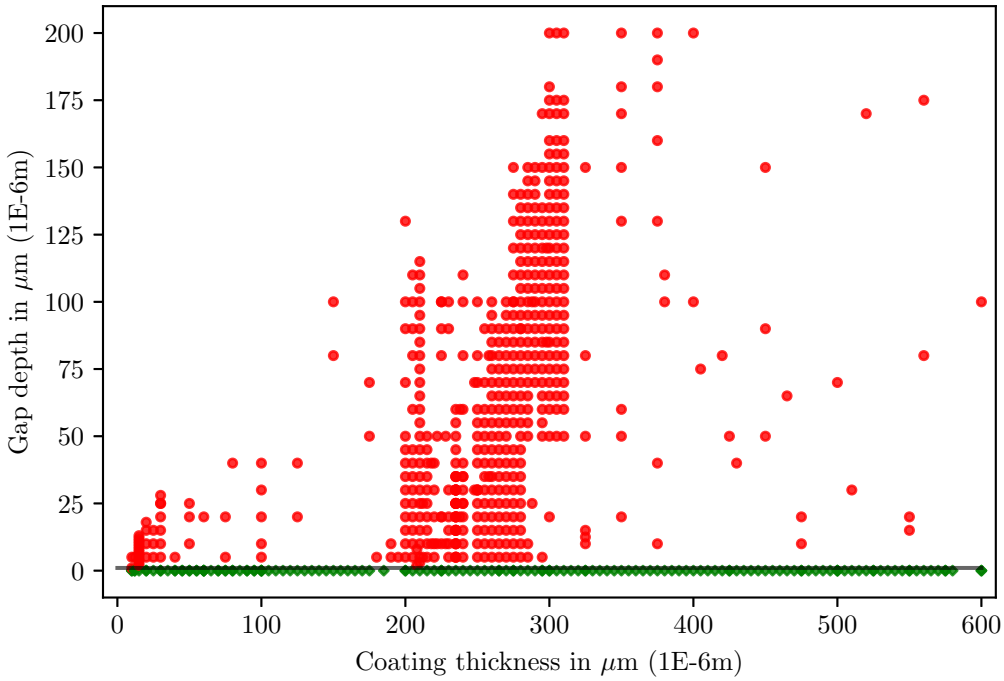


Figure 5.2: Simulation space for all 670 simulations with 148 (= 22.09%) simulations which are uniform (green diamonds) and 522 (= 77.91%) simulations with non-uniformness (red circles).

uniform data distribution like this makes learning harder for the network. To face this problem, a more evenly distribution of simulations over the entire domain is created by restricting the data set using a random selection of simulations only for training as it is shown in Figure 5.3.

5.1.2 Data Preprocessing

The first step of the development of a neural network is specifying the shape of the data at the input side. The Fourier transformed data from the raw displacement data of the simulations is stored in a file with high resolution, but a huge size too. To decrease the size of the input, dispersion curves are plotted from the 2D-FFT transformed displacement data and are stored as an image in PNG format. This

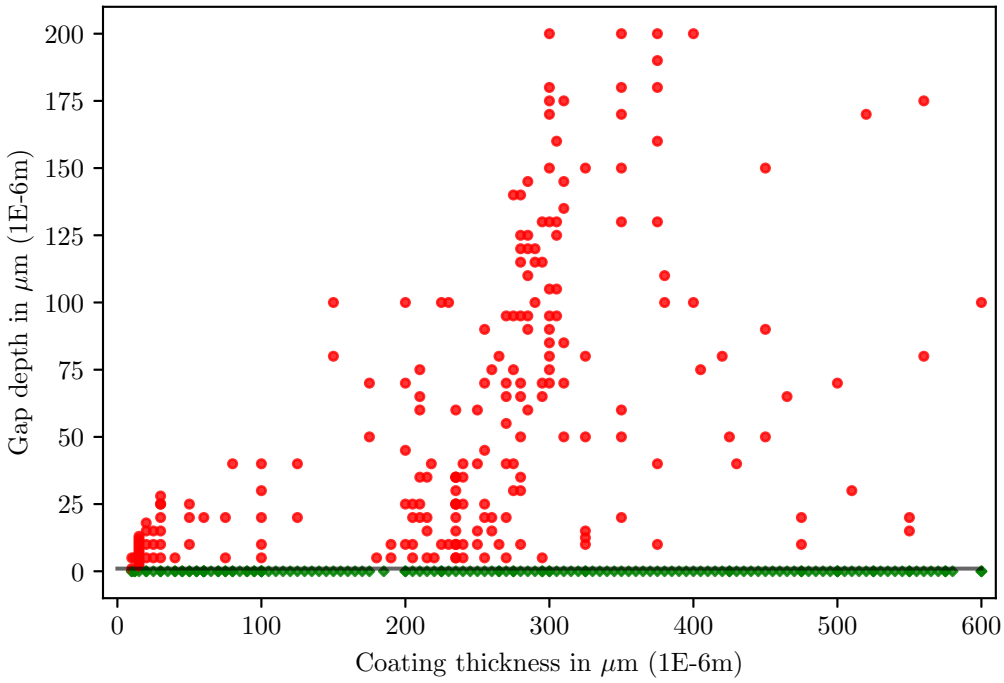


Figure 5.3: Simulation space for randomly selected 363 simulations with 148 (= 40.77%) simulations which are uniform and 215 (= 59.23%) simulations with non-uniformness.

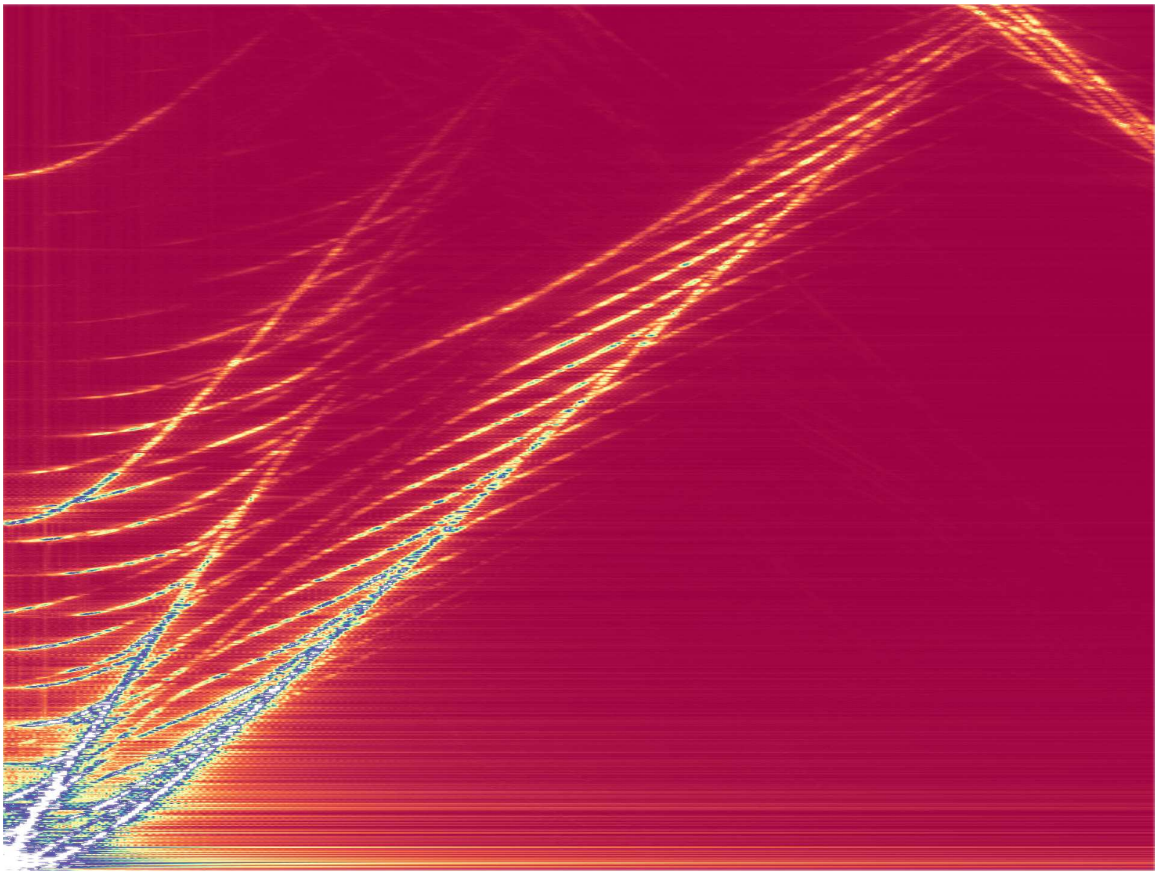


Figure 5.4: Example PNG network input of contour plot with three-channel RGB colors of a specimen with coating thickness of $h_{coating} = 200\mu m$ for a frequency f in the range $0\text{MHz} \leq f \leq 25\text{MHz}$ and a wavenumber k with limits $0\frac{1}{m} \leq k \leq 8000\frac{1}{m}$.

is feasible since PNG supports lossless compression as described in [52]. With this procedure, the size of the input data can be reduced by a factor of 30. An example of this is shown in Figure 5.4. The intensity values from the Fourier transform are color-coded as in previous dispersion plots and stored in this way with three color channels on an RGB scale. This approach is valid since the network does not care about absolute values and learns differences between the provided dispersion plots.[1] Since PNG files are generally referred to as *images*, this work will use the word *images* to refer to the dispersion curve inputs to the neural network too.

[1]Note that the training process takes actually takes place on grayscale images as described in the next section.

The created PNG images are then randomly shuffled and assigned to the train and the evaluation set, such that 70% of the data is used for training and the remaining 30% are used for testing.

5.2 Network Design

The network learning process consists of training on the training dataset and testing on the test set. Since the amount of data available, i.e. the number of simulations conducted, is fairly limited, the evaluation and test set are merged into one set.

The network by itself consists of two parts: data loading and passing the loaded input data through the convolutional layers. Both will be described in the following.

5.2.1 Data Loading

Like with most neural networks, the proposed networks in this work use batches of input images for training. For most training in this research, a batch size of 32 or 48 was used. The process of loading a random batch of images, passing them through the network, calculating the loss of each image, differentiating loss with respect to the network weights, and updating them accordingly, is referred to as one *epoch* of training.

Within each epoch, each three-channel input image is loaded and converted to a grayscale image as shown in Figure 5.5. With this conversion to a single-channel representation of intensities, the ambiguity of color selection during image creation from the preprocessing step is eliminated. A further advantage of using only one channel is that the network needs to process a smaller amount of data, but with the same information content.

After conversion to a one-channel image, the image is resized to a rectangular input of a given resolution. This research proposes an input resolution of 1024×1024 to capture even smaller changes in the image, but smaller input resolutions might be

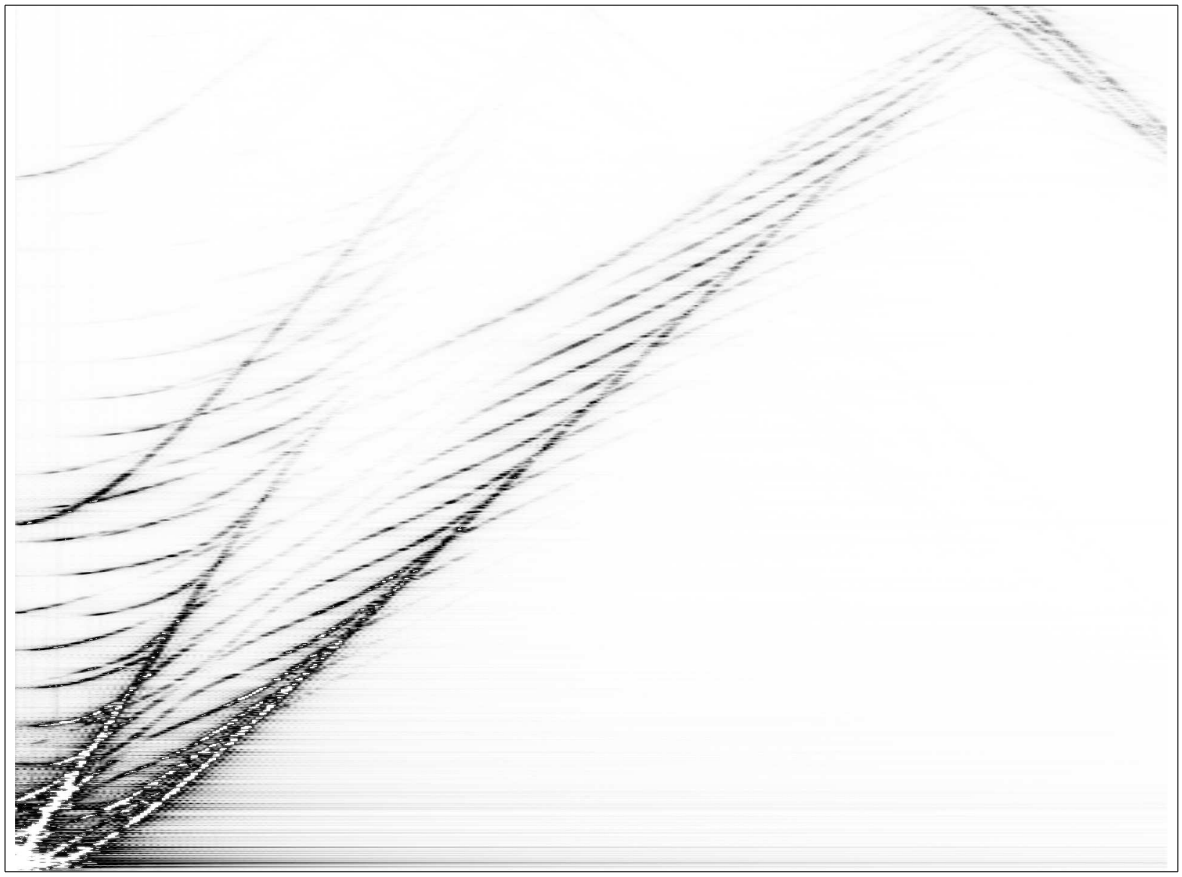
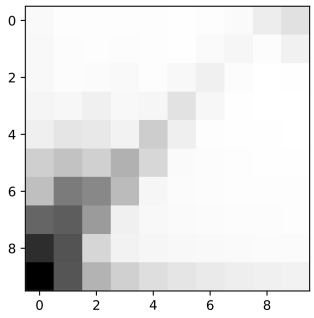
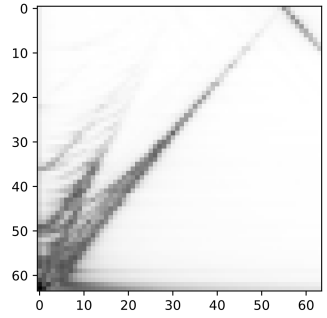


Figure 5.5: Example network input of contour plot with one channel grayscale colors of a specimen with coating thickness of $h_{coating} = 200\mu m$ for a frequency f in the range 0MHz $\leq f \leq$ 25MHz and a wavenumber k with limits $0\frac{1}{m} \leq k \leq 8000\frac{1}{m}$.

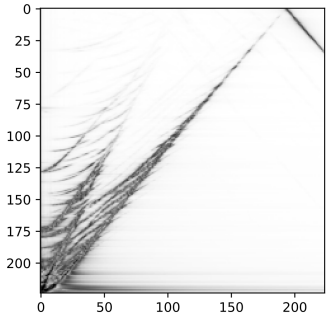
feasible according to the problem. It needs to be noted that the input resolution is a critical hyperparameter of the learning process. Using a resolution that is too small removes too much high-frequency information in the image, while a resolution chosen as too big slows down the learning process a lot. Different input resolutions for the same dispersion data are shown in Figure 5.6.



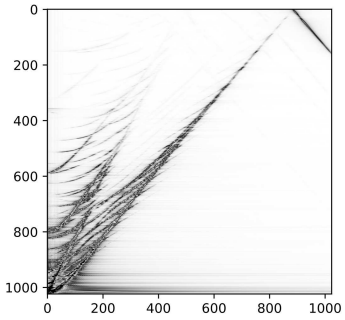
(a) 10×10 pixel resolution input image.



(b) 64×64 pixel resolution input image.



(c) 224×224 pixel resolution input image.



(d) 1024×1024 pixel resolution input image.

Figure 5.6: Examples of different network input resolutions for the same simulated dispersion curves.

5.2.2 Network Architecture

Within this research, two convolutional neural network architectures have been analyzed: *SimpleWaveInvNet* and a modified variant of *ResNet*. In the following, the detailed architecture of SimpleWaveInvNet and modified ResNet is further discussed.

SimpleWaveInvNet

SimpleWaveInvNet is a comparable simple network for thickness inversion developed in this research to capture small changes in the input by using the smallest number of parameters possible. It consists of three layer parts:

- Convolutional layers, consisting of four 2D convolutional layers, 2D max-pooling after the first, third, and fourth convolutional layer, ReLU as activation function after each convolutional layer, a dropout layer in front of the last convolutional layer and a 2D batch norm after the last convolutional layer.
- Average pooling layer which conducts 2D adaptive average pooling to a 5×5 output. This layer allows using of variation in the input size to the network without the need of restructuring the entire network architecture.
- Fully connected layers, comprising three fully connected layers with a dropout layer in front of the first and second fully connected layer, and a ReLU layer behind these two layers.

The loss function for the SimpleWaveInvNet is chosen to be the negative log-likelihood, or *NLLLoss*[53], which is a softmax activated version of the for two class classification established Cross-Entropy loss function. A schematic sketch of SimpleWaveInvNet can be found in Figure 5.7.

ResNet18

Additionally to SimpleWaveInvNet, a modified version of the popular CNN backbone ResNet has been used. One of ResNet's idiosyncrasies is its skip or shortcut connections, which allow the data to *skip* one or more layers so that the input of it will then be added to the output of it for further processing. This allows to reduce the training error for deep networks as well as it improves the accuracy since it allows to train

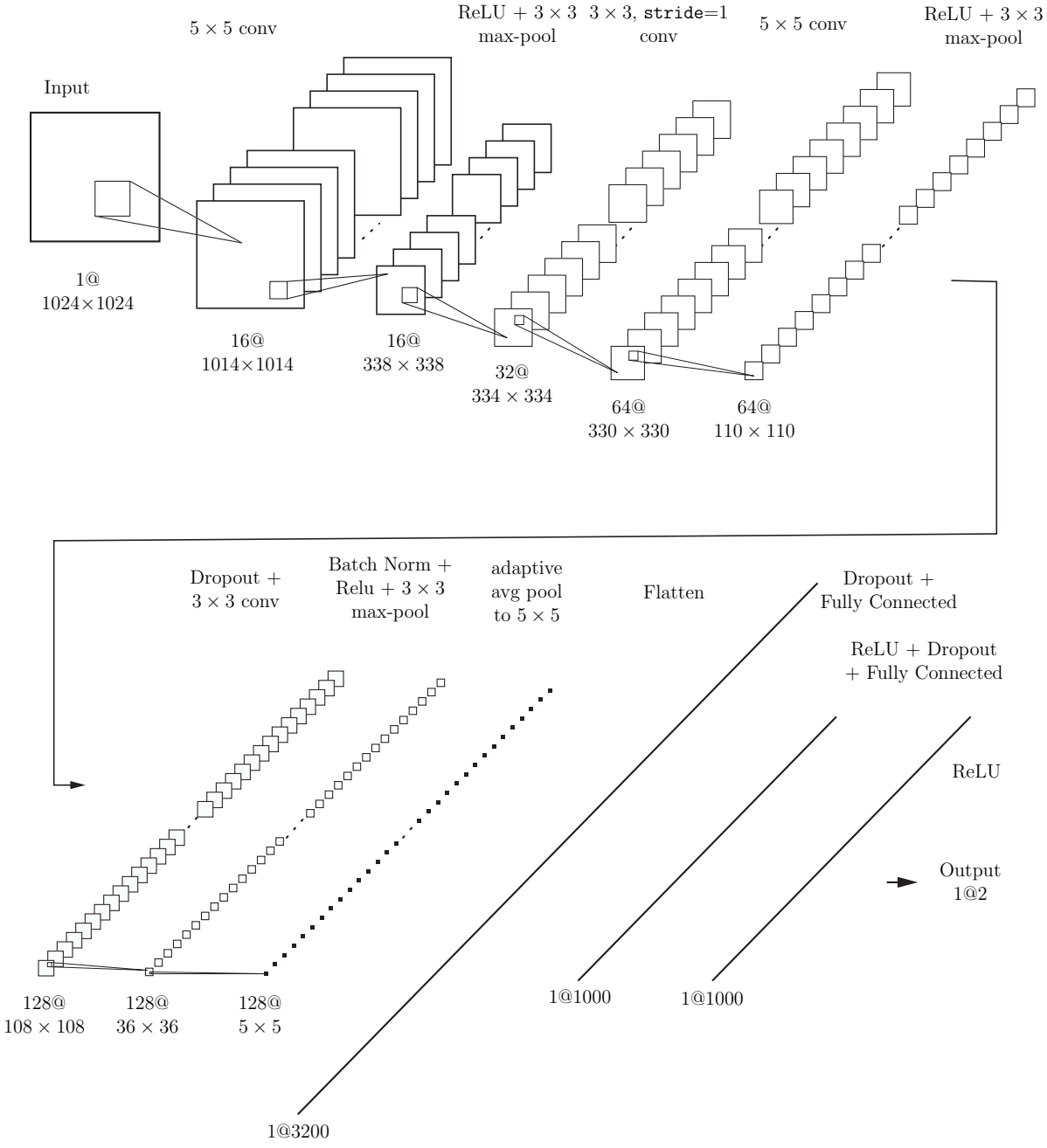


Figure 5.7: Schematic Drawing of SimpleWaveInvNet.

deeper networks. For a deeper understanding of ResNet, the author recommends [54] for further reading.

In this research, ResNet18, which means a ResNet version with 18 layers is utilized. The architecture is forked and the last linear layer is truncated. Instead of the linear layer with 1000 outputs from the original ResNet18, an adaptive average pooling layer is installed to allow the network to be capable to operate on input data with varying sizes. Behind the adaptive average pooling layer, a linear layer maps the network outputs to the two output classes.

For this network, the chosen loss function is the *Cross-Entropy loss*. ResNets with more layers have been tested on the given dataset but did not succeed in the learning process because of the limited size of the training set.

5.3 Network Performance

The network performance is evaluated for SimpleWaveInvNet and ResNet18 separately. First, SimpleWaveInvNet is discussed, followed by a discussion of why ResNet18 might not have learned well.

Next, accuracy is used as a performance metric. Accuracy is simply defined as the number of correct predictions divided by the total number of predictions. Since this work analyzes a binary classification problem, accuracy can be defined by

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

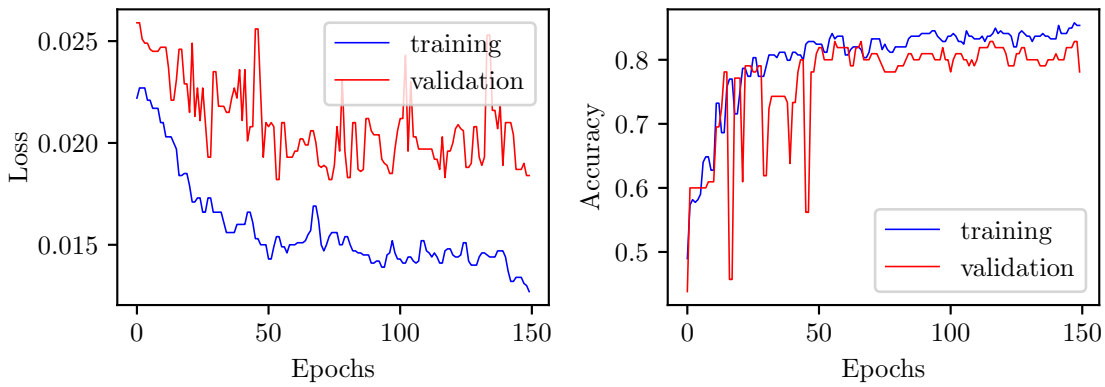
where TP = *true positive*, TN = *true negative*, FP = *false positive*, and FN = *false negative*.

5.3.1 SimpleWaveInvNet

SimpleWaveInvNet was trained over 150 epochs with a batch size of 32. Figure 5.8 shows the training loss history (a) and training accuracy history (b) for both the training (blue) and the validation set (red).

It can be seen that the training error for both training and validation set is decreasing fast in the beginning and then decreases slower until the end of the training process. The shape of the loss curves complies with the expected learning curves of a neural network.

The accuracy is increasing heavily in the beginning and then reaches a plateau, both for training and validation set. Again, the overall learning curve complies with the expected shape. It needs to be noted that the validation accuracy varies way more than the accuracy of the training set, which accompanies the high noise in the validation loss. A reason for this might be the limited data set size as well as a batch size chosen to be too small. With this, some validation sets might not be a good representation of the overall dataset.



(a) Training loss history.

(b) Training accuracy history.

Figure 5.8: SimpleWaveInvNet training performance after 140 epochs and median filtering.

In Figure 5.9, a confusion matrix from the evaluation set for SimpleWaveInvNet is displayed. The confusion matrix shows the ground truth labels on the vertical and

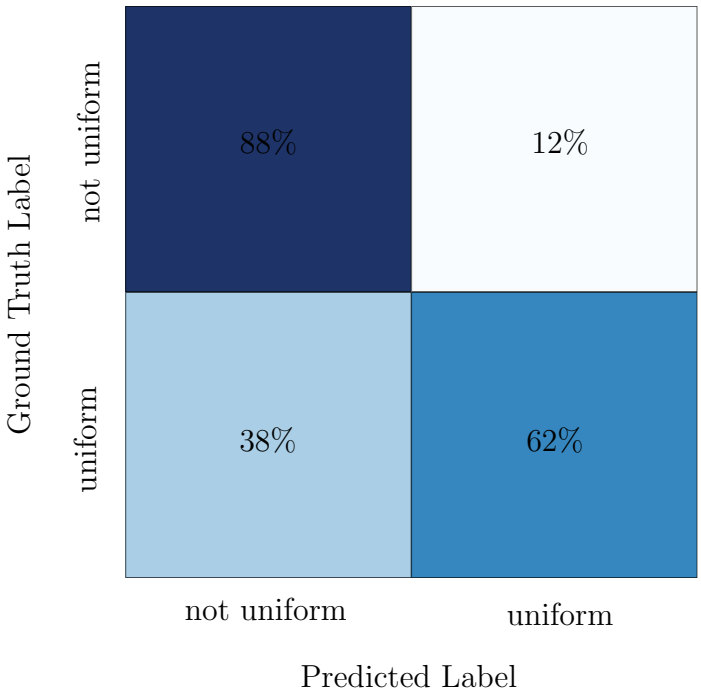


Figure 5.9: Confusion matrix for SimpleWaveInvNet.

the predicted labels on the horizontal axis. An optimal network would have 100% in the top left corner, meaning all data which is not uniform is classified as not uniform, and vice versa with uniform classification on the bottom right. For SimpleWaveInvNet, the majority of not uniform samples are classified correctly. Classifying when a sample actually has a uniform coating is not as unambiguous as for the non-uniform case, but shows a better performance than 50% still. This way is demonstrated that SimpleWaveInvNet is able to provide a uniform/non-uniform thickness classification.

Visualizing the classification labels of the test set after training shows that SimpleWaveInvNet struggles to classify correctly for coating thicknesses between 200µm and 300µm, but classifies most data above that range correctly. This can be seen in Figure 5.10. A reason for this might be that the dataset is not uniformly distributed over the entire parameter set. This way the network learns a shortcut: it learns the gradient relation from the previous chapter and learns that most data points over a certain coating thickness are more likely to be uniform.

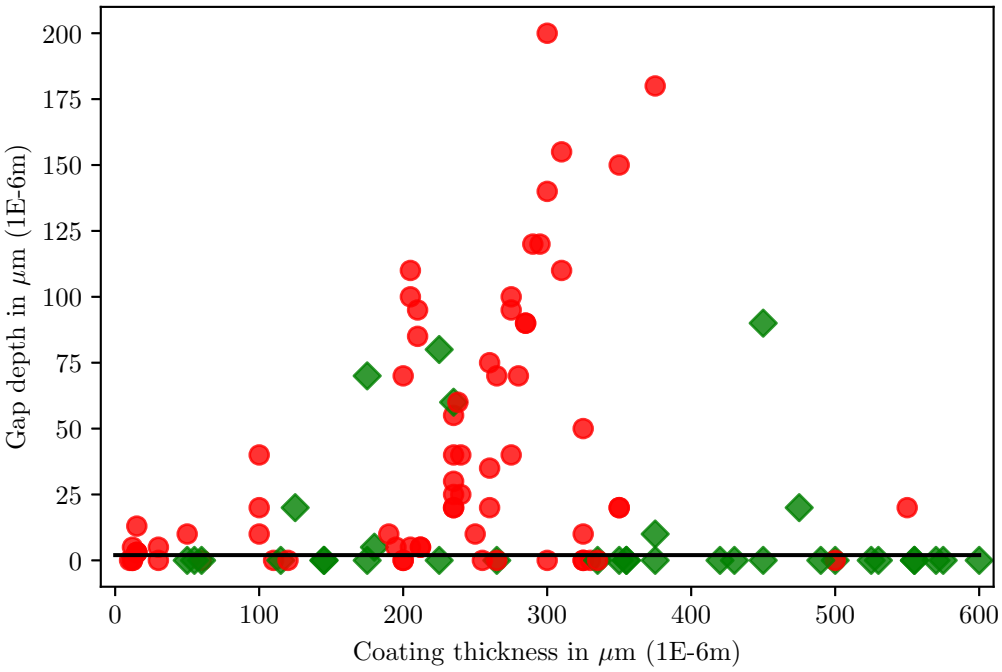
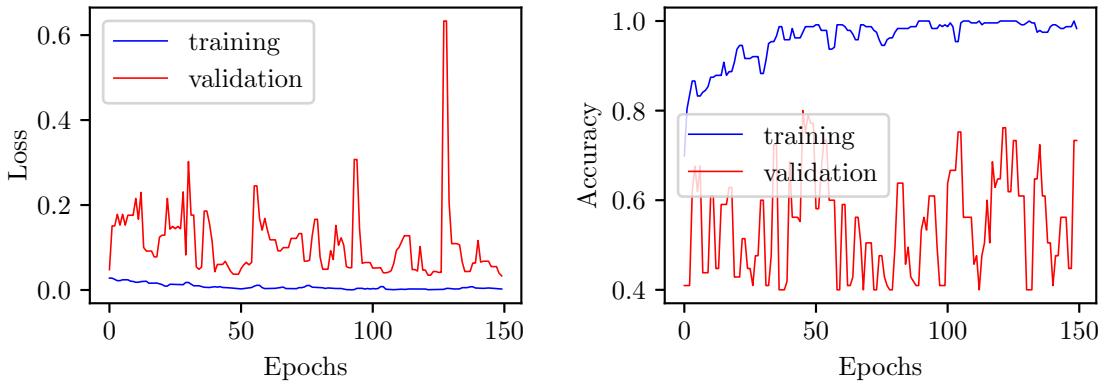


Figure 5.10: Classification of SimpleWaveInvNet for test set of 105 simulations with labels provided by network (green diamonds = uniform) and (red circles = not uniform).

5.3.2 ResNet

The same performance measures as for SimpleWaveInvNet are used for an evaluation of ResNet18. Comparing the training loss and training accuracy from Figure 5.11 shows that the validation loss and accuracy is fluctuating largely. Both validation measures do not show any expected and needed behavior, while the training loss and accuracy have the expected shape.



(a) Training loss history of ResNet for 140 epochs after median smoothing.

(b) Training accuracy history of ResNet for 140 epochs after median filtering.

Figure 5.11: SimpleWaveInvNet training performance.

Visualizing the network performance on the testing dataset in Figure 5.12 shows, that the network learns to simply classify everything as uniform. A reason for this bad performance could be the small dataset size for training combined with a network which has too many parameters to train. Additionally, only a smaller batch size of 20 was possible due to storage limitations that might have a negative impact on the training process as well.

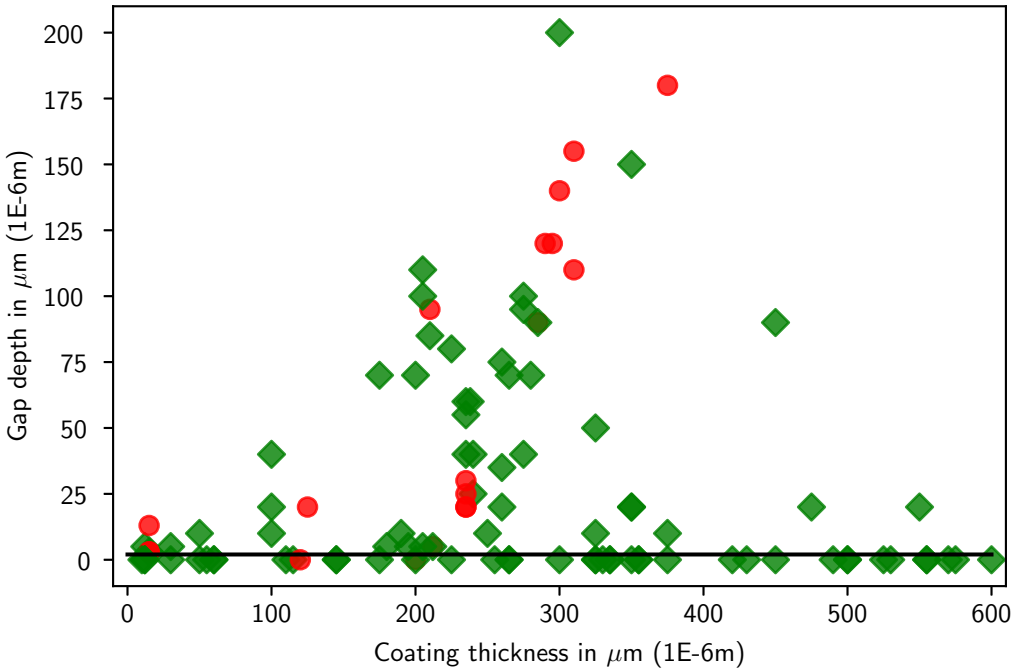


Figure 5.12: Classification of ResNet for test set of 105 simulations with labels provided by network (green diamonds = uniform) and (red circles = not uniform).

CHAPTER 6
CONCLUSIONS

The goal of this work is to obtain information about the material properties of a coating in a layered system with a plate via machine and deep learning. Therefore, a coating's thickness and uniformness are examined. This work shows (1.) that a machine learning approach is highly capable to invert the coating thickness and (2.) that a deep learning approach is capable to classify if there is a non-uniformness in form of a gap in the coating.

First, in this research, the forward problem is solved with a finite element analysis approach. After applying signal processing, the outcome of the forward problem is a frequency - wavenumber dispersion representation with characteristic information. To obtain information of the change in dispersive behavior for a variation in the coating thickness, various systems with varying coating thickness are simulated. Conducting a set of simulations with various uniform coating thicknesses shows that coating thicknesses can be classified into one of three groups of different modal behavior. The simulated dispersion representations are then used as an input for the inversion procedure.

For a thickness inversion, a machine learning based-approach is developed to classify the thickness of uniform coatings. For this, various machine learning classifiers are investigated. It is shown that the proposed machine learning approach is capable of successfully classifying uniform coating thicknesses. Additionally, it is shown that this approach reaches its limit for non-uniform coatings. The complexity is here that the material properties of coating and plate are comparably similar. This leads to only minor changes in the respective dispersion curves when a major non-uniformness is introduced into the coating.

To overcome this problem and classify the uniformness of a coating, a deep learning-based approach is proposed. The deep learning approach uses convolutional neural networks, hence his approach is data-driven. It is demonstrated that a convolutional neural network is capable of learning if a given specimen contains a non-uniformness in the coating or not for the given material combination in the investigated coating thickness range.

Recommendations for future work include increasing the training dataset, i.e. simulation of data more uniformly over the entire simulation space. For the input to the network, a better representation can be developed. This can be done either by developing a method that creates the network input data directly from the simulated displacement data without creating a dispersion plot in PNG format first, or by improving the network input after the creation of the dispersion graph. Since data underneath the $A0$ and $S0$ modes is not containing physically meaningful information, learning an additional neural network as an encoder could decrease the input size of the convolutional neural network and increase training performance this way. To increase the receptive field of the CNN, adding convolutional layers according to the pyramid scene parsing network [55] is another recommendation. Finally, it is recommended to evaluate the model on real experimental measurements

REFERENCES

[1] M. R. Karim, A. K. Mal, and Y. Bar-Cohen, "Inversion of leaky lamb wave data by simplex algorithm," *The Journal of the Acoustical Society of America*, vol. 88, no. 1, pp. 482–491, Jul. 1990.

[2] J. Stolzenburg, J. W. Doane, J. Jarzynski, and L. J. Jacobs, "Near field inversion method to measure the material properties of a layer," *NDT & E International*, vol. 36, no. 7, pp. 523–533, Oct. 2003.

[3] J. Koreck, C. Valle, J. Qu, and L. J. Jacobs, "Computational Characterization of Adhesive Layer Properties Using Guided Waves in Bonded Plates," *Journal of Nondestructive Evaluation*, vol. 26, no. 2-4, pp. 97–105, Oct. 2007.

[4] J. Koreck, "Computational characterization of adhesive bondproperties using guided waves in bonded plates," M.S. thesis, School of Civil and Environmental Engineering, Georgia Institute of Technology, Dec. 2006.

[5] Q. Kong, D. T. Trugman, Z. E. Ross, M. J. Bianco, B. J. Meade, and P. Gerstoft, "Machine learning in seismology: Turning data into insights," *Seismological Research Letters*, vol. 90, no. 1, pp. 3–14, Nov. 2018.

[6] Y. Wu and Y. Lin, "Inversionnet: A real-time and accurate full waveform inversion with cnns and continuous crfs," Oct. 2018. arXiv: 1811.07875 [eess.SP].

[7] R. Rojas-Gómez, J. Yang, Y. Lin, J. Theiler, and B. Wohlberg, "Physics-consistent data-driven waveform inversion with adaptive data augmentation," Sep. 2020. arXiv: 2009.01807 [cs.LG].

[8] Y. Ren, X. Xu, S. Yang, L. Nie, and Y. Chen, "A physics-based neural-network way to perform seismic full waveform inversion," *IEEE Access*, vol. 8, pp. 112 266–112 277, 2020.

[9] J. L. Rose, *Ultrasonic Guided Waves in Solid Media*. Cambridge University Press, 2014.

[10] K. F. Graff, *Wave Motion in Elastic Solids*. Dover Publications Inc., 1991, ISBN: 0-486-66745-6.

[11] J. D. Achenbach, *Wave Propagation in Elastic Solids*. Elsevier Science & Techn., Jan. 2016, 440 pp., ISBN: 9781483163734.

[12] P. Morse and H. Feshbach, *Methods of Theoretical Physics*, 11. McGraw-Hill Book Comp., Inc., New York, Toronto, London, 1953, vol. 1, p. 52.

[13] B. Auld, *Acoustic Fields and Waves in Solids Vol. II*. John Wiley & Sons, New York, Lonson, Syndey, Toronto, 1973.

[14] L. W. Schmerr, *Fundamentals of Ultrasonic Nondestructive Evaluation*. Springer-Verlag GmbH, Apr. 2016, 758 pp., ISBN: 9783319304632.

[15] J. L. Rose, "A Baseline and Vision of Ultrasonic Guided Wave Inspection Potential," *Journal of Pressure Vessel Technology*, vol. 124, no. 3, pp. 273–282, Aug. 2002.

[16] R. D. Mindlin, "Waves and vibrations in isotropic elastic plates," *Structural mechanics, ed. by. J. N. Goodier and N. J. Hoff, New York, Pergamon Press*, 1960.

[17] M. Lowe, "Matrix techniques for modeling ultrasonic waves in multilayered media," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 42, no. 4, pp. 525–542, Jul. 1995.

[18] A. Huber, *Dispersion calculator, version 1.11*, German Aerospace Center (DLR), Apr. 2021.

[19] R. Seifried, L. J. Jacobs, and J. Qu, "Propagation of guided waves in adhesive bonded components," *NDT & E International*, vol. 35, no. 5, pp. 317–328, Jul. 2002.

[20] M. Smith, *ABAQUS/Standard User's Manual, Version 6.14*. United States: Dassault Systéms Simulia Corp, 2014.

[21] B. Klein, *FEM - Grundlagen und Anwendungender Finite-Element-Methode im Maschinen- und Fahrzeugbau*, 10th ed. Springer Fachmedien Wiesbaden, 2015.

[22] K.-J. Bathe, *Finite element procedures*. United States Watertown, MA: K.J.Bathe, 2014, ISBN: 9780979004957.

[23] M. Niethammer, L. J. Jacobs, J. Qu, and J. Jarzynski, "Time-frequency representations of Lamb waves," *The Journal of the Acoustical Society of America*, vol. 109, no. 5, pp. 1841–1847, May 2001.

[24] J. F. James, *A Student's Guide to Fourier Transforms: With Applications in Physics and Engineering*, 3rd ed. Cambridge: Cambridge University Press, 2011, ISBN: 978-0-511-76230-7.

[25] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes*. Cambridge University Pr., 2007, 1248 pp., ISBN: 0521880688.

[26] L. Cohen, *Time-frequency analysis*, ser. Prentice Hall signal processing series. Englewood Cliffs, N.J: Prentice Hall PTR, 1995, ISBN: 978-0-13-594532-2.

[27] A. V. Oppenheim, *Discrete-Time Signal Processing*, ser. Prentice-Hall signal processing series. Englewood Cliffs, NJ: Prentice Hall, 1989, Book Title: Discrete-time signal processing /, ISBN: 978-0-13-216292-0.

[28] D. Alleyne and P. Cawley, "A two-dimensional fourier transform method for the measurement of propagating multimode signals," *The Journal of the Acoustical Society of America*, vol. 89, no. 3, pp. 1159–1168, Mar. 1991.

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[30] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Jan. 2017, arXiv: 1412.6980.

[31] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[32] 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.

[33] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, Massachusetts: The MIT Press, 2016, ISBN: 9780262035613.

[34] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer London, New York, Berlin, Heidelberg, 2011, ISBN: 978-1-84882-934-3.

[35] K. A. Terrani, "Accident tolerant fuel cladding development: Promise, status, and challenges," *Journal of Nuclear Materials*, vol. 501, pp. 13–30, Apr. 2018.

[36] C. Tang, M. Stueber, H. J. Seifert, and M. Steinbrueck, "Protective coatings on zirconium-based alloys as accident-tolerant fuel (ATF) claddings," *Corrosion Reviews*, vol. 35, no. 3, pp. 141–165, Aug. 2017, Publisher: De Gruyter.

[37] D. V. Nguyen *et al.*, "Mechanical behavior of a chromium coating on a zirconium alloy substrate at room temperature," *Journal of Nuclear Materials*, vol. 558, p. 153 332, Jan. 2022.

[38] F. Moser, L. J. Jacobs, and J. Qu, "Modeling elastic wave propagation in waveguides with the finite element method," *NDT & E International*, vol. 32, no. 4, pp. 225–234, Jun. 1999.

[39] PACE, *Partnership for an Advanced Computing Environment (PACE)*, Georgia Institute of Technologyy, 2017.

[40] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020.

[41] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *arXiv:1912.01703 [cs, stat]*, Dec. 2019, arXiv: 1912.01703.

[42] P. Virtanen *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, Mar. 2020.

[43] G. Chavent, *Nonlinear Least Squares for Inverse Problems: Theoretical Foundations and Step-by-Step Guide for Applications*, ser. Scientific Computation. Dordrecht: Springer Netherlands, 2010, ISBN: 9789048127856.

[44] K. Taunk, S. De, S. Verma, and A. Swetapadma, "A Brief Review of Nearest Neighbor Algorithm for Learning and Classification," in *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, May 2019, pp. 1255–1260.

[45] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[46] A. Patle and D. S. Chouhan, "SVM kernel functions for classification," in *2013 International Conference on Advances in Technology and Engineering (ICATE)*, Jan. 2013, pp. 1–9.

[47] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, ser. COLT '92, New York, NY, USA: Association for Computing Machinery, Jul. 1992, pp. 144–152, ISBN: 978-0-89791-497-0.

[48] C. E. Rasmussen, "Gaussian Processes in Machine Learning," in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, ser. Lecture Notes in Computer Science, O. Bousquet, U. von Luxburg, and G. Rätsch, Eds., Berlin, Heidelberg: Springer, 2004, pp. 63–71, ISBN: 978-3-540-28650-9.

[49] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012, ISBN: 978-0-262-01802-9.

[50] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-Validation," in *Encyclopedia of Database Systems*, L. LIU and M. T. ÖZSU, Eds., Boston, MA: Springer US, 2009, pp. 532–538, ISBN: 978-0-387-39940-9.

[51] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.

[52] D. Salomon, *Data Compression: The Complete Reference*, 4th ed. London: Springer, 2007, ISBN: 978-1-84628-602-5.

[53] D. Zhu, H. Yao, B. Jiang, and P. Yu, "Negative Log Likelihood Ratio Loss for Deep Neural Network Classification," *arXiv:1804.10690 [cs, stat]*, Apr. 2018, arXiv: 1804.10690.

[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015, arXiv: 1512.03385.

[55] S. Zhang, X. Li, and H. Jeong, "Measurement of rayleigh wave beams using angle beam wedge transducers as the transmitter and receiver with consideration of beam spreading," *Sensors*, vol. 17, no. 6, p. 1449, Jun. 2017.