SCHOOL OF ENGINEERING AND DESIGN
TECHNISCHE UNIVERSITÄT MÜNCHEN

BACHELOR'S THESIS

MATERIAL BEHAVIOR PREDICTION USING PHYSICS-INFORMED NEURAL NETWORKS AND FOURIER NEURAL OPERATORS

Author: Radu-Matei Vasilache
Matriculation Number: 03761745
Supervisor: Prof. Dr. Faidon-Stelios Koutsourelakis, Dr. Yaohua Zang
Submission Date: 28.11.2024

I hereby declare that this thesis is my own work and that I have documented all sources and material used. I confirm that this work has not been submitted elsewhere in any form for the fulfillment of any other degree or qualification.

Munich, 28.11.2024 Radu-Matei Vasilache

Acknowledgments

The support and guidance of Prof. Dr. Faidon-Stelios Koutsourelakis and Dr. Yaohua Zang is greatly appreciated. Mr. Tim Duka provided valuable code and insights that significantly advanced this thesis.

Abstract

Accurately and efficiently solving the partial differential equation (PDE) of the linear elasticity problem is a critical topic in the field of solid mechanics, especially for heterogeneous materials. This thesis explores the potential of deep learning methodologies, specifically Physics-Informed Neural Networks (PINNs) and Fourier Neural Operators (FNOs) to predict the stress and the displacement fields of two-phase materials under external loads. Traditional numerical methods, such as the Finite Element Method (FEM), offer high accuracy but are computationally expensive and less efficient for repeated evaluations required in inverse problems. PINNs and FNOs offer potential solutions to these challenges by leveraging deep learning architectures for predictive modeling.

The research investigates the influence of contrast ratio, image resolution, and problem setup on model accuracy. After choosing the architectures and the optimal hyperparameters, multiple models were trained and compared. The investigations revealed that increasing contrast ratios consistently reduced accuracy, with PINNs being unsuitable for heterogeneous materials and FNOs demonstrating better scalability but limited performance for high material heterogeneity. Higher image resolutions generally decreased accuracy for both methods, however they don't make huge impact on the performance of FNOs. Problem setup analyses across three scenarios revealed substantial variations in accuracy, highlighting the significant impact of external mechanical factors. Finally, investigations into FNO's output channel configurations confirmed that predicting multiple solution components simultaneously does not compromise its accuracy, showcasing the model's flexibility and robustness.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The significance of materials modeling is widely acknowledged, as advancements in engineering fields heavily depend on materials research. Predicting mechanical behavior is a fundamental aspect of this process, serving as the foundation for determining other critical properties. From stress and displacement fields, key insights such as effective material properties, localized damage risks, and stress-dependent features can be derived with ease. Furthermore, a reliable method for predicting properties from a given microstructure enables faster and more accurate solutions to inverse problems, enhancing efficiency and precision in materials analysis.

Approaches to solving the linear elasticity problem typically fall into two categories: analytical or numerical. While analytical methods can provide exact solutions for specific scenarios, such as one-dimensional linear elasticity or two-dimensional Bernoulli plates [1, 2], they struggle to address more general cases. Complexities introduced by irregular geometries or material heterogeneity pose significant challenges for analytical techniques. In contrast, numerical methods are better suited for these more complex problems. Comprehensive studies have explored their application to solving linear elasticity equations [3, 4, 5], with the Finite Element Method (FEM) emerging as the most prominent numerical solver. Renowned for its accuracy, FEM excels in providing precise predictions but comes with high computational costs. These limitations are particularly pronounced in inverse problems, where repeated property evaluations are required, as highlighted in [6].

In recent years, deep learning has emerged as one of the most promising subfields of machine learning. Beyond its publicized successes in areas such as natural language processing (NLP) and Computer Vision, deep learning has gained significant attention from researchers for its potential to address challenges in various scientific and engineering domains. Achieving high-accuracy solutions for complex partial differential equations (PDEs) while maintaining low computational costs remains an ongoing challenge, driving the development of innovative approaches. High-dimensional PDEs, commonly encountered in fields such as physics, finance, and engineering, are especially affected by the ”curse of dimensionality,” where computational demands grow exponentially with each additional dimension. Deep learning methods have shown potential in overcoming these challenges. Papers [7] and [8] leverage backward stochastic differential equations (BSDEs) to reformulate the PDEs as optimization problems, where neural networks approximate the gradient of the solution at each time step. In [9], the Deep Galerkin Method (DGM) solves high-dimensional PDEs by using a neural network trained on randomly sampled spatial points and therefore obtaining mesh-free solutions across complex domains. Paper [10] solves high-dimensional PDEs [11] by reformulating them as variational problems, training by minimizing the associated energy functional.

The research in this thesis is focused on the linear elasticity problem for homogeneous and

heterogeneous materials. Two distinct deep learning approaches were investigated: Physics-informed Neural Networks (PINNs) and Fourier Neural Operators (FNOs). The PINN method has demonstrated superiority over numerical methods, particularly for high-dimensional PDEs, due to its mesh-free nature. Unlike traditional deep learning approaches, PINNs do not rely on large datasets; instead, they approximate the underlying PDEs and enforce boundary conditions directly. Since their introduction by [34], PINNs have been implemented and researched for different problems [12, 13, 14] and with different configurations [15, 16, 17]. The papers [18, 19] provide a comprehensive overview of the current state of research in this field, making them an excellent resource for readers new to the PINN methodology. The physics-informed neural networks have been also successfully applied in the field of solid mechanics [20, 21, 22]. Especially interesting for this thesis are the papers [23, 24], that investigate the performance of the PINN method on solving the linear elasticity problem for heterogeneous materials.

Introduced by [35], the Fourier Neural Operator (FNO) is an innovative deep learning architecture designed to map between infinite-dimensional function spaces. Unlike Physics-Informed Neural Networks (PINNs), FNO is entirely data-driven and, once trained, can instantly predict the solution of a PDE for a given set of parameters. If demonstrated to achieve high accuracy, this method could be particularly valuable for materials modeling, where rapid solutions to forward problems are essential. Current research on FNOs has focused on adapting and applying this method on high dimensional problems in specific domains such as turbulence modeling [25], weather prediction [26, 27] or multiphase flows [28]. Most research on FNOs focuses on analyzing their performance on PDEs with varying initial or boundary conditions, with comparatively less attention given to their application to problems involving heterogeneous materials. However, paper [29] introduces an FNO variant, called IFNO, that efficiently handles sharp contrasts and nonlinearities in heterogeneous materials, enabling accurate predictions with reduced memory costs and improved computational efficiency.

In this thesis, the performances of the Physics-informed Neural Network and the Fourier Neural Operator on heterogeneous materials are investigated. The reason is to conclude whether the two methods can be successfully applied also for two-phased materials. The following chapter offers the reader a thorough review of the theoretical aspects of the concepts discussed in the thesis. The "Methodology" chapter focuses on the technical details of the implementation, such that the reader has a better understanding of the way the research was conducted. Here, the methods of generating the microstructures and obtaining the ground truth solutions are explained. Moreover, several comparisons of the most important model hyperparameters are conducted. The last chapter, "Results and Discussion", contains the investigations on the performances of the two methods on heterogeneous materials. The two key features specific to the materials investigated in the thesis are the contrast ratio between the two phases and the number of pixels of the microstructure. The influence of these two properties, together with the problem setup, namely the load/ source term and the boundary conditions, on the relative error of the Physics-informed Neural Network and the Fourier Neural Operator are analyzed.

Chapter 2

Theoretical Aspects

2.1 Microstructures

The research presented in this thesis revolves around two-phase linear elastic materials, in which each phase is assumed to be linear-elastic and isotropic. For both phases, the same typical Poisson's ratio of $\nu_1 = \nu_2 = 0.3$ is assumed, while different Young's moduli are imposed, correlated through the contrast ratio ($CR = E_2/E_1$). It is assumed that $E_1 = 1$.

The materials are modeled as two-dimensional structures composed of square cells of uniform dimensions, each of which is homogeneous. The resolution of the microstructures can be varied as needed . A spatial coordinate domain of $(0, 1) \times (0, 1)$ is associated with the material. Therefore, when points are generated on the material, each point is assigned an x and y coordinate between 0 and 1, a Poisson's ratio of $\nu = 0.3$, and a specific Young's modulus E, which depends on the phase to which it belongs.

2.2 The equations of linear elasticity

The governing equations of linear elasticity are fundamental to the first method presented in this thesis and are therefore discussed in detail in this section. For an isotropic material, under the regime of small deformations in three-dimensional space, these equations are expressed as follows:

- Equilibrium - Conservation of Linear Momentum (3 equations):

$$\sigma_{ij,j} + f_i = 0 \tag{2.1}$$

  where σ_{ij} represents the components of the stress tensor, f_i denotes the components of the body force per unit volume (source term), and the subscript after the comma indicates partial differentiation.

- Strain-Displacement Relations (6 equations):

$$\varepsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}) \tag{2.2}$$

  where ϵ_{ij} is the strain tensor and u_i are the displacement components.

- Constitutive Law (6 equations):

$$\sigma_{ij} = \lambda \varepsilon_{kk} \delta_{ij} + 2\mu \varepsilon_{ij} \tag{2.3}$$

where λ and μ are the Lamé parameters, ϵ_{kk} is the trace of the strain tensor, and δ_{ij} is the Kronecker delta.

Together, these equations form a system of partial differential equations (PDEs) governing the behavior of the material under the assumptions of linear elasticity. The equilibrium equations (2.1) describe the balance of forces, providing three PDEs that relate the stress components to the body forces. The strain-displacement relations (2.2) express the strain tensor in terms of the displacement field, introducing six additional equations. The constitutive laws (2.3) link the stress tensor to the strain tensor through material properties. The coupled system has 15 equations with 15 unknowns, namely, the displacement (u_i), the stress (σ_{ij}), and the strain (ϵ_{ij}) components. Solving this system requires appropriate boundary conditions.

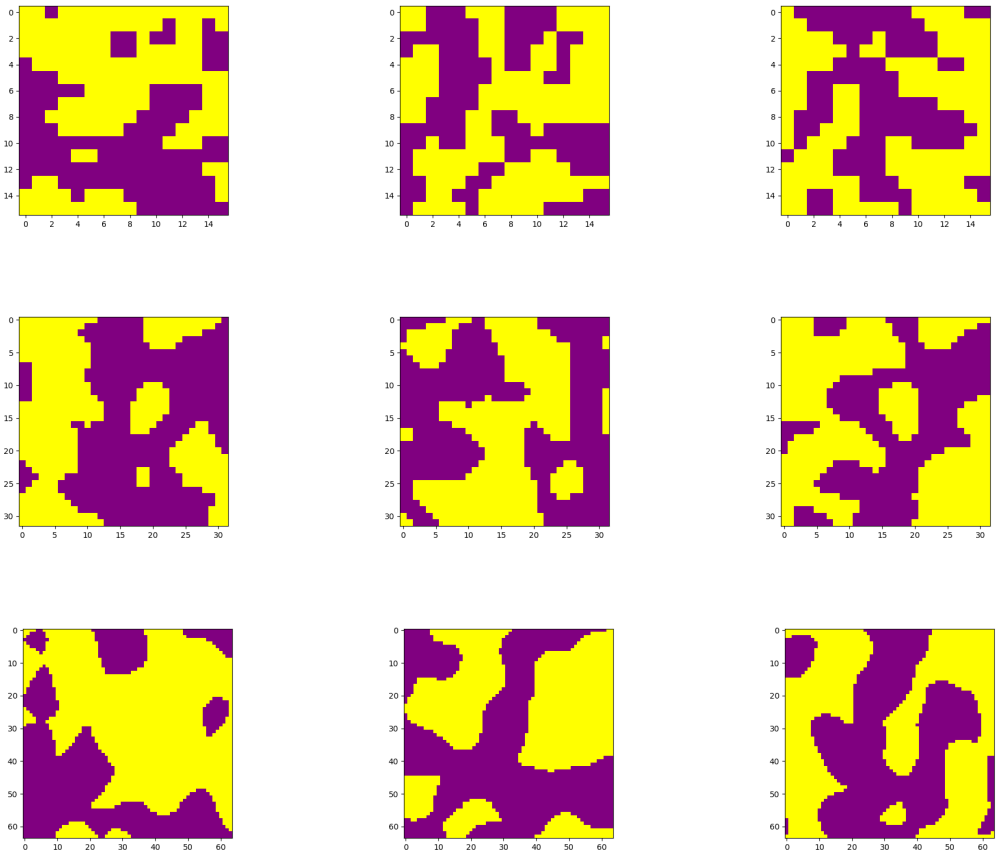

Figure 2.1: Three exemplary two-phase microstructures. **Top**: 16×16, **Middle**: 32×32, **Bottom**: 64×64.

2.2.1 Weak Form

The weak form of a partial differential equation (PDE) reformulates the problem to emphasize integral relationships among the variables using a test function [45]. This approach is particularly useful in the Finite Element Method (FEM), where it facilitates the approximation of solutions in function spaces.

To derive the weak form, the strong form of the governing equations, such as the equilibrium equation in linear elasticity, is first introduced:

$$\nabla \cdot \boldsymbol{\sigma} + \mathbf{f} = 0 \quad \text{in } \Omega,$$

where Ω is the domain, $\boldsymbol{\sigma}$ is the stress tensor, and $\mathbf{f}$ represents the body forces.

The weak form is obtained by multiplying the strong form by a test function $\mathbf{v} \in V$, where $V = [H^1(\Omega)]^2$ is the space of admissible test functions, and integrating over the domain Ω. The solution will belong to $U = [L^2(\Omega)]^{2\times 2}$. Applying integration by parts to transfer derivatives from the stress tensor to the test function, the weak form is expressed as:

$$\int_\Omega \boldsymbol{\sigma} : \nabla \mathbf{v} \, d\Omega = \int_{\partial\Omega_N} (\boldsymbol{\sigma} \cdot \mathbf{n}) \cdot \mathbf{v} \, dS + \int_\Omega \mathbf{f} \cdot \mathbf{v} \, d\Omega \tag{2.4}$$

where σ is the stress tensor, $\mathbf{n}$ represents the normal vector on the boundary, $\mathbf{f}$ denotes the volumetric body force, and $\mathbf{v}$ is the test function.

2.2.2 Assumptions and Observations Regarding the Problem

The equations of linear elasticity are typically applicable to real-world scenarios in three dimensions. The stress and strain tensors comprise nine components, of which six are independent, while the stiffness tensor contains 81 components, with 21 being independent. In Voigt notation, the constitutive law is expressed as follows:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{13} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ & & C_{33} & C_{34} & C_{35} & C_{36} \\ & & & C_{44} & C_{45} & C_{46} \\ & & & & C_{55} & C_{56} \\ sym. & & & & & C_{66} \end{bmatrix} \cdot \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \varepsilon_{12} \\ \varepsilon_{23} \\ \varepsilon_{13} \end{bmatrix} = \mathbf{C} \cdot \boldsymbol{\varepsilon} \tag{2.5}$$

For the research, the equations are simplified to the two-dimensional case. Therefore, the Equation 2.5 reduces to:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ & C_{22} & C_{23} \\ sym. & & C_{33} \end{bmatrix} \cdot \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{12} \end{bmatrix} = \mathbf{C} \cdot \boldsymbol{\varepsilon} \tag{2.6}$$

In order for this process to be consistent, the following assumptions must be made:

1. The source term (force inside the domain) and the traction boundary conditions must be two-dimensional (the z-component is equal to 0). Otherwise, the stress components involving the third dimension will be non-zero, requiring their consideration in the computations.

2. The strain component along the z-dimension is neglected. Typically, in the plane stress case (which is established by adhering to the first assumption), this strain component is non-zero. As it can be observed from the equations, σ_{11} and σ_{22} are influenced by it; thus, without this assumption, reducing the problem to a two-dimensional one would not be possible. To address this, it can be assumed that the material is anisotropic or that ϵ_{33} is sufficiently small to be neglected.

3. The boundary conditions for the displacement in the z-direction are unknown, and no specific conditions are imposed on ϵ_{33}. The strain-displacement relations do not pose any issues for the desired simplification, as the z-components of the strain tensor and displacement do not influence ϵ_{11}, ϵ_{22}, or ϵ_{12}. This assumption ensures that the problem remains well-posed and avoids over-definition.

2.3 Fourier Transform

The Fourier Transform is a fundamental concept underlying the second method presented in this thesis. Therefore, it is important to provide an overview of this concept in this chapter.

2.3.1 Fourier Series

The Fourier series is a mathematical tool that allows the decomposition of any periodic function into harmonics. It provides a way to represent complex, periodic signals using simpler trigonometric components. For any periodic function with period T, it can be expressed as an infinite sum of sine and cosine functions:

$$f(t) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos\left(\frac{2\pi nt}{T}\right) + b_n \sin\left(\frac{2\pi nt}{T}\right) \right)$$

where a_0, a_n, and b_n are the Fourier coefficients given by:

$$a_0 = \frac{1}{T}\int_0^T f(t)\,dt, \quad a_n = \frac{2}{T}\int_0^T f(t)\cos\left(\frac{2\pi nt}{T}\right)\,dt, \quad b_n = \frac{2}{T}\int_0^T f(t)\sin\left(\frac{2\pi nt}{T}\right)\,dt$$

Fourier series provide a powerful method for analyzing periodic functions by decomposing complex signals into simpler sine and cosine components. This decomposition aids in understanding and manipulating signals in various applications, including signal processing and electrical engineering. Additionally, Fourier series simplify the solution of differential equations by transforming complex time-domain problems into more manageable frequency-domain representations [30].

2.3.2 Fourier Transform

The Fourier Transform generalizes the concept of the Fourier series to accommodate non-periodic functions, allowing for the transformation of a function from the spatial domain into the frequency domain. This transformation provides a powerful method for analyzing signals by revealing the frequency components that constitute the original function. For continuous functions defined in the spatial domain, the corresponding representation in the frequency domain is also continuous, maintaining the integrity of the information while offering insights into the underlying frequency characteristics. This ability to transition between domains is essential in various fields, including signal processing, communications, and engineering, where understanding the frequency content of signals is crucial for analysis and design.

The Fourier Transform (FT) of a continuous function $f(t)$ is given by:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}\,dt$$

Similarly, the process of converting a function from the frequency domain back to the time domain is referred to as the Inverse Fourier Transform (IFT).

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} \, d\omega$$

The Fourier Transform is a critical tool used across various fields for converting signals from the time domain to the frequency domain. In signal processing, it facilitates filtering and data compression techniques, such as MP3 and JPEG, by reducing file sizes while preserving quality. In telecommunications, it enables efficient modulation and demodulation of signals for reliable transmission. The Fourier Transform also plays a key role in image processing, allowing for enhancements and edge detection by analyzing frequency components. Additionally, it assists in solving differential equations in physics, making it invaluable for studying waves and vibrations, and is essential in medical imaging techniques like MRI for image reconstruction[31].

Theorem 2.3.1 (Convolution Theorem)**.** *Let $f(t)$ and $g(t)$ be two integrable functions. The convolution of these functions, denoted by $(f * g)(t)$, is defined as:*

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) \, d\tau$$

The Fourier Transform of the convolution is given by:

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\}$$

In mathematical terms, if $F(\omega)$ is the Fourier Transform of $f(t)$ and $G(\omega)$ is the Fourier Transform of $g(t)$, then:

$$\mathcal{F}\{f * g\} = F(\omega) G(\omega)$$

Similarly, the inverse of the Convolution Theorem states that:

$$\mathcal{F}^{-1}\{F(\omega) G(\omega)\} = f(t) * g(t)$$

Theorem 2.3.1 is a fundamental property of the Fourier Transform, as it significantly accelerates the computation of the convolution of two functions by minimizing the resources required when performed in the frequency domain.

2.3.3 Discrete Fourier Transform (DFT)

The Discrete Fourier Transform (DFT) is a powerful mathematical tool used to analyze discrete signals and convert them from the time domain into the frequency domain. It transforms a finite sequence of equally spaced samples of a function into a same-length sequence of coefficients representing the function's frequency components. The DFT is particularly valuable in digital signal processing, where it enables efficient computation and analysis of signals in applications such as audio processing, image compression, and telecommunications.

The mathematical formulation of the DFT for a sequence of N samples $x[n]$ is given by:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-i \frac{2\pi}{N} kn}$$

where $X[k]$ represents the frequency component at index k, and n ranges from 0 to $N - 1$.

The inverse DFT, which reconstructs the original time-domain sequence from its frequency components, is defined as:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{i \frac{2\pi}{N} kn}$$

2.3.4 Fast Fourier Transform (FFT)

The Fast Fourier Transform (FFT) is an efficient algorithm designed to compute the Discrete Fourier Transform (DFT) and its inverse, significantly reducing the computational complexity from $O(N^2)$ to $O(N \log N)$. This enhancement makes the FFT particularly valuable in digital signal processing, allowing for real-time analysis of signals and systems. The FFT leverages symmetries in the DFT to minimize the number of calculations needed, making it possible to process large datasets quickly. Its applications span various fields, including audio and image processing, telecommunications, and data compression. While there are several algorithms for performing the FFT, the Cooley-Tukey algorithm is the most widely used, based on recursively dividing the DFT into smaller DFTs.

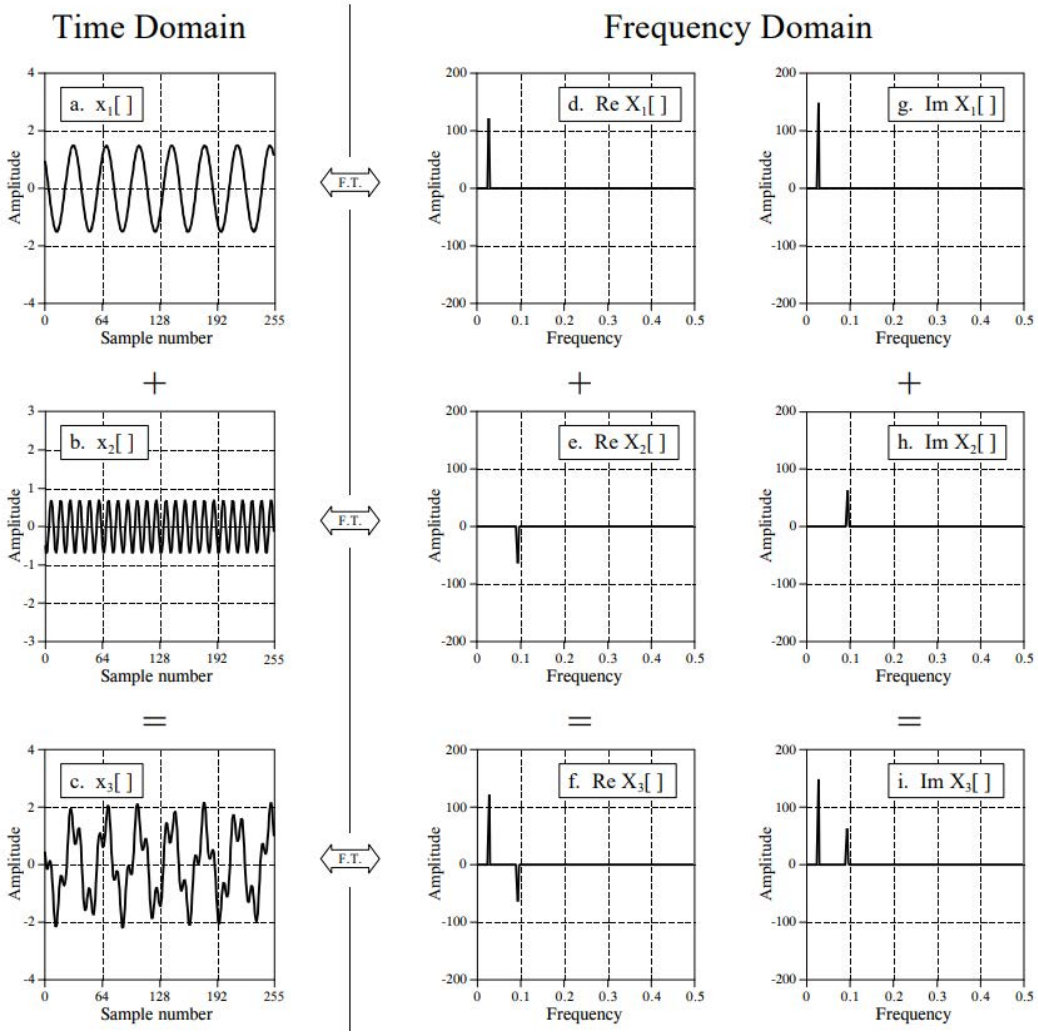

Figure 2.2: Equivalence of the Time and Frequency Domains

2.4 Neural Networks

Neural networks (NNs), also known as artificial neural networks (ANNs), form the foundation of deep learning, which is currently regarded as the most promising area within machine learning. These networks have garnered considerable attention for their exceptional ability to provide predictions with unprecedented accuracy. In traditional machine learning, much of the success relies on feature engineering, where domain experts create features to enhance model performance. In contrast, ANNs largely eliminate the need for this process, enabling deep learning models to autonomously extract relevant features from the dataset.

Inspired by the neural networks found in nature, the fundamental principle of deep learning methods lies in their capacity to automatically adjust parameters for improved predictions. Initially, all parameters are typically assigned random values and are subsequently adapted during the training process to best fit the data using a specific optimization algorithm. For an in-depth exploration of neural networks, their functionality, and the techniques employed for parameter adjustment, readers are encouraged to consult more specialized resources on the topic [32, 33].

2.4.1 Fully Connected Neural Networks

Fully connected neural networks (FCNNs), also referred to as multilayer perceptrons (MLPs), are among the most common and foundational neural network architectures. They are typically the first type of neural network presented in educational contexts, as they provide a basis for understanding more complex architectures. Additionally, fully connected layers often appear in the structure of various other network designs, highlighting their significance in the field of deep learning.

FCNNs are composed of an input layer, one or more hidden layers, and an output layer. As illustrated in Figure 2.3, every neuron in each layer is connected to all neurons in the following layer. Mathematically, the computations for each layer are expressed as follows:

$$\mathbf{x}^{(i+1)} = \phi\left(\mathbf{W}^{(i)}\mathbf{x}^{(i)} + \mathbf{b}^{(i)}\right) \tag{2.7}$$

where $\mathbf{x}^{(i)}$ is the input to the layer, $\mathbf{W}^{(i)}$ are the weights of the layer, $\mathbf{b}^{(i)}$ are the biases of the layer, and ϕ is the non-linear activation function.

A fully connected neural network with h hidden layers can thus be represented as:

$$\mathbf{y}_{NN} = f\left(\mathbf{x}^{(0)}\right) = \phi\left(\mathbf{W}^{(h)}\phi\left(\mathbf{W}^{(h-1)}\cdots\phi\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(h-1)}\right)\cdots + \mathbf{b}^{(h)}\right) \tag{2.8}$$

Non-linear activation functions are essential between matrix multiplications to capture the non-linear behavior of the model. Without these activation functions, all matrix multiplications would effectively combine into a single transformation, reducing the network to just one layer:

$$\mathbf{y}_{NN} = \mathbf{W}^{*}\mathbf{x}^{(0)} + \mathbf{b}^{*}$$

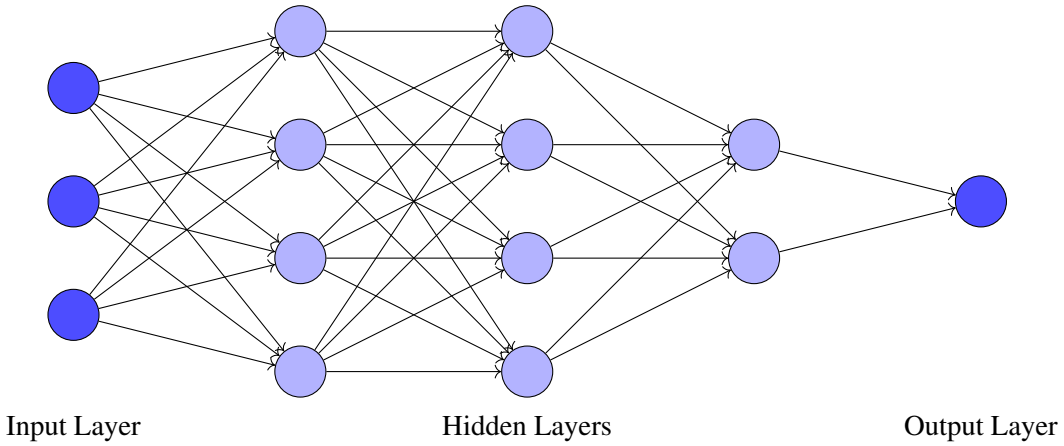

Figure 2.3: Schematic of a Fully Connected Neural Network (FCNN)

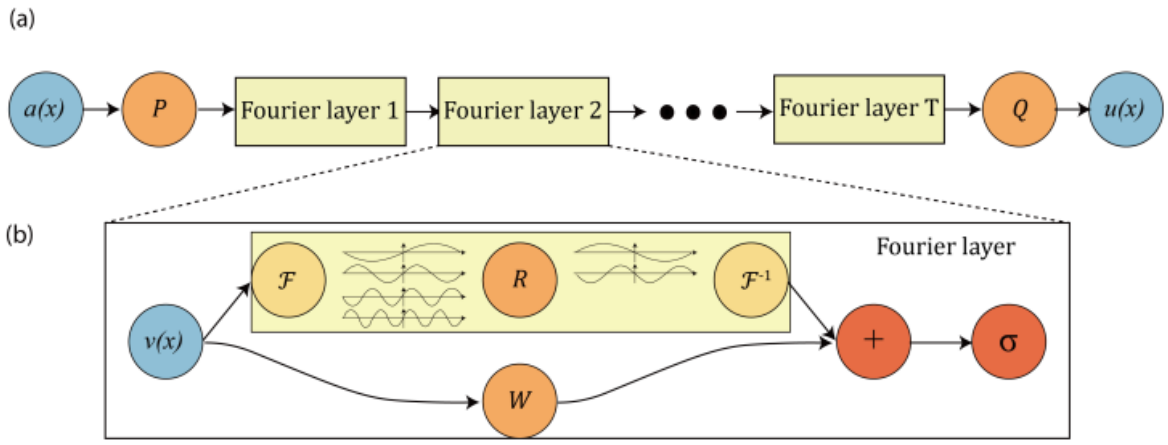

Figure 2.4: **(a)** The architecture of the neural operator; **(b)** Fourier Layer.

2.4.2 Physics-informed Neural Networks

Introduced by [34], physics-informed Neural Networks (PINNs) represent a novel and powerful approach that integrates physical laws into the training process of neural networks. Unlike traditional NNs, which primarily rely on data to learn mappings between inputs and outputs, PINNs leverage established physical principles governing the system, such as conservation laws, initial and boundary conditions, and differential equations. This integration not only enables the model to make data-driven predictions but also ensures that these predictions align with the underlying physical phenomena, resulting in more accurate and generalizable solutions.

In a conventional neural network, the primary objective of the loss function is to minimize the discrepancy between the model's predictions and the actual observed data. In contrast, Physics-Informed Neural Networks (PINNs) enhance this loss function by incorporating additional terms that represent the physical constraints of the system being modeled. These supplementary terms guide the model to not only fit the available data but also conform to the underlying physics of the problem at hand. By embedding these physical laws directly into the training process, PINNs can often operate effectively with less data compared to traditional neural networks, while simultaneously producing solutions that are more consistent with physical principles. Consequently, this approach leads to enhanced model robustness and reliability, making PINNs particularly valuable in applications where data may be scarce or difficult to obtain.

According to [34], the loss for PINNs is defined as

$$\mathcal{L} = \lambda_d \mathcal{L}_d + \lambda_p \mathcal{L}_p \tag{2.9}$$

where

$$\mathcal{L}_d = \frac{1}{N_d} \sum_{i=1}^{N_d} \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|_2^2 \tag{2.10}$$

is the data loss, which measures the difference between the predicted output $\hat{\mathbf{y}}_i$ and the actual output $\mathbf{y}_i$ for N_d data points. And:

$$\mathcal{L}_p = \frac{1}{N_p} \sum_{j=1}^{N_p} |\mathcal{F}(\mathbf{u}_j)|^2 \tag{2.11}$$

is the physics loss, where $\mathcal{F}$ represents the residual of the governing equations evaluated at N_p collocation points. The parameters λ_d and λ_p are weighting factors that balance the contributions of the data loss and physics loss in the overall loss function $\mathcal{L}$.

PINNs typically employ fully connected (dense) layers, allowing every neuron in one layer to connect to every neuron in the subsequent layer (Figure 2.3). This architecture is effective for approximating complex functions, making it suitable for capturing the underlying behavior of the physical system being modeled. The input to a PINN often consists of both the independent variables of the system (e.g., spatial coordinates and time) and any additional parameters relevant to the physical problem. This allows the network to learn from the full context of the problem, including boundary and initial conditions. Both Figures 2.5 and 2.6 from paper [34] illustrate the prediction and the analytical solution of PINN models for two different problems.

In this thesis, the partial differential equation (PDE) under investigation considers only the spatial coordinates as input features. Additionally, the initial model presented—specifically the one that implements a Physics-Informed Neural Network (PINN)—is trained solely based on physical accuracy, without utilizing any supplementary data.

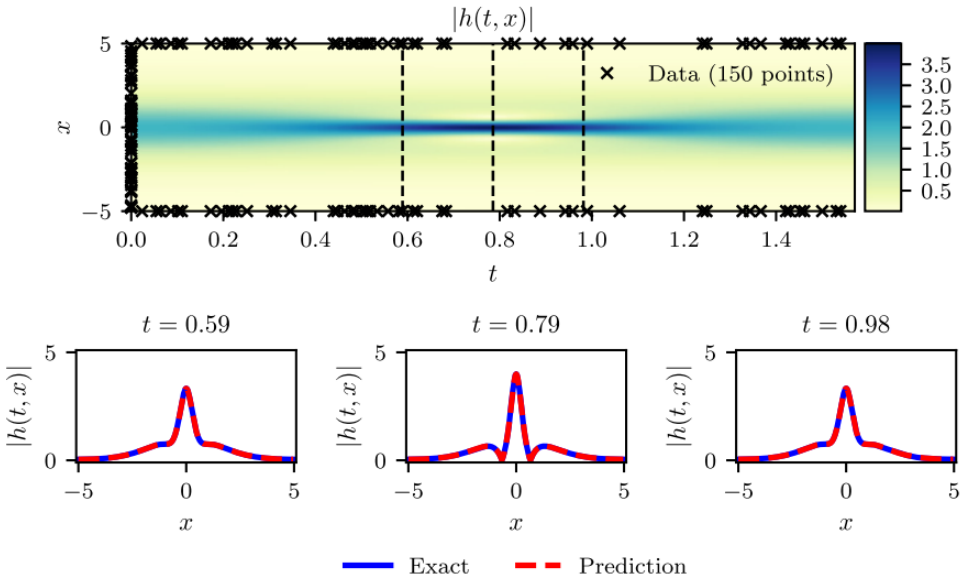

Figure 2.5: *Schrodinger equation*: **Top:** Predicted solution $|h(t,x)|$ along with the initial and boundary training data. In addition, we are using 20,000 collocation points generated using a Latin Hypercube Sampling strategy. **Bottom:** Comparison of the predicted and exact solutions corresponding to the three temporal snapshots depicted by the dashed vertical lines in the top panel. The relative L^2 error for this case is $1.97 \cdot 10^{-3}$.

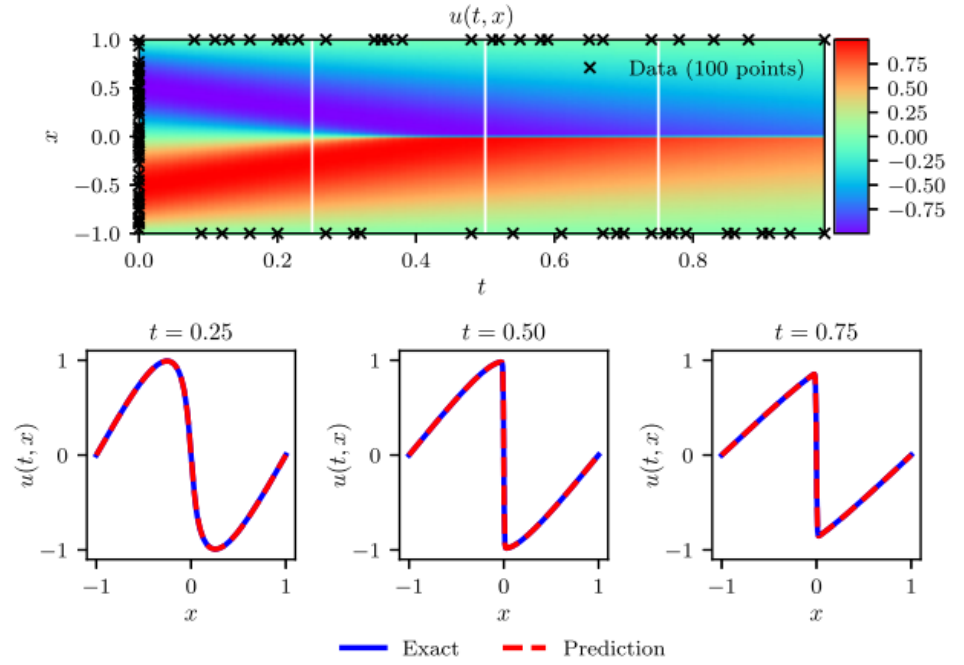

Figure 2.6: *Burgers' equation*: **Top:** Predicted solution $u(t,x)$ along with the initial and boundary training data. In addition, we are using 10,000 collocation points generated using a Latin Hypercube Sampling strategy. **Bottom:** Comparison of the predicted and exact solutions corresponding to the three temporal snapshots depicted by the white vertical lines in the top panel. The relative L^2 error for this case is $6.7 \cdot 10^{-4}$.

2.4.3 Fourier Neural Operator

The Fourier Neural Operator (FNO) is a novel deep learning architecture, able to learn mappings between infinite-dimensional spaces of functions, in which the integral operator is restricted to a convolution, and instantiated through a linear transformation in the Fourier domain [35].

Let $D \subset \mathbb{R}^d$ be a bounded, open set, and let $A = A(D; \mathbb{R}^{d_a})$ and $U = U(D; \mathbb{R}^{d_u})$ be separable Banach spaces of functions taking values in $\mathbb{R}^{d_a}$ and $\mathbb{R}^{d_u}$, respectively. Furthermore, let $G^\dagger : A \to U$ be a typically non-linear map.

Suppose the observations $\{a_j, u_j\}_{j=1}^N$ are given, where $a_j \sim \mu$ is an i.i.d. sequence from the probability measure μ supported on A, and $u_j = G^\dagger(a_j)$ is possibly corrupted with noise. FNO aims to construct an approximation of $G^\dagger$ by defining a parametric map $G : A \times \Theta \to U$ or equivalently $G_\theta : A \to U, \theta \in \Theta$ for some finite-dimensional parameter space Θ by selecting $\theta^\dagger \in \Theta$ such that $G(\cdot, \theta^\dagger) = G_\theta^\dagger \approx G^\dagger$.

This leads to the following optimization problem, where the objective is to minimize the cost function $C : U \times U \to \mathbb{R}$:

$$\min_{\theta \in \Theta} E_{a \sim \mu}[C(G(a, \theta), G^\dagger(a))]$$

Figure 2.4(a) illustrates the general structure of a neural operator, as described in [35, 36] The neural operator is structured as an iterative architecture represented by $v_0 \to v_1 \to \ldots \to v_T$, where v_j for $j = 0, 1, \ldots, T-1$ is a sequence of functions taking values in $\mathbb{R}^{d_v}$. The input $a \in A$ is initially transformed into a higher-dimensional representation given by $v_0(x) = P(a(x))$ using a local transformation P, which is typically parameterized by a shallow fully connected neural network. Subsequent updates are then applied iteratively as $v_t \to v_{t+1}$ (as defined below). The output $u(x) = Q(v_T(x))$ represents the projection of v_T through the local transformation $Q : \mathbb{R}^{d_v} \to \mathbb{R}^{d_u}$. In each iteration, the update $v_t \to v_{t+1}$ is defined as the composition of a non-local integral operator K and a local, nonlinear activation function σ:

$$v_{t+1}(x) := \sigma\left(W v_t(x) + K(a; \varphi) v_t(x)\right), \quad \forall x \in D \tag{2.12}$$

where $K : A \times \Theta_K \to L(U(D; \mathbb{R}^{d_v}), U(D; \mathbb{R}^{d_v}))$ maps to bounded linear operators on $U(D; \mathbb{R}^{d_v})$ and is parameterized by $\varphi \in \Theta_K$. The transformation $W : \mathbb{R}^{d_v} \to \mathbb{R}^{d_v}$ is a linear transformation, and $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-linear activation function.

Definition 1 (Kernel Integral Operator K) Definition of the general kernel integral operator in 2.12.

$$\left(K(a; \varphi) v_t\right)(x) := \int_D \kappa\left(x, y, a(x), a(y); \varphi\right) v_t(y)\, dy, \quad \forall x \in D \tag{2.13}$$

where $\kappa_\varphi : \mathbb{R}^{2(d+d_a)} \to \mathbb{R}^{d_v \times d_v}$ is a neural network parameterized by $\varphi \in \Theta_K$.

Definition 3 (Fourier Integral Operator K) Define the Fourier integral operator

$$\left(K(\varphi) v_t\right)(x) = \mathcal{F}^{-1}\left(R_\varphi \cdot (\mathcal{F} v_t)\right)(x), \quad \forall x \in D \tag{2.14}$$

where R_φ is the Fourier transform of a periodic function $\kappa : D \to \mathbb{R}^{d_v \times d_v}$ parameterized by $\varphi \in \Theta_K$. Figure 2.4(b) illustrates the inner structure of one Fourier layer.

A key advantage of the Fourier Neural Operators is the discretization invariance, as they are capable of learning from and evaluating functions that are discretized in an arbitrary manner. Since the parameters are learned directly in Fourier space, resolving the functions in physical space is achieved by projecting onto the basis $e^{2\pi i h x_k}$, which is well-defined across $\mathbb{R}^d$. This characteristic enables zero-shot super-resolution. Additionally, the architecture maintains a consistent error at any resolution of the inputs and outputs. In contrast, the standard CNN methods exhibit an error that increases with resolution [35]. Figure 2.7 illustrates a comparison of FNO with other NNs for 3 different equations [35].

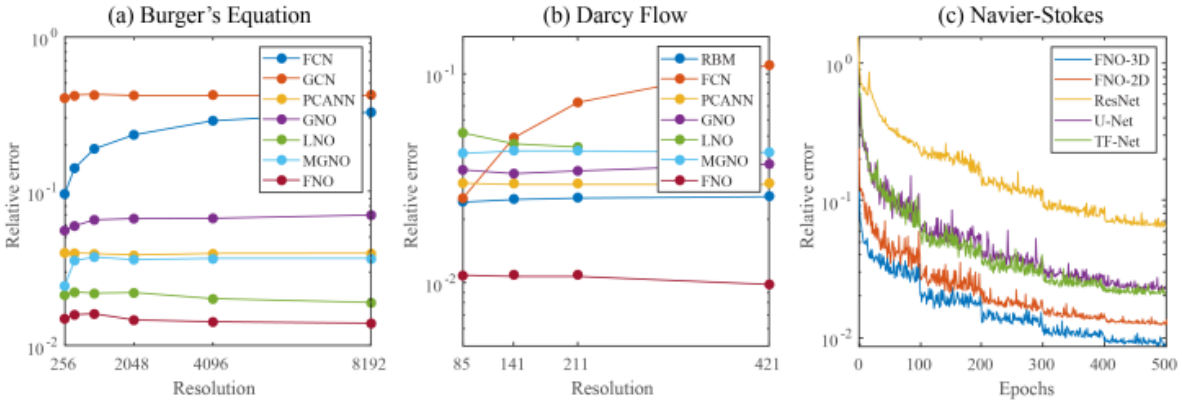

Figure 2.7: FNO: Comparison for different PDEs. **Left:** Benchmarks on Burgers' equation; **Mid:** Benchmarks on Darcy Flow for different resolutions; **Right:** The learning curves on Navier-Stokes $\nu = 1 \times 10^{-3}$ with different benchmarks. Train and test on the same resolution.

Chapter 3

Methodology

This thesis investigates two neural network methodologies for solving the linear elasticity problem in two-dimensional, two-phased materials: Physics-Informed Neural Networks (PINNs) and Fourier Neural Operators (FNOs). This chapter discusses, on one hand, the process of generating the materials, and on the other hand, the key aspects of the practical implementation of these models. Section 3.1 on sampling from distributions to generate microstructures is based on the bachelor thesis [37], which closely follows Tim Duka's internship report [38].

3.1 Generation of Microstructures

Each microstructure is represented by a unit square, discretized into uniformly distributed quadratic pixels. The resolution of the generated images (or image size) can be set to any natural number, but, for this thesis, the researched resolutions are 16×16, 32×32, 64×64. Each pixel in the microstructure was assigned a random decimal value, generated using a Multivariate Gaussian Generator (MVG) $z \sim \mathcal{N}(\mu, \Sigma)$. The density function of the MVG is defined as

$$f_z(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{\det(2\pi\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{3.1}$$

By choosing an appropriate Covariance Matrix $\boldsymbol{\Sigma}$, it is possible to obtain cohesive single-phase regions, that can represent a two-phase material realistically. The covariance $(\boldsymbol{\Sigma})_{ij} = \Sigma_{ij}$ between two pixels e_i and e_j will depend on the differences in their midpoints' x- and y-coordinates as:

$$\Sigma_{ij} = \sigma^2 e^{-d\left(\frac{1}{r_x^2}|x_i-x_j|^2+\frac{1}{r_y^2}|y_i-y_j|^2\right)} \tag{3.2}$$

where the standard deviation is defined as $\sigma = 1$ and $d = \ln 100$, such that $\frac{1}{e^d} = 0.01$.

The Equation 3.2 reveals that the covariance is characterized by an ellipse centered around $\begin{bmatrix} x_i \\ y_i \end{bmatrix}$ with $width = 2r_x$ and $height = 2r_y$. Given the area of the ellipse $A_{ellipse}$ and the aspect ration $a_r = \frac{r_x}{r_y}$, the horizontal and the vertical radii can be computed as follows:

$$r_x = \sqrt{\frac{A_{ellipse}a_r}{\pi}} \tag{3.3}$$

$$r_y = \sqrt{\frac{A_{ellipse}}{\pi a_r}} \tag{3.4}$$

Additionally, the volume fraction of each phase can be controlled, at least in an average sense, by defining a cutoff value z_{cutoff}. Elements with values below this cutoff are assigned to phase one, while those above are assigned to phase two. Given that each value z_i follows a Gaussian marginal distribution $z_i \sim \mathcal{N}(\mu_i, \Sigma_{ii} = \sigma^2 = 1)$, the probability that an element e_i belongs to phase one is given by

$$\Pr[z_i \leq z_{\text{cutoff}}] = \Phi\left(\frac{z_{\text{cutoff}} - \mu_i}{\sigma}\right) \tag{3.5}$$

Thus, with a fixed cutoff value z_{cutoff}, the desired volume fraction ϕ can be achieved on average by taking the mean μ_i as

$$\mu_i = z_{cutoff} - \sigma\Phi^{-1}(\phi) \tag{3.6}$$

Each sample was computed by drawing from an MVG as defined previously in this section. After generation, microstructures are assigned, depending on their pixel value, an elastic modulus of either E_1 or $E_2 = CR \cdot E_1$. For both phases, the same Poisson's ratio $\nu_1 = \nu_2 = \nu$ was chosen. For simplification, the following values are assumed: volume fraction $\phi = 0.5$, ellipse area $A_{ellipse} = 0.25$, aspect ratio $a_r = 1$, Young's modulus $E_1 = 1$ and Poisson's ratio $\nu = 0.3$.

3.2 Ground-truth Solution

Unlike Convolutional Neural Networks (CNNs) [37], PINNs do not necessarily require pre-generated data, as they are trained by approximating the governing physical equations. In contrast, the FNO method relies on substantial amounts of data for training, as it aims to approximate the mapping between the microstructure and the solution of the partial differential equations (PDEs). Therefore, a method for labeling the microstructures is needed.

There are several approaches for obtaining the solution of a PDEs. Analytical methods can be used only on a handful of well studied PDEs [41]. Although for the linear elasticity problem the analytical solution can be found [39], accurate numerical methods are desired for their uncomplicated implementations. Finite element method (FEM) is computationally very expensive [40] but it provides high accuracy predictions, that are often used as reference solution in different approaches [37]. Thus, FEM was chosen to obtain the ground truth solution of the PDE for the investigated microstructures. The FEM serves on one side for labeling the microstructures in the dataset for the FNO method, while also providing the reference solution for determining the accuracy of the PINN approach. Alternatively, for the latter method, the analytical solution for a certain problem configuration (source term + boundary conditions) was used for verifying the correctness of the implementation and the accuracy of the model [44].

The code for solving the linear elasticity problem using the finite element method is based on the implementation publicly available on the FEniCS website [42]. The code was adapted such that coordinates dependent source term and boundary conditions are allowed [43]. Moreover, Neumann boundary conditions, which are usually encountered when solving the linear elasticity PDE, are also taken into consideration.

Equation 2.4 can be rephrased as follows:

$$\int_\Omega \boldsymbol{\sigma} : \nabla \mathbf{v}\, dx = \int_\Omega \mathbf{f} \cdot \mathbf{v}\, dx + \int_{\partial\Omega_T} \mathbf{T} \cdot \mathbf{v}\, ds \tag{3.7}$$

where $\mathbf{T} = \boldsymbol{\sigma} \cdot \mathbf{n}$ is known as the *stress vector* at the boundary and is often prescribed as a boundary condition.

By combining the Equations 2.2 and 2.3 the direct dependency of the stress tensor $\boldsymbol{\sigma}$ on the displacement vector $\mathbf{u}$ can be obtained:

$$\boldsymbol{\sigma} = \lambda(\nabla \cdot \mathbf{u})\mathbf{I} + \mu\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right) \tag{3.8}$$

Inserting Equation 3.8 into Equation 3.7 gives the variational form with $\mathbf{u}$ as unknown. Moreover, because the inner product of a symmetric and an anti-symmetric tensor vanishes, the product $\boldsymbol{\sigma} : \nabla\mathbf{v}$ becomes $\boldsymbol{\sigma} : \boldsymbol{\epsilon}(\mathbf{v})$. Therefore:

$$\begin{gathered} a(\mathbf{u}, \mathbf{v}) = L(\mathbf{v}) \\ \leftrightarrow \\ \int_\Omega \boldsymbol{\sigma}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{v})\, dx = \int_\Omega \mathbf{f} \cdot \mathbf{v}\, dx + \int_{\partial\Omega_T} \mathbf{T} \cdot \mathbf{v}\, ds \end{gathered} \tag{3.9}$$

Equation 3.9 gets solved by the FEniCS solver using the FEM method. The returned solution is the displacement field $\mathbf{u}$. Using Equation 3.8 the stress field $\boldsymbol{\sigma}$ can easily be obtained.

3.3 Physics-informed Neural Network Architecture

This section of the thesis aims to provide a clear understanding of how the Physics-Informed Neural Network (PINN) model was implemented and how its performance was evaluated. The focus will be primarily on the conceptual framework, rather than the code implementation itself. For those interested in the coding aspects, the GitHub repository offers valuable insights.

3.3.1 Model architecture

The PINN used in this thesis is composed out of two separate neural networks: one for predicting the stress field $\boldsymbol{\sigma}$ and the other for predicting the displacement field $\mathbf{u}$. Both take as input the spatial coordinates of the sample $\mathbf{x}$ and $\mathbf{y}$.

It is common to assume from the start that $\sigma_{12} = \sigma_{21}$ and to only compute one of them. However, the output of the stress model in this thesis has one feature, for each of the stress tensor components. The reason for not making the previous assumption was to observe whether the model will predict by the end of the training that $\sigma_{12} = \sigma_{21}$.

Noteworthy is the PINN model used in [44], which implements a single network with 5 output features $u_x, u_y, \sigma_{11}, \sigma_{12}, \sigma_{22}$. For the same problem configuration (source term + boundary conditions), both PINN models seem to have similar accuracies.

Table 3.1 presents the analyzed configurations for the PINN models utilized in 3.3.3. In this context, the letter **S** denotes the stress field model, while the letter **D** represents the displacement field model. The second column of each table lists the number of neurons in each layer of the network, whereas the last column indicates the total number of parameters for the model. Notably, all models employ the Tanh activation function across all layers, with the exception of the final layer, where no activation function is applied.

3.3.2 Loss function

As previously discussed in Section 2.4.2, the essence of Physics-Informed Neural Networks (PINNs) is rooted in the definition of the loss function. Unlike most neural networks that rely on predefined loss functions, PINNs necessitate a user-defined loss function tailored to the

S1	[2, 40, 40, 40, 40, 40, 4]	4,864	D1	[2, 40, 40, 40, 40, 40, 4]	4,864
S2	[2, 40, 40, 40, 4]	3,564	D2	[2, 40, 40, 40, 2]	3,482
S3	[2, 40, 50, 40, 20, 4]	5,114	D3	[2, 40, 50, 40, 20, 2]	5,072

Table 3.1: **First column**: Model Name; **Second column**: Network structure; **Third column**: Number of parameters

specific problem being addressed. In this subsection, the definition of the loss function for the linear elasticity problem will be addressed.

While equation 2.9 contains terms for both the data and physics losses, the PINN model implemented in this thesis has been trained and evaluated only on the latter one.

$$\mathcal{L} = \alpha_0 L_{\text{Inner}} + \alpha_1 L_{\text{Boundary}} + \alpha_2 L_{\text{InnerU}} + \alpha_3 L_{\text{BoundaryU}} + \alpha_4 L_{\sigma} \tag{3.10}$$

where $\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4$ are hyperparameters and the loss components are defined as:

$$\begin{aligned}
L_{\text{Inner}} &= \frac{1}{N_i} \sum_{i=1}^{N_i} \|\nabla \cdot \boldsymbol{\sigma}_{NN}^{(i)} + \mathbf{f}^{(i)}\|_2^2 \\
L_{\text{Boundary}} &= \frac{1}{N_{b1}} \sum_{i=1}^{N_{b1}} \|\boldsymbol{\sigma}_{NN}^{(i)} \cdot \mathbf{n}^{(i)} - \mathbf{t}_{boundary}^{(i)}\|_2^2 \\
L_{\text{InnerU}} &= \frac{1}{N_i} \sum_{i=1}^{N_i} \|\lambda^{(i)} \text{tr}(\boldsymbol{\varepsilon}_{NN}^{(i)})\mathbf{I} + 2\mu_{NN}^{(i)} \boldsymbol{\varepsilon}_{NN}^{(i)} - \boldsymbol{\sigma}_{NN}^{(i)}\|_2^2 \\
L_{\text{BoundaryU}} &= \frac{1}{N_{b2}} \sum_{i=1}^{N_{b2}} \|\mathbf{u}_{NN}^{(i)} - \mathbf{u}_{boundary}^{(i)}\|_2^2 \\
L_{\sigma} &= \frac{1}{N} \sum_{i=1}^{N} \left(\sigma_{12}^{(i)} - \sigma_{21}^{(i)}\right)^2
\end{aligned} \tag{3.11}$$

where N_i is the number of inner points, N_{b1} and N_{b2} are the number of boundary points with *stress vector*, respectively *displacement vector* boundary conditions, and $N = N_i + N_{b1} + N_{b2}$ is the total number of discretized points on the image.

The terms L_{Inner} and L_{InnerU} evaluate the satisfaction of Equations 2.1, 2.2, and 2.3, whereas L_{Boundary} and $L_{\text{BoundaryU}}$ assess the fulfillment of the boundary conditions. L_{σ} checks the similarity of the σ_{12} and σ_{21} components of the stress tensor.

3.3.3 Hyperparameters tuning

Hyperparameters tuning is an important procedure when creating a neural network, as it influences greatly the training process and the accuracy of the model. The focus of this subsection lies on analyzing what configuration of hyperparameters gives the most accurate results. The parameters varied in this section are listed in Table 3.2.

Model structure

The structure of the networks plays a key role in the training process, and therefore has a great impact on the accuracy of the model. A deeper network, characterized by an increased number of layers, can capture intricate relationships but may also risk overfitting, especially

Hyperparameter	Value
Stress model	S1, S2, S3
Displ. model	D1, D2, D3
Num. innerPoints	500, 5000, 50000
Num. boundaryPoints	500, 5000, 10000
Batch size	100, 500, 1000
Learning rate	0.001, 0.005, 0.01
Num. epochs	500

Table 3.2: PINN: Table of varied hyperparameters.

when trained on limited datasets. Conversely, a shallower architecture may struggle to model the underlying complexities, resulting in underfitting.

A total of 9 combinations of stress and displacement neural network structures were analyzed in this thesis. The full configuration for this investigation is presented in Table 3.3.

Hyperparameter	Value
Stress model	S1, S2, S3
Displ. model	D1, D2, D3
Num. innerPoints	5000
Num. boundaryPoints	5000
Batch size	500
Learning rate	0.001
Num. epochs	500

Table 3.3: PINN: Parameters varied for the Model Configuration investigation. Only the values of *Stress model* and *Displ. model* are varied. The rest are fixed.

As illustrated in Figure 3.1(top), all model configurations demonstrate strong performance in predicting the stress field. By epoch 500, the model with the lowest performance, (S3, D2), achieves a relative error of just 1%. Furthermore, it is noteworthy that none of the models have reached a plateau by this epoch, indicating that continuing to increase the number of epochs could yield even greater improvements in results. Figure 3.1(bottom) offers valuable insights for selecting the most suitable model configuration. One noteworthy observation is the instability present in the displacement model, which may arise from its limited access to direct displacement information. It's important to note that all data points providing direct measurements of displacement are confined to one of the four boundaries of the material. This boundary-focused data can restrict the model's ability to accurately capture displacement values throughout the entire domain. Additionally, the displacement model is trained concurrently with the stress model, creating a significant interdependence between the two. This means that any inaccuracies or fluctuations in the stress model can directly impact the performance of the displacement model.

The **S1** model configurations show a tendency to overfit, likely due to its relatively simple architecture of five hidden layers with 40 neurons each. In contrast, **S2** achieves superior results, despite having fewer parameters than **S1**, suggesting that fewer layers can sometimes enhance accuracy. Configurations involving **S3** perform worse, despite its pyramid-like structure, which is typically expected to deliver better results. Therefore, **S2** and **D3** are chosen for further investigation, as this combination demonstrates the most promising performance.

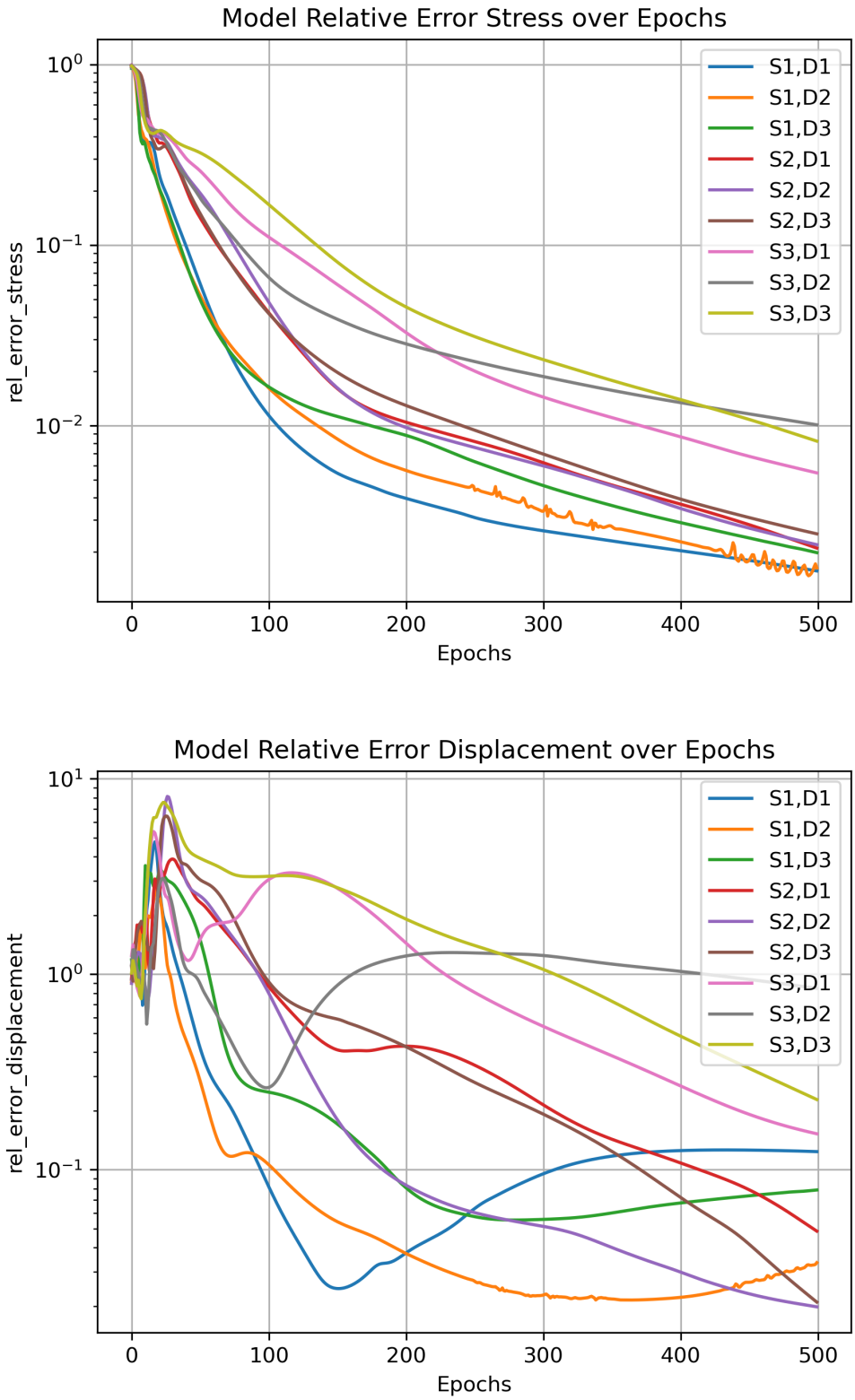

Figure 3.1: Comparison of different PINN model configurations. The top plot shows the relative errors for the stress field (σ), while the bottom plot depicts the relative errors for the displacement field ($\mathbf{u}$).

Number of points and batch size

In this experiment the effect of the number of points and the batch sizes on the predictive abilities of the PINN models is investigated. Table 3.4 shows the configurations of inner and boundary points and batch sizes considered. The models used in this experiment are (S2, D3) and the training is done over 500 epochs, using a learning rate of 0.001.

Set	1	2	3	4	5	6	7	8	9
InnerPoints	500	5000	5000	5000	5000	50000	50000	50000	50000
BoundaryPoints	500	500	5000	5000	5000	500	5000	5000	5000
Batch Size	100	100	100	500	1000	100	100	500	1000

Table 3.4: Investigated sets of InnerPoints, BoundaryPoints and Batch Sizes

The evolution of the relative errors is illustrated in Figure 3.2. For the stress model, all plots exhibit similar trends; however, the green and pink curves are particularly noteworthy. By epoch 500, these configurations achieve impressive relative errors below 0.1%, while the remaining sets maintain relative errors under 1%, which is still a satisfactory outcome. The lower section of Figure 3.2 further emphasizes the superior performance of the third and seventh sets, which demonstrate enhanced accuracy. In contrast, for the displacement model, the green plot shows a relative error of 3.4% by epoch 500, while the pink plot achieves 1.3%. Notably, since none of the plots have plateaued, it is likely that increasing the number of epochs will reduce the relative errors below 1%. This suggests that both sets are viable options for further consideration. However, because of the significantly shorter training time for the third set, this is preferred.

Learning rate

The learning rate is an important setting in training neural networks that controls how quickly the model learns. If the learning rate is too high, the model might jump around and miss the best solution. On the other hand, if it's too low, the learning process can drag on and get stuck in less optimal solutions. Finding the right balance for the learning rate is key to helping the model learn effectively and perform well.

Hyperparameter	**Value**
Model	(S2, D3)
Num. innerPoints	5000
Num. boundaryPoints	5000
Batch size	100
Learning rate	0.001, 0.005, 0.01
Num. epochs	500

Table 3.5: PINN: Varied parameters for Learning Rate investigation. Only the values of *Learning rate* are varied. The rest are fixed.

Three learning rates were investigated. Table 3.5 shows the considered hyperparameters for this experiment. It can be seen in Figure 3.3 that for both the stress and the displacement models the frequency and the amplitude of the plot oscillations are increasing with increased learning rate. A high learning rate causes the model to make large weight updates, leading

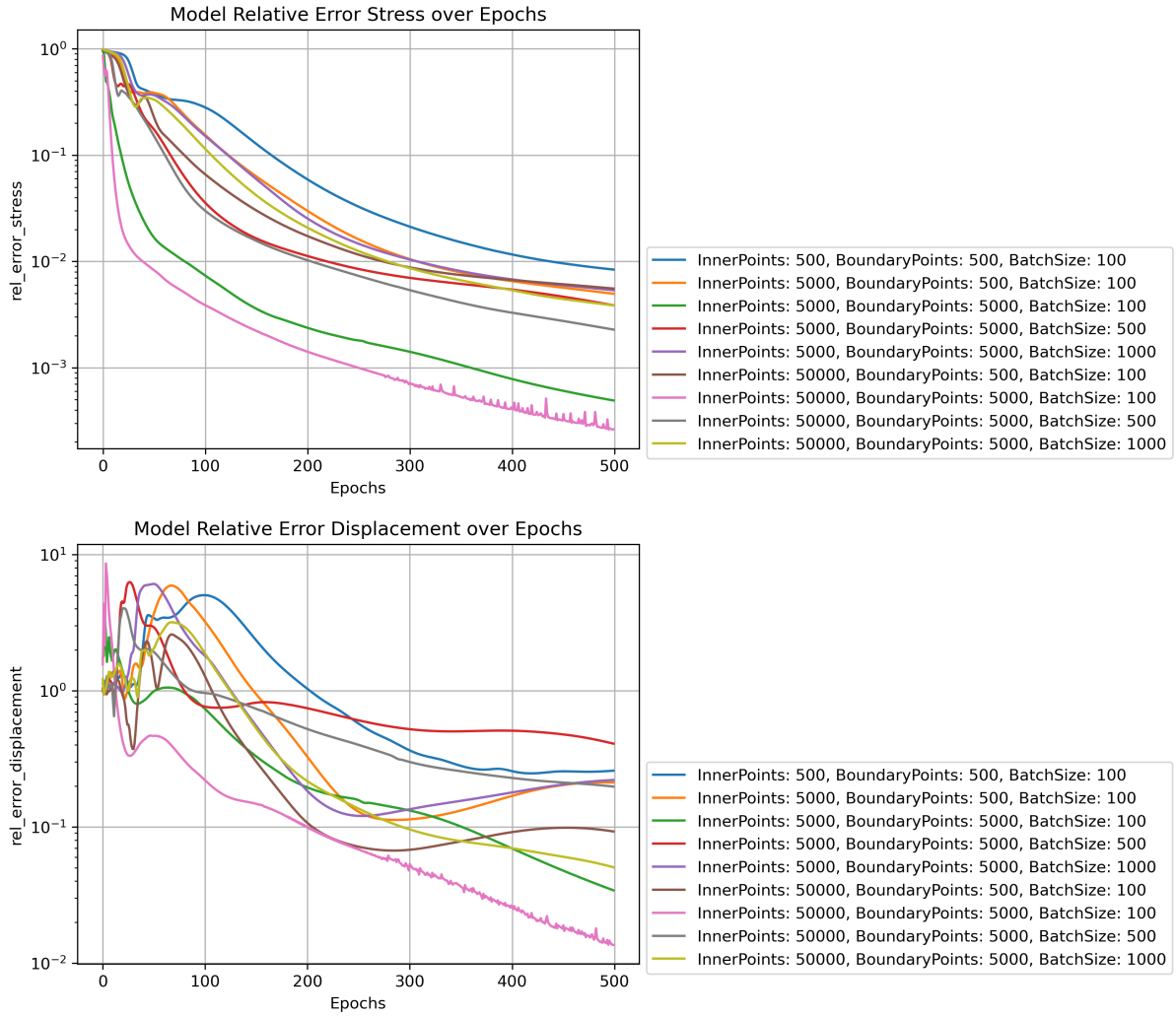

Figure 3.2: Comparison of different InnerPoints, BoundaryPoints and Batch Sizes.

to significant fluctuations instead of steady convergence. Therefore, this experiment clearly indicates that a Learning Rate of 0.001 is desired.

3.3.4 Training

The preceding graphs were generated by training the models on both interior and boundary points within the domain. While the boundary points were uniformly distributed along the edges, the inner points were created randomly within the domain. Although an attempt was made to generate the inner points uniformly, this approach yielded suboptimal results compared to the random method. This discrepancy may be attributed to the PINN method's limitations in generalizing effectively with uniformly distributed points.

The model parameters were initialized using Xavier initialization, which demonstrated a notable improvement in accuracy during testing.

Although L2 regularization with a weight decay of 1×10^{-4} was not initially included in the hyperparameter tuning process, it will be utilized in Chapter 4 to enhance model performance.

Hyperparameter	Model	Num. InnerPoints	Num. BoundaryPoints	Batch size	Learning rate
Value	(S2, D3)	5000	5000	100	0.001

Table 3.6: PINN: Final Configuration.

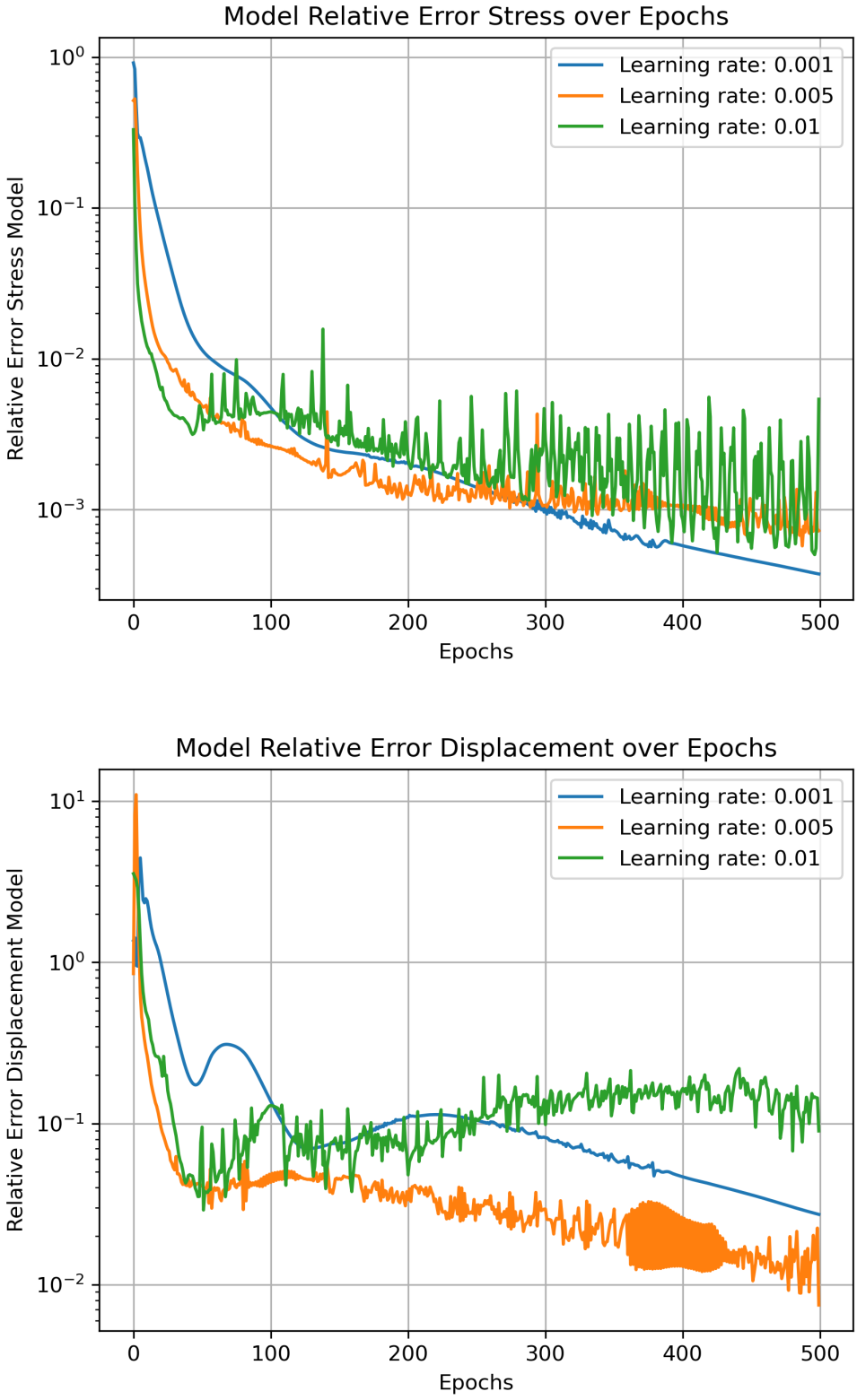

Figure 3.3: Comparison of PINN performance for different learning rates. The top plot illustrates the relative errors for the stress field (σ), while the bottom plot shows the relative errors for the displacement field ($\mathbf{u}$).

3.4 Fourier Neural Operator Architecture

This section of the thesis is dedicated to elucidating the essential technical aspects of the Fourier Neural Operator (FNO) implementation. Rather than delving into the code itself, the emphasis will be placed on the underlying conceptual framework. For those interested in the coding details, the GitHub repository provides valuable insights related to both the Fourier Neural Operator (FNO) models and the Physics-Informed Neural Network (PINN).

3.4.1 Model architecture and Dataset shape

The FNO implementation heavily relies on the Neuraloperator library [46] provided by the authors of [35]. The library employs the neural network architecture explained in Section 2.4.3 and plotted in Figure 2.4.

The performance of the model is significantly influenced by the quality of the dataset on which it is trained. As previously noted, the microstructures and their corresponding solutions were derived by solving the linear elasticity problem using the Finite Element Method. Subsequently, the input and output datasets were generated based on a uniformly discretized set of points across the microstructure. The shape of the input dataset is given by $(N, \text{res}_x, \text{res}_y, 3)$, while the output dataset has a shape of $(N, \text{res}_x, \text{res}_y, 6)$, where N is the number of samples, res_x and res_y are the resolutions in the x and y directions, and 3 and 6 are the number of channels.

The first channel of the input contains the Young's Modulus for the discretized points, while the other 2 channels their spatial coordinates. Including spatial coordinates in the input dataset of a Fourier Neural Operator (FNO) model is essential because it helps the model understand how physical phenomena vary across space. These coordinates provide crucial context for learning patterns and relationships within the data, enabling the model to capture the dynamics of spatially-dependent processes accurately. Without this information, the FNO would struggle to make reliable predictions, as it wouldn't recognize how variables change from one location to another.

The output dataset comprises six channels, which include the two components of the displacement vector and the four components of the stress tensor. The relationship between displacement and stress, as described by Equation 3.8, involves gradients that can be computed on a discretized mesh; however, this approach often yields inaccurate results. Therefore, it is essential for the FNO model to predict both fields to ensure reliable outcomes.

3.4.2 Loss function and Accuracy

For the FNO model, all trainings and comparisons were done using the L2Loss, implemented directed in the neuraloperator library [46]. This loss function was chosen as it computes precisely the metric of interest, namely the relative error.

$$L_2(\mathbf{y}_{NN}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=1}^{N} \frac{\|\mathbf{y}_{NN}^{(i)} - \hat{\mathbf{y}}^{(i)}\|_2^2}{\|\hat{\mathbf{y}}^{(i)}\|_2^2} \tag{3.12}$$

where N is the number of samples, $\mathbf{y}_{NN}$ is the prediction generated by the model, and $\hat{\mathbf{y}}$ is the corresponding ground truth solution.

The training dataset comprises 80% of the total data, while the validation dataset, which is used to assess the model's accuracy, constitutes the remaining 20%. The key distinction

between the loss computation during training and the relative error assessment during validation is that, during training, the average loss is calculated per batch, whereas in validation, the average relative error is computed per individual sample.

3.4.3 Hyperparameter tuning

This subsection of the thesis is dedicated to analyzing various hyperparameter configurations to identify the optimal performing model. Unlike the Physics-Informed Neural Network, the Fourier Neural Operator (FNO) offers limited flexibility in adjusting the model structure. Therefore, the emphasis in this section will be placed primarily on the data utilized for training. The parameters varied in this analysis are listed in Table 3.7.

Hyperparameter	Value
Resolution	32×32, 64×64, 128×128, 256×256
Num. modes	'quarter', 'half'
Num. hiddenChannels	32, 64, 128, 256, 512
Num. samples	500, 1000, 1500
Batch size	32, 64, 128

Table 3.7: FNO: Table of varied hyperparameters.

Resolution and Number of modes

Resolution and the number of modes are vital factors influencing the performance of Fourier Neural Operators (FNOs). Resolution determines the level of detail the model can capture in the input data, referring to the number of points in the microstructure for each sample and channel. Higher resolution enables the model to learn intricate patterns and relationships more effectively, leading to better prediction accuracy. However, excessively high resolution can incur unnecessary computational costs and risk overfitting.

Similarly, the number of modes impacts FNO performance by representing the frequency components used to approximate functions within the model. More modes allow the FNO to capture complex patterns and variations, enhancing its ability to model intricate dynamics. Yet, using too many modes can lead to overfitting, making the model overly specialized to the training data and less effective with new data. Finding the right balance between resolution and modes is essential for ensuring adaptability across various scenarios while maintaining efficiency, making their optimization crucial for maximizing FNO performance.

A total of 8 combinations of resolutions and number of modes were analyzed in this section. The full configuration of this investigation is presented in Table 3.8.

Hyperparameter	Value
Resolution	32×32, 64×64, 128×128, 256×256
Num. modes	'quarter', 'half'
Num. hiddenChannels	64
Num. samples	500
Batch size	32

Table 3.8: FNO: Parameters varied for the Resolution and Num. of Samples investigation. The rest are fixed.

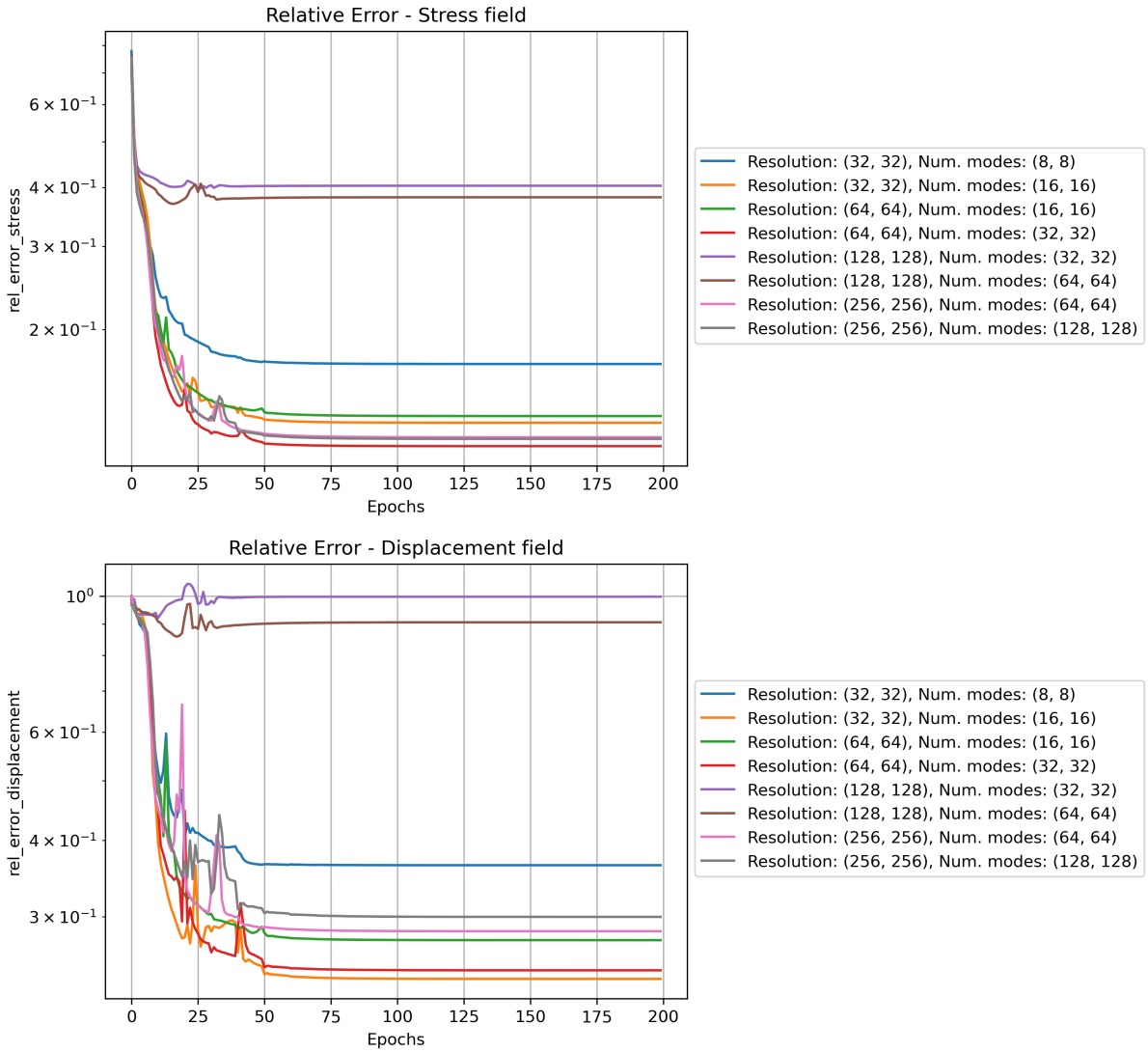

Figure 3.4: Comparison of FNO performance for different resolutions and number of modes.

Figure 3.4 illustrates the relative errors associated with various configurations of resolution and the number of modes. In both plots for a resolution of 128, the relative errors are notably high for both the stress and displacement fields. The red plot, which corresponds to a resolution of 64×64 and 16×16 modes, ranks among the top performers in both graphs, indicating that this configuration is a strong candidate for the model. Additionally, the orange plot merits attention, as it shows the best relative error for the displacement field.

Given that the configuration with a resolution of 64×64 and 32×32 modes has the potential to capture more features from a potentially more complex dataset, it will be prioritized as the preferred choice.

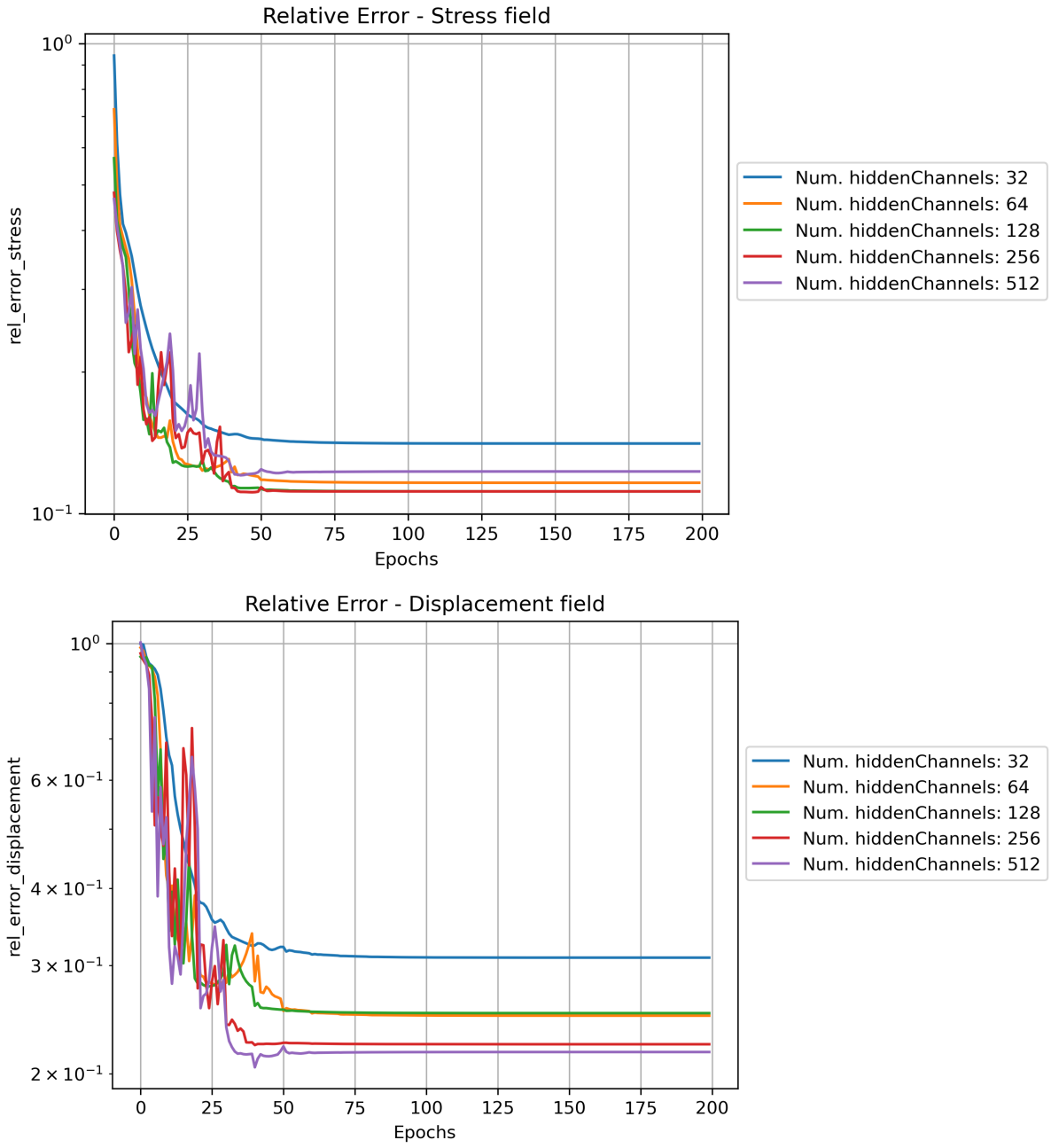

Figure 3.5: Comparison of FNO performance for different number of hidden channels.

Number of hidden channels

The hidden channels serve as additional layers of abstraction, allowing the model to learn and represent complex features from the input data more effectively. By increasing the number of hidden channels, the FNO can capture more complex pattern, enhancing its ability to approximate the underlying functions of the problem being solved. Similarly to other hyperparameters, a too high number of hidden channels raises the risk of overfitting, especially if the model becomes too complex relative to the amount of trianing data available. Table 3.9 shows the varying hyperparameters for this experiment.

Hyperparameter	Value
Resolution	64×64
Num. modes	'half'
Num. hiddenChannels	32, 64, 128, 256, 512
Num. samples	500
Batch size	32

Table 3.9: FNO: Parameters varied for Hidden Channels investigation. Only the values of *Num. hiddenChannels* are varied. The rest are fixed.

Figure 3.5 illustrates the evolution of the relative error for both the stress and displacement fields over the course of the epochs. As expected, the configuration with 32 hidden channels exhibits the poorest performance. Among the stress components, the other four configurations demonstrate relatively similar results. However, for the displacement components, the yellow and green plots overlap, suggesting comparable performance, while the red and purple plots perform better than the previous two. Overall, the model employing 256 hidden channels emerges as the top performer.

Number of samples and Batch size

The training process for the FNO model contrasts with that of Physics-Informed Neural Networks (PINNs) in that it depends on external data to identify the patterns essential for approximating the underlying system function. Since generating datasets and training FNOs are both resource-intensive and time-consuming, it is important to identify the smallest dataset size that still yields satisfactory results. Additionally, the influence of batch size will be evaluated to further optimize the training procedure.

Hyperparameter	Value
Resolution	64×64
Num. modes	'half'
Num. hiddenChannels	64
Num. samples	500, 1000, 1500
Batch size	32, 64, 128

Table 3.10: FNO: Parameters varied for Number of Samples and Batch Sizes investigation. The rest are fixed.

Figure 3.6 shows a comparison between 9 combinations of number of samples and batch sizes. As expected, higher the number of samples and lower the batch size, more accurate results for both of the models. Therefore, the best result is obtained for the configuration made of 1500 number of samples and batch size of 32. However, the data generation and the training time in this case is significantly higher than the cases with 1000 or 500 samples. Considering the need to generate even more datasets and to train different models for multiple scenarios in the next chapter, the chosen configuration for further investigation will be made out of 1000 samples and batch size of 64.

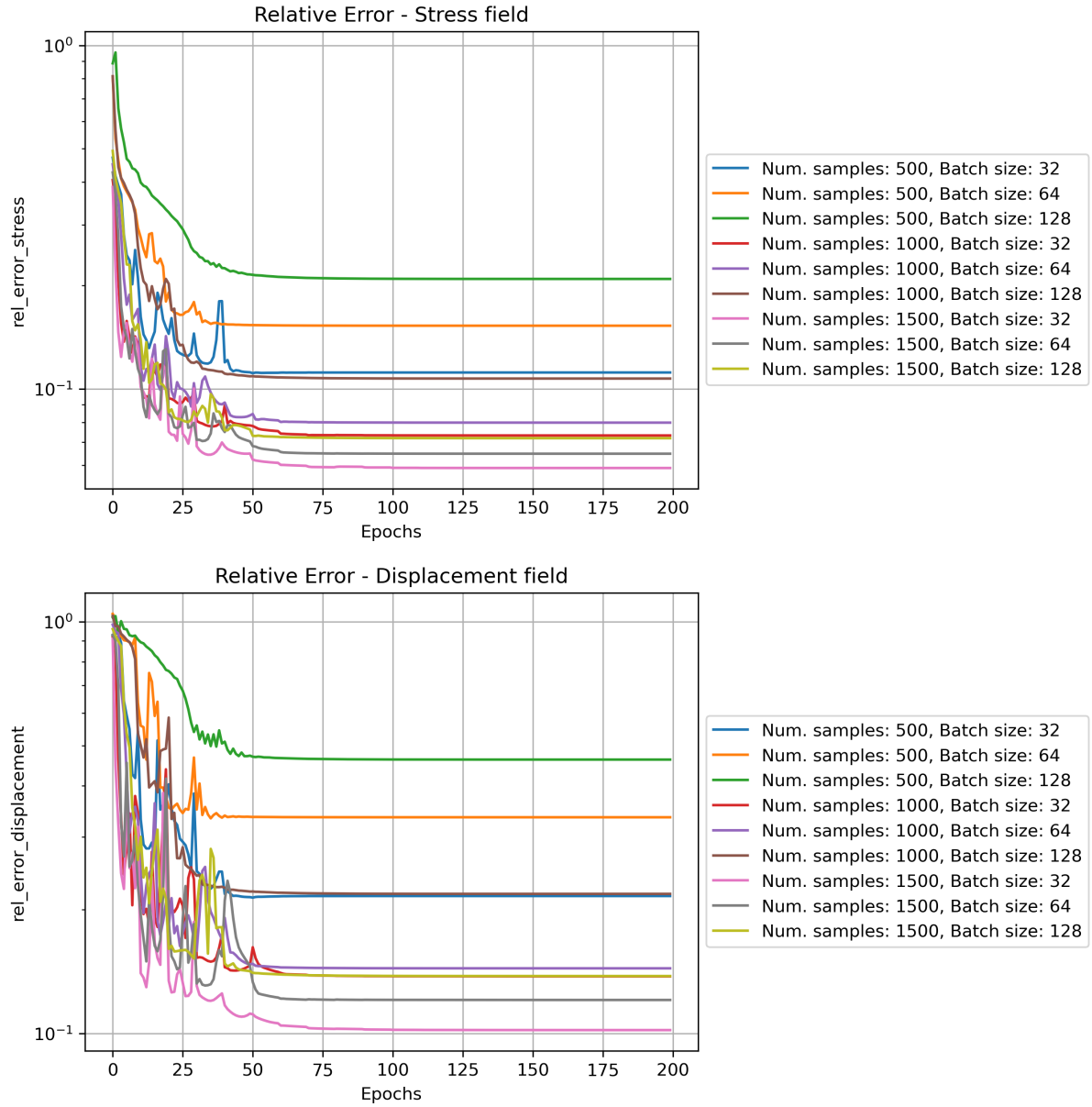

Figure 3.6: Comparison of FNO performance for different numbers of samples and batch sizes. The top plot shows the relative errors for the stress field (σ), while the bottom plot illustrates the relative errors for the displacement field ($\mathbf{u}$).

3.4.4 Training

During the training, a scheduler was implemented with a step size of 10 and a gamma value of 5. A learning rate scheduler can impact significantly the accuracy because it dynamically adjusts the learning rate throughout the training process. This adjustment helps the model converge more effectively, avoiding issues like overshooting the loss minimum or stagnating. Common strategies include step decay, which reduces the learning rate at specific intervals, and adaptive methods that adjust based on validation performance. By fine-tuning the learning rate, a scheduler improves the model's learning efficiency, leading to enhanced performance and faster convergence.

Training a Fourier Neural Operator (FNO) model is a time-consuming process, as generat-

ing the dataset and training the model require substantial effort. This reinforces the idea that while the Finite Element Method (FEM) is highly accurate, it is also computationally expensive. In contrast, a neural network can provide immediate predictions once it has been trained.

Hyperparameter	Resolution	Num. modes	num. hiddenChannels	Num. samples	Batch size
Value	64×64	32×32	256	1000	64

Table 3.11: FNO: Final configuration

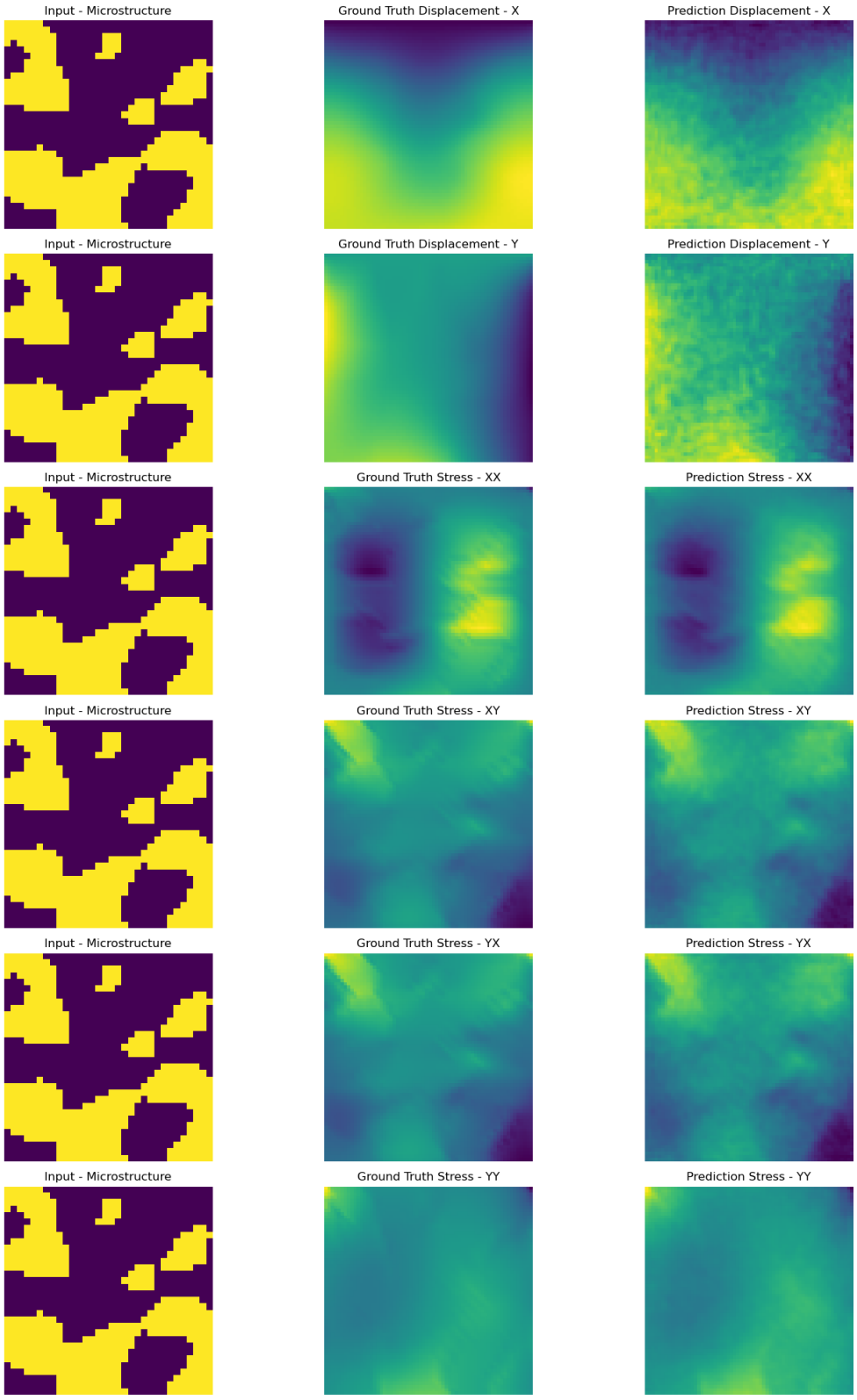

Figure 3.7: Comparison between the prediction and the ground truth solution for the model defined in Table 3.11. **Relative Errors:** Displacement - X: 11.88%, Displacement - Y component: 37.02%, Stress - XX: 7.17%, Stress - XY: 11.88%, Stress - YX: 12.11%, Stress - YY: 13.6%.

Chapter 4

Results and Discussion

The previous chapter focused on presenting and explaining the technical foundations of the Physics-Informed Neural Network (PINN) and the Fourier Neural Operator (FNO). In this chapter, attention is directed toward an in-depth analysis of the performance and behavior of these two models under varying contextual conditions. This investigation is essential to the research, as it assesses how each model responds to different material types, force loads, and displacement constraints. Accordingly, the primary focus of the analysis includes factors such as contrast ratio, image resolution, and problem setup, defined by source terms and boundary conditions.

For the purposes of this thesis, the analysis is limited to the configurations defined by the hyperparameters in Tables 4.1 and 4.2. These configurations were selected based on comparative analysis of multiple hyperparameter values, as they demonstrated the most favorable performance. It should be noted that in the analyses presented in the previous chapter, the image resolution was set to 32, and the problem setup corresponded to the first entry in Table 4.3. In the case of the PINN method, the contrast ratio was set to 1, while for the FNO models the contrast ratio was set to 2. While these configurations are likely optimal for other contrast ratios, image sizes, and problem setups, alternative scenarios may benefit from different configurations.

Hyperparameter	Model	Num. InnerPoints	Num. BoundaryPoints	Batch size	Learning rate
Value	(S2, D3)	5000	5000	100	0.001

Table 4.1: PINN configuration

Hyperparameter	Resolution	Num. modes	num. hiddenChannels	Num. samples	Batch size
Value	64×64	32×32	256	1000	64

Table 4.2: FNO configuration

4.1 Investigation of the Contrast Ratio

The contrast ratio plays a critical role in shaping the mechanical behavior of two-phase materials. Higher contrast ratios often lead to steeper gradients at phase boundaries, which can significantly challenge the ability of numerical methods to accurately predict solutions. In this

experiment, the effect of the contrast ratio on the predictive performance of the model was examined.

The analysis considered contrast ratios $CR \in \{1, 2, 5, 10, 20\}$. First, the results for the PINN and FNO methods are discussed individually, followed by a comparative analysis between the two approaches.

PINN

Figure 4.1 illustrates the evolution of relative errors over epochs for the stress and displacement models. For $CR = 1$, the PINN performs exceptionally well, achieving relative errors below 1% for both models, as expected from earlier findings. In the stress error plot (top), a clear trend emerges: as the contrast ratio increases, so does the relative error. The most significant gap is observed between $CR = 1$ and $CR = 2$, highlighting the substantial impact of material heterogeneity on the predictive capabilities of the PINN. The displacement error plot (lower) reinforces this trend but also reveals that for $CR \in \{2, 5, 10\}$, the relative errors converge to similar values after approximately 500 epochs. Interestingly, the model with the highest contrast ratio ($CR = 20$) consistently maintains a relative error close to 1, indicating significant difficulty in capturing the solution. Additionally, while previous results showed that the stress model generally outperforms the displacement model in terms of error, this trend diminishes as the contrast ratio increases, suggesting that high material heterogeneity affects both predictions more equally.

Most studies on PINNs typically train models for significantly longer durations, often tens of thousands to millions of epochs. However, due to the high computational cost of training multiple models in this thesis, the number of epochs was limited to 1000. Despite this limitation, the results clearly demonstrate that while the PINN architecture excels for low contrast ratios, it struggles to handle the steep gradients and complex behaviors associated with high contrast ratios, underscoring the need for more advanced architectures or techniques to address these challenges effectively.

FNO

As shown in Figure 4.2, the trends observed for the PINN method are also present for the FNO. For the homogeneous case ($CR = 1$), the FNO achieves exceptional results, even outperforming the PINN, with relative errors decreasing rapidly and converging to very low values. This highlights the FNO's strong ability to handle homogeneous materials effectively.

The significant gap between the blue ($CR = 1$) and orange ($CR = 2$) plots again underscores the increased difficulty of predicting the behavior of heterogeneous materials compared to homogeneous ones. This difficulty is further emphasized by the consistent trend of increasing relative error with higher contrast ratios ($CR = 5, 10, 20$). However, compared to the PINN results, the FNO shows a more stable convergence and lower relative errors even for higher contrast ratios, indicating a better suitability for heterogeneous materials. Noteworthy is that no visible improvement can be seen for the FNO models after approximately 100 epochs, while PINNs are typically trained for significantly more epochs, usually exceeding 1000 epochs.

These results suggest that both methods struggle with increasing contrast ratios. Even though FNO has an advantage over the PINN, the relative errors observed in this study remain unsatisfactory for a reliable PDE solver in highly heterogeneous materials, underscoring the need for further advancements in model architectures or training strategies.

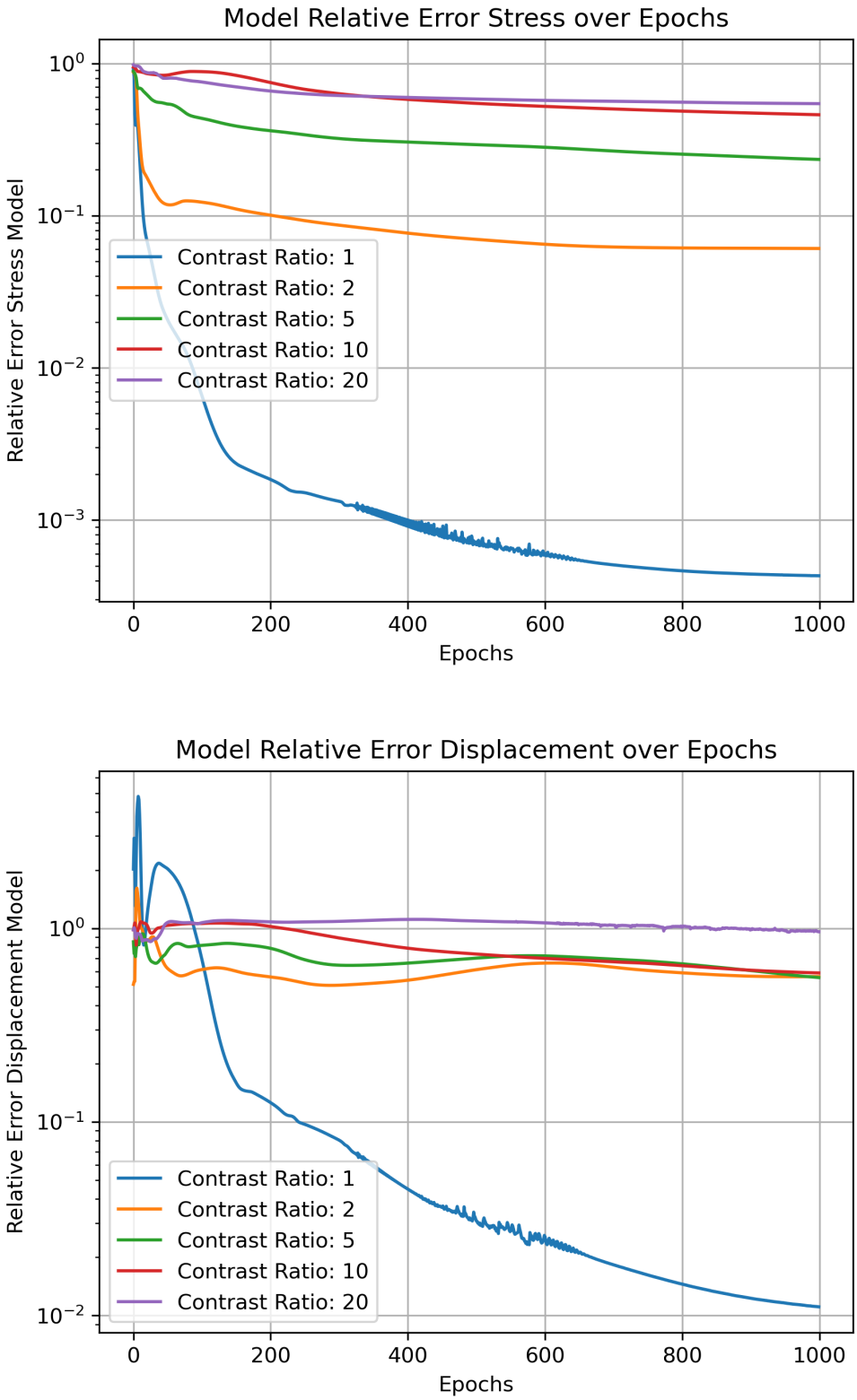

Figure 4.1: PINN: Relative error comparison for different contrast ratios. The top plot shows the relative errors for the stress field (σ), while the bottom plot illustrates the relative errors for the displacement field ($\mathbf{u}$).

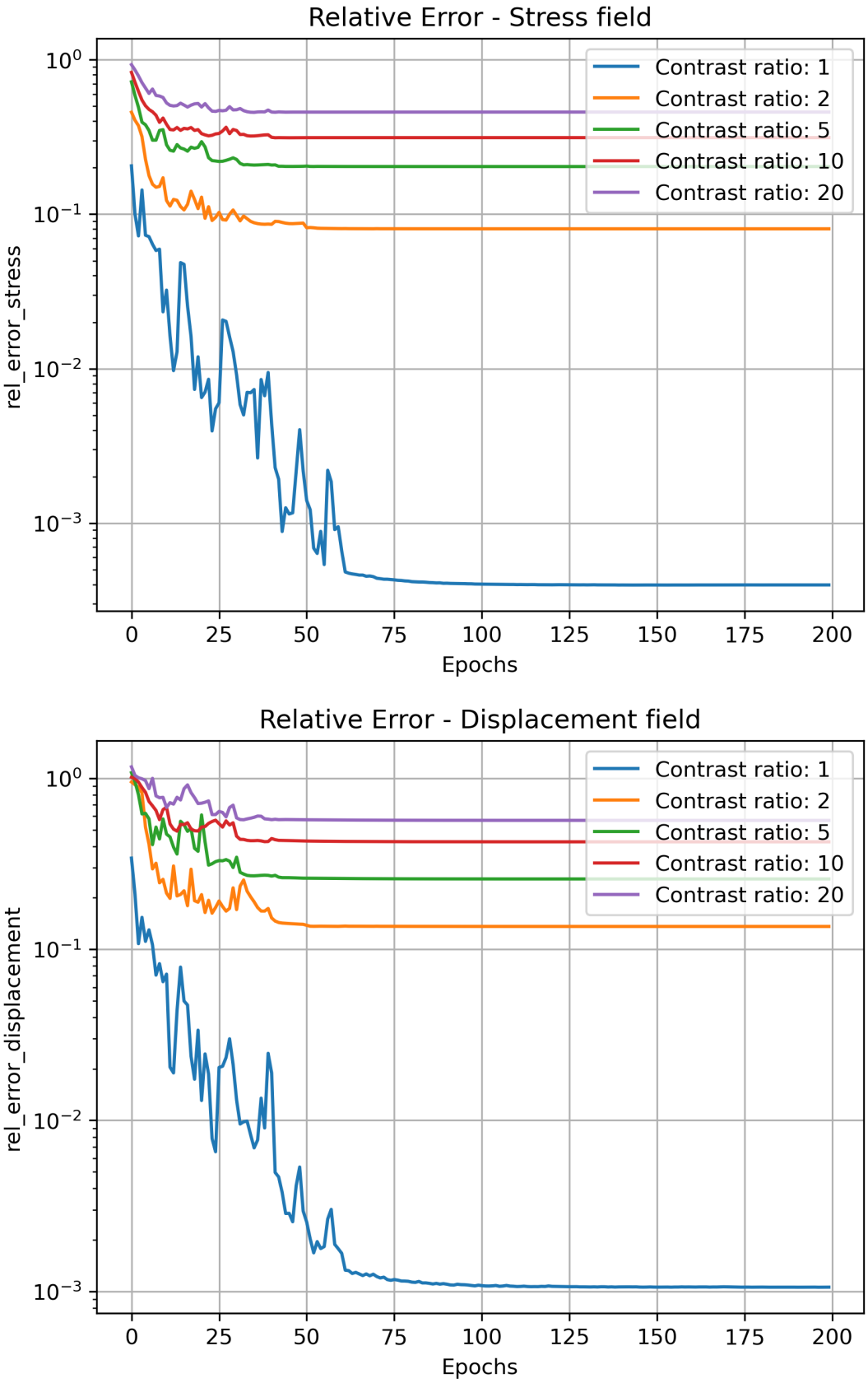

Figure 4.2: FNO: Relative error comparison for different contrast ratios. The top plot shows the relative errors for the stress field (σ), while the bottom plot illustrates the relative errors for the displacement field ($\mathbf{u}$).

4.2 Investigation of the Image Size

Section 3.1 has offered the reader a comprehensive explanation about the process of generating microstructures. Besides the variables needed for determining the covariance matrix, which were set to fixed values, the most important characteristic of a microstructure is the image size. In this experiment, the microstructure resolution is varied such that the performance of the methods to be observed for changing material complexity.

The analysis considers image sizes $ImageSize \in \{16, 32, 64\}$. Both the PINN and FNO are analyzed individually and then a comparison between the two is made.

PINN

Figure 4.3 shows clearly a separation of two groups of lines, for both the stress and the displacement model, corresponding to the contrast ratio. This confirms one more time the difference in the performance of the PINN between the cases of homogeneous and heterogeneous materials.

A contrast ratio of 1 means that both phases have the same Young's modulus, which makes the resolution of the image irrelevant for the performance of the PINN. However, in both the upper and lower pictures in Figure 4.3 a significant gap can be seen between the 3 plots corresponding to $CR = 1$. This suggests that multiple reruns of the model on the same material, under the same mechanical conditions can return similar, but different results, which is expected from a deep learning model.

Contrary to the other two experiments, where the same microstructure was used for all the comparison cases, this investigation deals with newly generated materials for each case, accordingly to the image size. This can bring inconsistencies between the results in different comparisons for the same case.

FNO

Figure 4.4 shows a clear tendency that the relative error is increasing with increasing image size, which is present in both models. This indicates that FNO encounters slight difficulties with more complex shapes of the phases regions. Noteworthy is the small size of the gaps between the plots. Between the blue $(ImageSize = 16)$ and the green $(ImageSize = 64)$ plots there is a difference of approximately 2% for both the stress and the displacement, even though the image size was quadrupled. This suggests that although the image resolution plays a role in the accuracy of the FNO model, it is not so important.

In comparison to PINN, it can be seen that FNO performs slightly worse for lower image sizes, but much better for higher image sizes. This shows that FNO has the potential to be a reliable PDE solver for heterogeneous materials with more complex microstructures. However, no investigation on very high image sizes was done, due to the long data generation and model training time. An interesting result for future research would be the performance of the PINN and FNO for $ImageSize \to \infty$, which simulates materials with smoothly shaped phases.

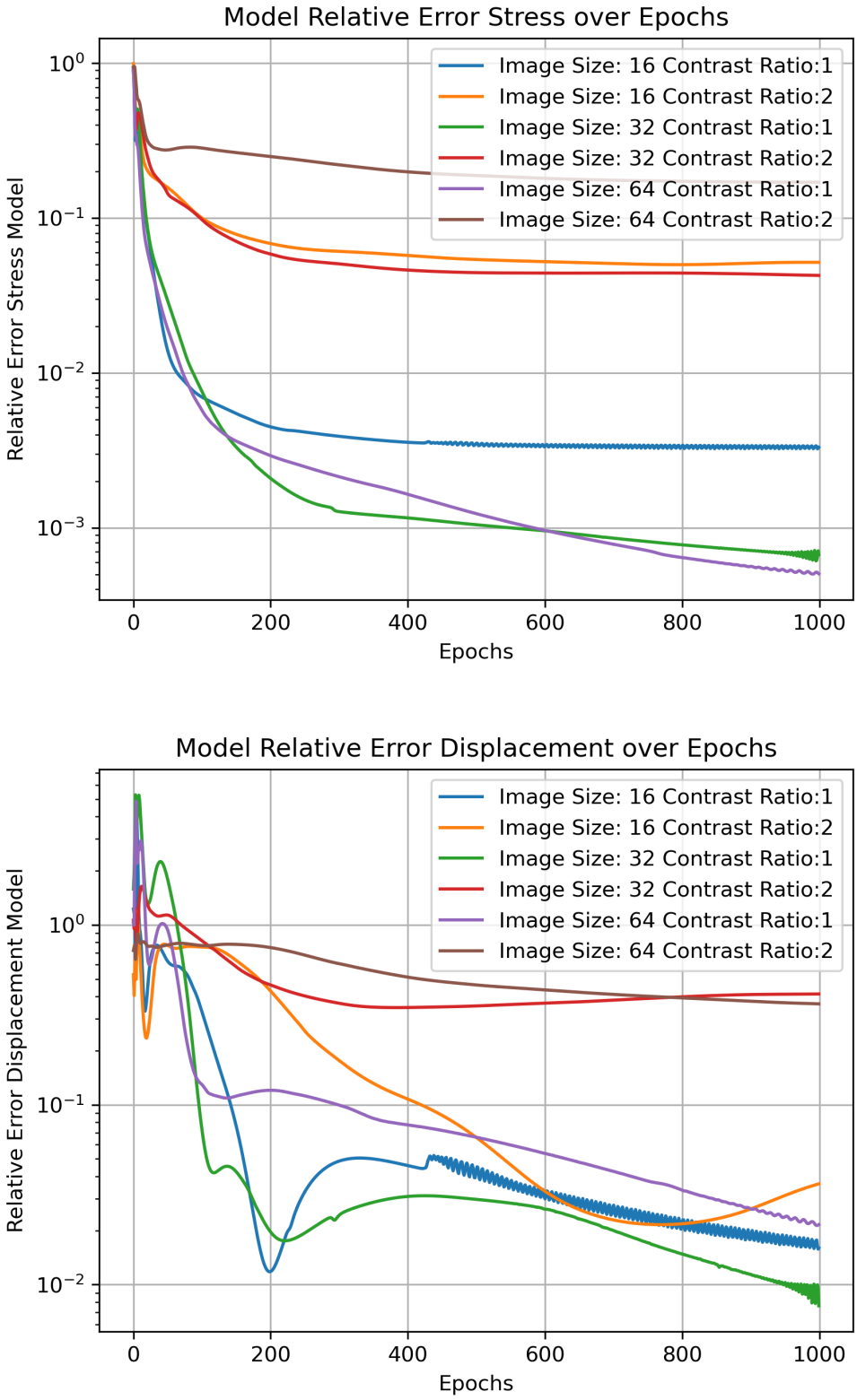

Figure 4.3: PINN: Relative error comparison for different image sizes. The top plot shows the relative errors for the stress field (σ), while the bottom plot illustrates the relative errors for the displacement field ($\mathbf{u}$).

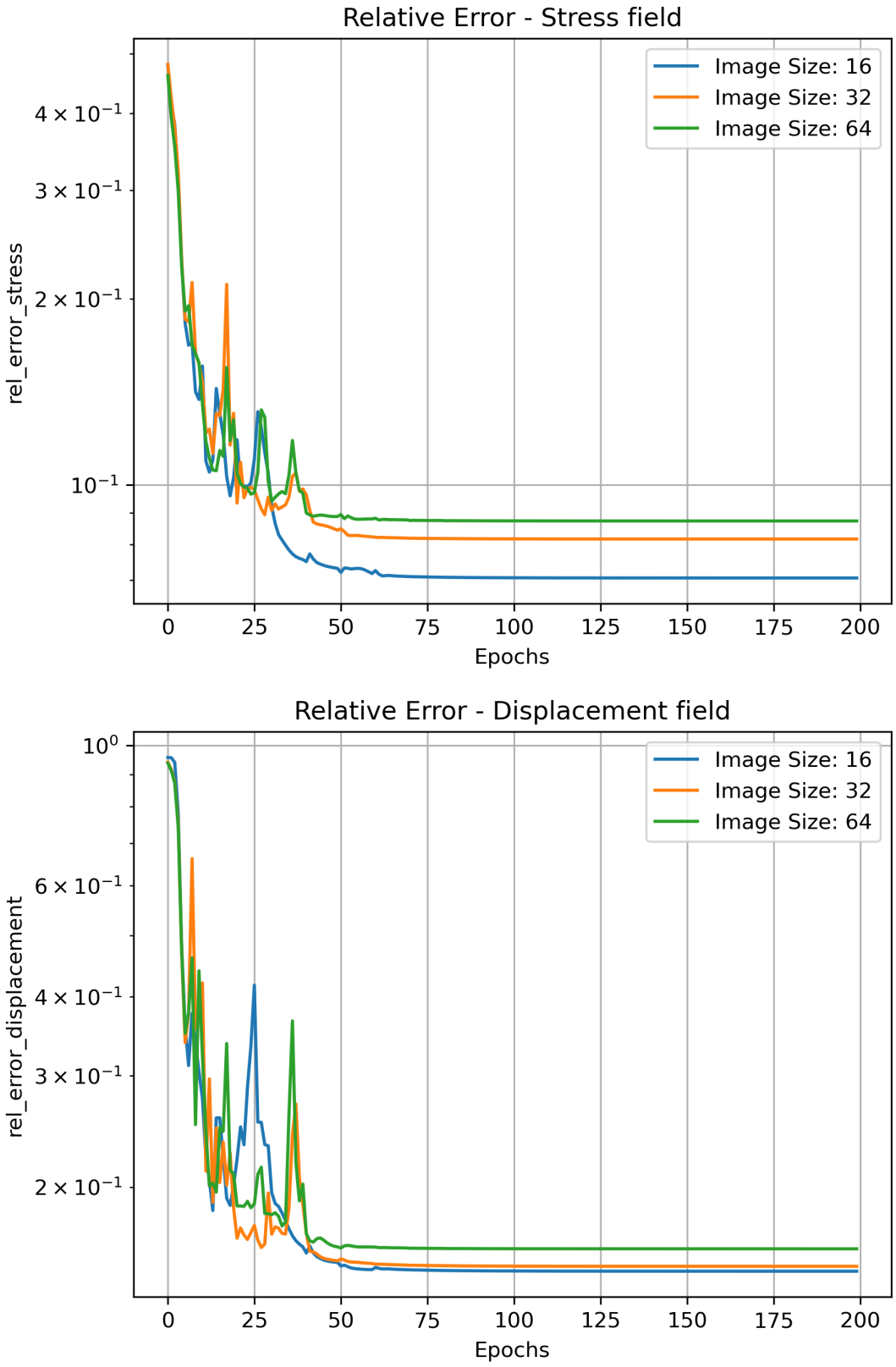

Figure 4.4: FNO: Relative error comparison for different image sizes. The top plot shows the relative errors for the stress field (σ), while the bottom plot illustrates the relative errors for the displacement field ($\mathbf{u}$).

4.3 Investigation of the Problem Setup

When evaluating the performance of a PDE solver on heterogeneous materials, the primary focus is often on material properties, such as contrast ratio, shape, or microstructure. However, the results are also significantly influenced by other parameters of the PDE, particularly those related to the mechanical context in which the material operates, such as source terms and boundary conditions. Examining the performance across multiple variations of these parameters enhances the reliability of the predictions and provides a more comprehensive assessment of the two methods under analysis.

Figure 4.5 illustrates the general setup of the three configurations considered, while Table 4.3 details the specific definitions of the functions underlying the source term and boundary conditions.

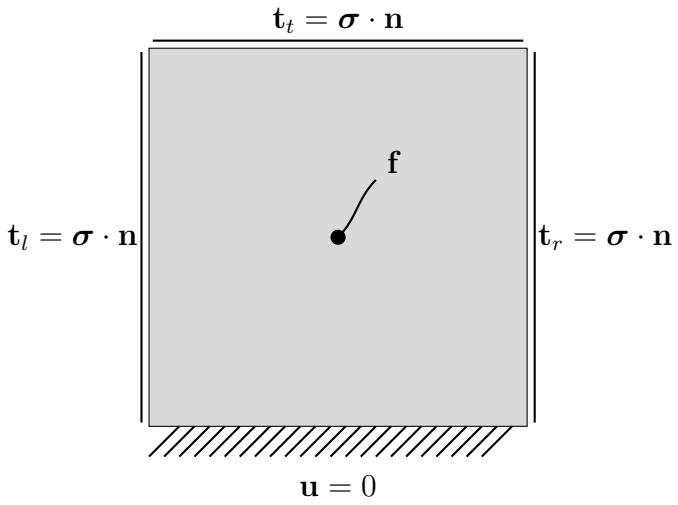

Figure 4.5: Illustration of the General Problem Setup. On the bottom boundary the material is fixed ($\mathbf{u} = 0$). On the other three boundaries traction forces act on the material.

PINN

Figure 4.6 reaffirms the increased complexity introduced by a higher contrast ratio. As expected, the configuration with $CR = 1$ generally performs better than those with higher contrast ratios. Interestingly, the purple plot ($Problem = p2, CR = 1$) initially performs worse during the first few hundred epochs but exhibits a steeper improvement rate by epoch 1000, suggesting it may eventually outperform the other configurations. For both contrast ratios, problem "p2" consistently shows the poorest performance, indicating that the method struggles to predict solutions in scenarios involving localized forces and abrupt changes in boundary conditions. For $CR = 2$, the PINN demonstrates the best performance on problem "p1," implying that for heterogeneous materials, uniaxial tension problems are easier to solve compared to other configurations. Furthermore, for heterogeneous materials, the relative error for the displacement model is slightly smaller than for the stress model in problems "p1" and "p2." This behavior suggests that the uniformity of the source term and boundary conditions is more favorable for the displacement model.

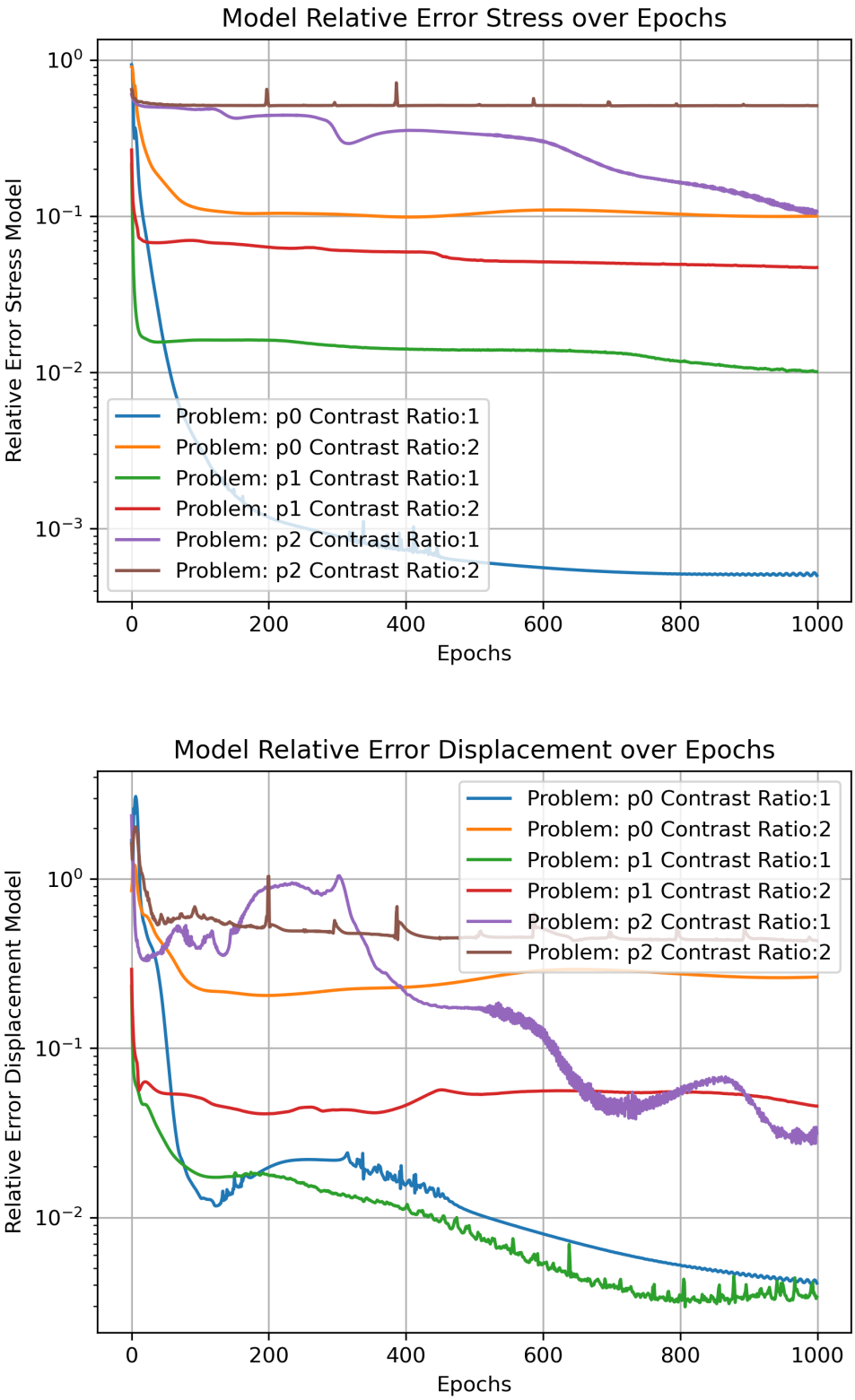

Figure 4.6: PINN: Relative error comparison for different problem setups. The top plot shows the relative errors for the stress field (σ), while the bottom plot illustrates the relative errors for the displacement field ($\mathbf{u}$).

Case	Source Term (f)	Boundary	Force Boundary Conditions (t)
$p0$	$f_x = \lambda(4\pi^2 \cos(2\pi x)\sin(\pi y) - \pi Q y^3 \cos(\pi x)) + \mu (9\pi^2 \cos(2\pi x)\sin(\pi y) - \pi Q y^3 \cos(\pi x))$ $f_y = \lambda(-3Qy^2 \sin(\pi x) + 2\pi^2 \sin(2\pi x)\cos(\pi y)) + \mu(-6Qy^2 \sin(\pi x) + 2\pi^2 \sin(2\pi x)\cos(\pi y) + \frac{\pi^2}{4} Q y^4 \sin(\pi x))$ Defined on: $[0,1] \times [0,1]$	Left ($x = 0$)	$t_x = 0, t_y = -\mu\pi(Qy^4/4 + \cos(\pi y))$
		Top ($y = 1$)	$t_x = \mu\pi(-\cos(2\pi x) + Q/4\cos(\pi x))$ $t_y = \lambda Q \sin(\pi x) + 2\mu Q \sin(\pi x)$
		Right ($x = 1$)	$t_x = 0, t_y = \mu\pi(-Qy^4/4 + \cos(\pi y))$
$p1$	$f_x = 0, f_y = 0$ Defined on: $[0,1] \times [0,1]$	Left ($x = 0$)	$t_x = 0, t_y = 0$
		Top ($y = 1$)	$t_x = 0, t_y = Q$
		Right ($x = 1$)	$t_x = 0, t_y = 0$
$p2$	$f_x = 0, f_y = 0$ Defined on: $[0,1] \times [0,1]$	Left ($x = 0$)	$t_x = -Q$ if $0.4 < y < 0.6$, $t_x = 0$ otherwise, $t_y = 0$
		Top ($y = 1$)	$t_y = Q$ if $0.4 < x < 0.6$, $t_y = 0$ otherwise, $t_x = 0$
		Right ($x = 1$)	$t_x = Q$ if $0.4 < y < 0.6$, $t_x = 0$ otherwise, $t_y = 0$

Table 4.3: Definitions of source terms and boundary forces for cases $p0$, $p1$, and $p2$.

FNO

Figure 4.7 compares the performance of the FNO model across different problem configurations for a fixed contrast ratio of $CR = 2$. The FNO demonstrates the best performance for problem "p1," achieving an impressive relative error of approximately 2% for both the stress and displacement fields. For problem "p2," the FNO also performs well, with relative errors of 2.5% for stress and 6% for displacement. In contrast, problem "p0" shows the poorest performance among the three, but the model still achieves satisfactory results overall.

When compared to the PINN model at $CR = 2$, the FNO consistently outperforms it across all three configurations, establishing itself as the superior PDE solver for heterogeneous materials. Notably, while the PINN struggles significantly with problem "p2," delivering unsatisfactory results, the FNO handles the problem effectively. This highlights the FNO's ability to tackle scenarios involving abrupt changes in boundary conditions, where the PINN falls short. For both models, problem "p1" achieves the lowest relative error, reinforcing that the uniaxial tension problem is the easiest to solve among the three configurations.

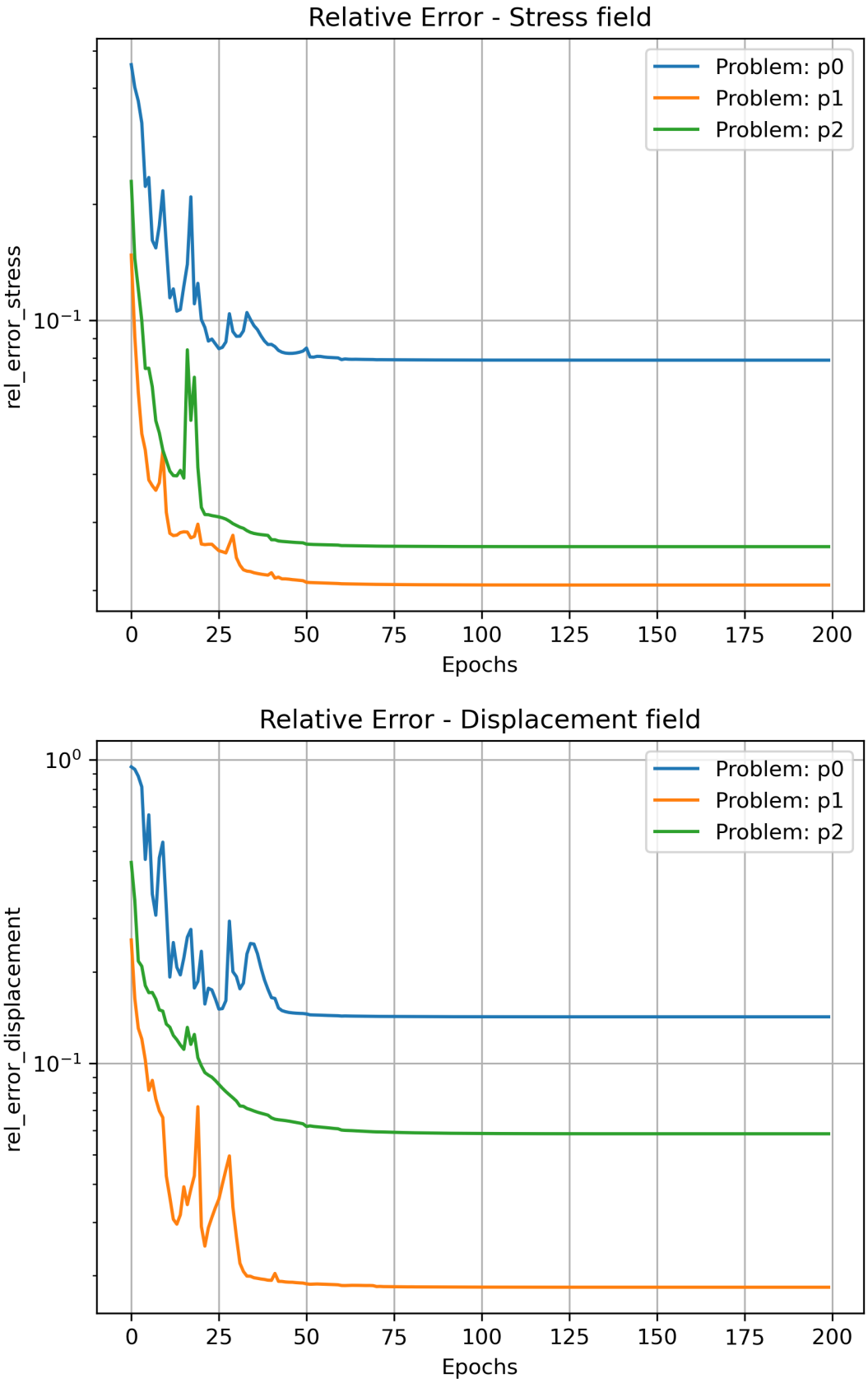

Figure 4.7: FNO: Relative error comparison for different problem setups. The top plot shows the relative errors for the stress field (σ), while the bottom plot illustrates the relative errors for the displacement field ($\mathbf{u}$).

4.4 Investigation of the FNO's output channels

In the previous investigations, an FNO model with six output channels was used to simultaneously predict $u_x, u_y, \sigma_{xx}, \sigma_{xy}, \sigma_{yx}, \sigma_{yy}$. This raises an intriguing question: does the performance of the FNO deteriorate when predicting all components of interest at once?

Figure 4.8 illustrates the evolution of the relative error for the x-component of the displacement, u_x, under two scenarios. In the first scenario ($C_{out} = 6$), the FNO predicts all six components simultaneously, while in the second ($C_{out} = 1$), the FNO is tasked with predicting only u_x.

The results in Figure 4.8 demonstrate the similarity of the two plots, with the difference in relative error being a mere 1.2%. This suggests that the FNO's performance is not significantly impacted by the number of output channels, at least for the problem of linear elasticity.

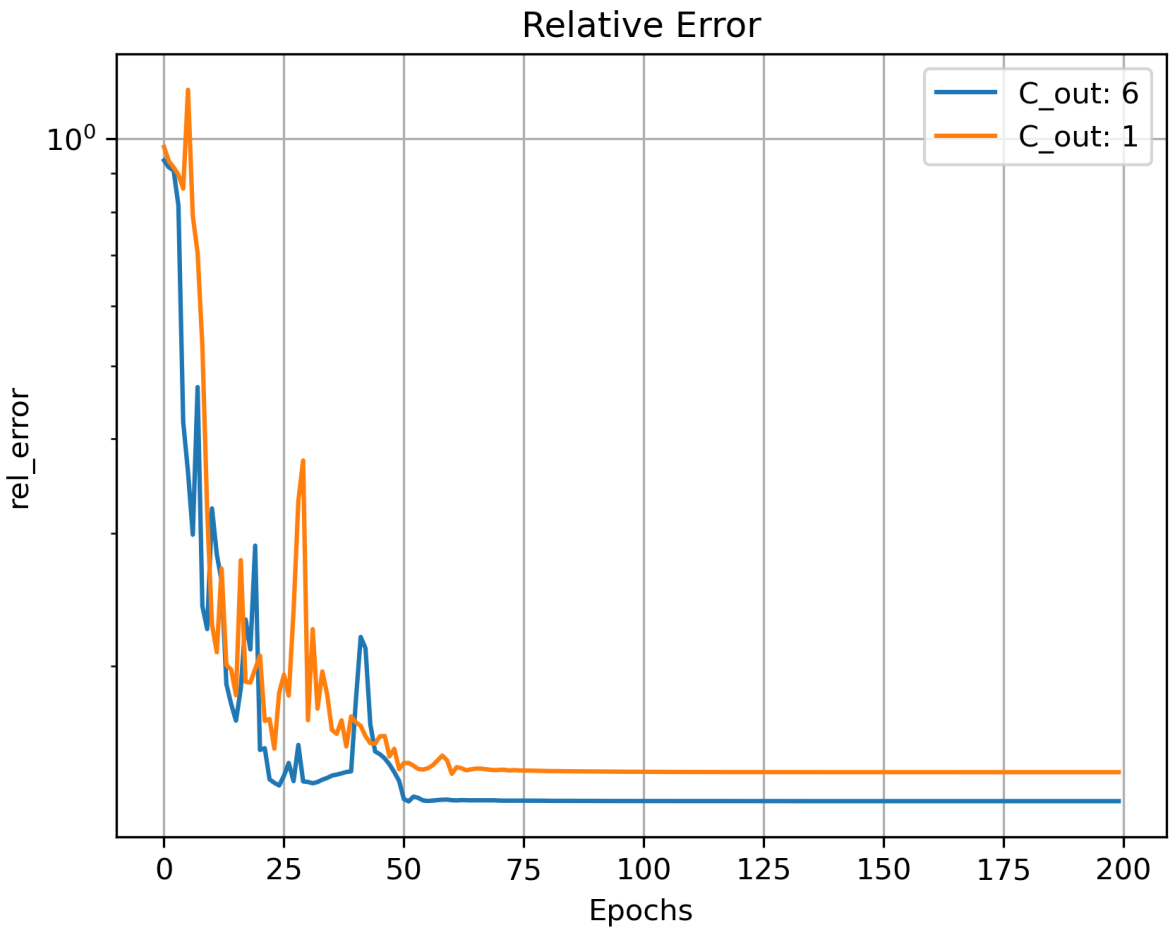

Figure 4.8: FNO: Relative error comparison for different number of output channels. In this figure only the relative error of the u_x component of the displacement field is analyzed.

Chapter 5

Summary and Conclusions

Accurately and efficiently solving the partial differential equation (PDE) of the linear elasticity problem is a critical topic in the field of solid mechanics. The solution provides insights into the mechanical behavior of materials under external excitation, with far-reaching implications in engineering, where evaluating and designing materials drives innovation. From the stress and displacement fields, further material properties can be computed, and the risk of failure under specific loads can be predicted.

This thesis investigated two deep learning methods for solving the linear elasticity PDE in heterogeneous materials: the Physics-Informed Neural Network (PINN) and the Fourier Neural Operator (FNO). A variant of the PINN was implemented, where two separate neural networks were used to predict the stress and displacement fields, deviating from the original formulation presented in [34]. The FNO was implemented as described in its original work [35], using the library provided by the authors. Hyperparameter tuning was performed for both methods to identify optimal configurations before proceeding with the experiments.

The experiments conducted analyzed the performance of these methods under various scenarios. The **investigation of Contrast Ratio** revealed a consistent decline in accuracy as the contrast ratio increased, with the most notable gap occurring between the homogeneous case ($CR = 1$) and the first heterogeneous case ($CR = 2$). The PINN variant proved unsuitable for heterogeneous materials overall, while FNO achieved acceptable results only for low contrast ratios. The **Image Size analysis** demonstrated that accuracy generally decreased with increasing image resolution for both methods. For the highest and lowest image sizes, the PINN exhibited a significant performance gap, whereas FNO showed only a marginal difference of 2%, indicating its robustness in handling higher resolutions and irregular PDE solutions. Conversely, PINNs performed better for lower image resolutions. The **Problem Setup experiment** evaluated the methods' performance across three distinct scenarios involving varying source terms and boundary conditions. Contrary to expectations, the results highlighted that the models' accuracy is significantly influenced by external mechanical factors, such as boundary conditions, rather than solely by the material's intrinsic properties. Lastly, the **FNO Output Channel investigation** demonstrated that predicting all solution components simultaneously had an insignificant impact on the model's relative error, showcasing FNO's flexibility in handling multi-output problems.

Bibliography

[1] A. E. H. Love, *A Treatise on the Mathematical Theory of Elasticity*, vol. 1, Cambridge: Cambridge University Press, 1892. [Online]. Available: https://hal.science/hal-01307751v1.

[2] P. L. Gould, *Introduction to Linear Elasticity*, 2nd ed., New York: Springer-Verlag, 1994. doi: 10.1007/978-1-4612-4296-3.

[3] L. Marin and D. Lesnic, "The Method of Fundamental Solutions for the Cauchy Problem in Two-dimensional Linear Elasticity," *Int. J. Solids Struct.*, vol. 41, no. 12, pp. 3425–3438, 2004. doi: 10.1016/j.ijsolstr.2004.02.009.

[4] Z. Cai and G. Starke, "Least-Squares Methods for Linear Elasticity," *SIAM J. Numer. Anal.*, vol. 42, no. 2, pp. 826–842, 2004. doi: 10.1137/S0036142902418357.

[5] P. Le Tallec, "Numerical Methods for Nonlinear Three-dimensional Elasticity," in *Handbook of Numerical Analysis*, vol. III, Techniques of Scientific Computing (Part 1), P. G. Ciarlet and J. L. Lions, Eds., 1st ed. Amsterdam: Elsevier, 1994, pp. 465–607.

[6] D. T. Fullwood, S. R. Niezgoda, B. L. Adams, and S. R. Kalidindi, "Microstructure sensitive design for performance optimization," *Progress in Materials Science*, vol. 55, pp. 477–562, 2010. doi: 10.1016/j.pmatsci.2009.08.002.

[7] J. Han and W. E., "Deep Learning Approximation for Stochastic Control Problems," *arXiv*, Nov. 2016. [Online]. Available: https://arxiv.org/abs/1611.07422.

[8] W. E., J. Han, and A. Jentzen, "Deep Learning-Based Numerical Methods for High-Dimensional Parabolic Partial Differential Equations and Backward Stochastic Differential Equations," *Commun. Math. Stat.*, vol. 5, no. 4, pp. 349–380, 2017. doi: 10.1007/s40304-017-0117-6.

[9] J. Sirignano and K. Spiliopoulos, "DGM: A Deep Learning Algorithm for Solving Partial Differential Equations," *J. Comput. Phys.*, vol. 375, pp. 1339–1364, 2018. doi: 10.1016/j.jcp.2018.08.029.

[10] W. E. and B. Yu, "The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems," *Commun. Math. Stat.*, vol. 6, no. 1, pp. 1–12, 2018. doi: 10.1007/s40304-018-0127-z.

[11] J. Han, A. Jentzen, and W. E., "Solving High-Dimensional Partial Differential Equations Using Deep Learning," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 115, no. 34, pp. 8505–8510, 2018. doi: 10.1073/pnas.1718942115.

[12] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis, "Physics-Informed Neural Networks for Heat Transfer Problems," *J. Heat Transf.*, vol. 143, no. 6, 2021. doi: 10.1115/1.4050542.

[13] A. M. Tartakovsky, C. Ortiz Marrero, P. Perdikaris, G. D. Tartakovsky, and D. Barajas-Solano, "Physics-Informed Deep Neural Networks for Learning Parameters and Constitutive Relationships in Subsurface Flow Problems," *Water Resour. Res.*, vol. 56, 2020. doi: 10.1029/2019WR026731.

[14] J. D. Smith, K. Azizzadenesheli, and Z. E. Ross, "EikoNet: Solving the Eikonal Equation With Deep Neural Networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 12, pp. 10685–10697, Dec. 2021. doi: 10.1109/TGRS.2020.3039165.

[15] E. Kharazmi, Z. Zhang, and G. E. Karniadakis, "VPINNs: Variational Physics-Informed Neural Networks for Solving Partial Differential Equations," *J. Comput. Phys.*, vol. 407, pp. 109227, 2020. doi: 10.1016/j.jcp.2019.109227.

[16] A. A. Ramabathiran and P. Ramachandran, "SPINN: Sparse, Physics-based, and Partially Interpretable Neural Networks for PDEs," *J. Comput. Phys.*, vol. 445, pp. 110600, 2021. doi: 10.1016/j.jcp.2021.110600.

[17] X. Meng, Z. Li, D. Zhang, and G. E. Karniadakis, "PPINN: Parareal Physics-Informed Neural Network for Time-Dependent PDEs," *Comput. Methods Appl. Mech. Engrg.*, vol. 370, pp. 113250, 2020. doi: 10.1016/j.cma.2020.113250.

[18] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, "Physics-Informed Neural Networks (PINNs) for Fluid Mechanics: A Review," *Acta Mech. Sin.*, vol. 37, no. 12, pp. 1727–1738, 2021. doi: 10.1007/s10409-021-01148-1.

[19] S. Cuomo, V. Schiano Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific Machine Learning Through Physics-Informed Neural Networks: Where we are and What's Next," *J. Sci. Comput.*, vol. 92, pp. 88, 2022. doi: 10.1007/s10915-022-01939-z.

[20] M. Vahab, E. Haghighat, M. Khaleghi, and N. Khalili, "A Physics-Informed Neural Network Approach to Solution and Identification of Biharmonic Equations of Elasticity," *J. Eng. Mech.*, vol. 148, no. 2, p. 04021154, 2022. doi: 10.1061/(ASCE)EM.1943-7889.0002062.

[21] E. Zhang, M. Dao, G. E. Karniadakis, and S. Suresh, "Analyses of Internal Structures and Defects in Materials Using Physics-Informed Neural Networks," *Sci. Adv.*, vol. 8, no. 7, p. eabk0644, 2022. doi: 10.1126/sciadv.abk0644.

[22] E. Samaniego, C. Anitescu, S. Goswami, V. M. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, and T. Rabczuk, "An Energy Approach to the Solution of Partial Differential Equations in Computational Mechanics via Machine Learning: Concepts, Implementation and Applications," *Comput. Methods Appl. Mech. Engrg.*, vol. 362, p. 112790, 2020. doi: 10.1016/j.cma.2019.112790.

[23] E. Zhang, M. Yin, and G. E. Karniadakis, "Physics-Informed Neural Networks for Nonhomogeneous Material Identification in Elasticity Imaging," Preprint, 2020. [Online]. Available: https://arxiv.org/abs/2009.04525.

[24] S. Rezaei, A. Harandi, A. Moeineddin, B.-X. Xu, and S. Reese, "A Mixed Formulation for Physics-Informed Neural Networks as a Potential Solver for Engineering Problems in Heterogeneous Domains: Comparison with Finite Element Method," *Comput. Methods Appl. Mech. Engrg.*, vol. 401, p. 115616, 2022. doi: 10.1016/j.cma.2022.115616.

[25] Z. Li, W. Peng, Z. Yuan, and J. Wang, "Fourier Neural Operator Approach to Large Eddy Simulation of Three-Dimensional Turbulence," *Theor. Appl. Mech. Lett.*, vol. 12, p. 100389, 2022. doi: 10.1016/j.taml.2022.100389.

[26] J. Pathak, S. Subramanian, P. Harrington, S. Raja, A. Chattopadhyay, M. Mardani, T. Kurth, D. Hall, Z. Li, K. Azizzadenesheli, P. Hassanzadeh, K. Kashinath, and A. Anandkumar, "FourCastNet: A Global Data-Driven High-Resolution Weather Model Using Adaptive Fourier Neural Operators," Preprint, 2022. [Online]. Available: https://arxiv.org/abs/2202.11214.

[27] T. Kurth, S. Subramanian, P. Harrington, J. Pathak, M. Mardani, D. Hall, A. Miele, K. Kashinath, and A. Anandkumar, "FourCastNet: Accelerating Global High-Resolution Weather Forecasting Using Adaptive Fourier Neural Operators," in *Proc. Platform for Advanced Scientific Computing Conf.*, Davos, Switzerland, 2023, pp. 1–11. doi: 10.1145/3592979.3593412.

[28] G. Wen, Z. Li, K. Azizzadenesheli, A. Anandkumar, and S. M. Benson, "U-FNO: An Enhanced Fourier Neural Operator-Based Deep-Learning Model for Multiphase Flow," *Adv. Water Resour.*, vol. 163, p. 104180, 2022. doi: 10.1016/j.advwatres.2022.104180.

[29] H. You, Q. Zhang, C. J. Ross, C.-H. Lee, and Y. Yu, "Learning Deep Implicit Fourier Neural Operators (IFNOs) With Applications to Heterogeneous Material Modeling," *Comput. Methods Appl. Mech. Engrg.*, vol. 398, p. 115296, 2022. doi: 10.1016/j.cma.2022.115296.

[30] R. N. Bracewell, *The Fourier Transform and Its Applications*. New York: McGraw-Hill, 1999.

[31] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*. San Diego: California Technical Publishing, 1997.

[32] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. doi: 10.1038/nature14539.

[33] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.

[34] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

[35] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier Neural Operator for Parametric Partial Differential Equations," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021. [Online]. Available: https://arxiv.org/abs/2010.08895.

[36] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Neural Operator: Graph Kernel Network for Partial Differential Equations," Preprint, 2020. [Online]. Available: https://arxiv.org/abs/2003.03485.

[37] C. C. V. Wilson, *Development and Comparison of Several Deep Learning Models to Capture Structure-Property Linkages in High-Contrast Two-Phase 2D Composites*. B.Sc. Thesis, Technische Universität München, 2022.

[38] T. Duka, *Learning the Structure-Property Linkage Between Two-Dimensional, Two-Phase Microstructures and the Effective Stiffness Tensor for Medium Contrast Ratios with the Use of Neural Networks*. Research paper, Technische Universität München, 2020.

[39] P.-S. Koutsourelakis, *Structural Mechanics Modeling*, Lecture Notes. Technische Universität München, 2023.

[40] P. Fernandez-Zelaia, Y. C. Yabansu, and S. R. Kalidindi, "A Comparative Study of the Efficacy of Local/Global and Parametric/Nonparametric Machine Learning Methods for Establishing Structure–Property Linkages in High-Contrast 3D Elastic Composites," *Integr. Mater. Manuf. Innov.*, vol. 8, pp. 217–228, 2019. doi: 10.1007/s40192-019-00129-4.

[41] L. C. Evans, *Partial Differential Equations*, vol. 19. Providence, RI: American Mathematical Society, 1994.

[42] FEniCS Project, "FEniCS Tutorial: Finite Element Method." [Online]. Available: https://fenicsproject.org/pub/tutorial/html/._ftut1008.html.

[43] H. P. Langtangen and A. Logg, *Solving PDEs in Python: The FEniCS Tutorial I*, 1st ed. Cham, Switzerland: Springer, 2016.

[44] Jinghu, *PINN-Elasticity*, 2024. [Online]. Available: https://github.com/JINGHU-UT/PINN-elasticity.

[45] T. J. Sullivan, "A Brief Introduction to Weak Formulations of PDEs and the Finite Element Method," Technical Report, 2014. [Online]. Available: https://warwick.ac.uk/fac/sci/hetsys/studentinformation/induction/mathsinduction/pde/pde.pdf.

[46] Neural Operator, *NeuralOperator/NeuralOperator*, GitHub repository. [Online]. Available: https://github.com/neuraloperator/neuraloperator.