On the Interplay between Deep Learning, Partial Differential Equations and Inverse Problems

Derick Nganyu Tanyu

Zentrum für Industriemathematik

Universität Bremen

A thesis submitted for the award of the degree of

Doktor der Ingenieurwissenschaften
(Dr. -Ing.)

Supervisor & First Reviewer: Prof. Dr. Dr. h.c. Peter Maaß
(University of Bremen)

Second Reviewer: Prof. Dr. Axel Klar
(RPTU Kaiserslautern-Landau)

Date of Defense: 04 December 2023

Acknowledgements

I express my deepest gratitude to my unwavering support system, my phenomenal wife Nancy, and my fantastic son Isaiah-Zane. Your steadfast encouragement and support throughout this journey have been the driving force behind my perseverance.

To my parents and siblings, Fideline, Kevin, Victorine, and Victor, your support and countless small gestures of love and care have inspired me.

My friends in both my home country and Germany have been a wellspring of motivation. Our discussions have been a source of joy and inspiration, lifting my spirits during challenging times.

To my advisors, Peter and Andreas, your guidance and mentorship have been pivotal in my academic journey. Peter, I am particularly thankful for your unwavering belief in me and encouragement to strike a healthy work-life balance. Your dedication sets a remarkable example for all academicians. I would also like to sincerely thank our group's indispensable secretaries, Judith and Dörte, for their constant and proactive support.

I also appreciate Georg for the constant encouragement and feedback from our days at AIMS Ghana.

I wish to acknowledge my secondary school teachers, Mr. Sanji E., Mr. Agock A., and Mr. Ntoghio P. You made mathematics beautiful and initiated this path. Your guidance was invaluable, and I am truly grateful.

I am fortunate to have worked alongside exceptional colleagues at ZeTeM, including Sören, whose enthusiasm for brainstorming always sparks new ideas; Jianfeng, whose keen attention to detail always enhanced our work; and Janek, Jean, Nick, Tom, Max, Vladimir, David, Daniel and countless others who have contributed to a collaborative and vibrant research environment.

Thank you all for being part of this incredible journey

Abstract

The intersection of deep learning and mathematics has led to profound developments in both fields. On one front, researchers have sought a deeper understanding of deep learning's mathematical underpinnings, aiming to enhance its robustness. Simultaneously, deep learning has been harnessed to address mathematical challenges, paving the way for scientific machine learning. This interdisciplinary synergy has revolutionised scientific computing, particularly in the context of partial differential equations (PDEs). Innovative neural network architectures have emerged, tailor-made for solving specific classes of PDEs, capitalising on inherent PDE properties. These advancements have significantly impacted mathematical modelling, where parametric PDEs are pivotal in representing natural and physical processes in science and engineering.

This thesis looks at these specialised neural network methods and extends them for parametric studies and the solution of related inverse problems. The relevance of these approaches is showcased across various industrial applications; namely, in a continuum mechanics problem encountered in the automotive industry during vehicle development. Building upon this foundation, the research extends to more intricate PDEs encountered in scientific and engineering domains, including the Navier-Stokes equation, Helmholtz equation, advection equation, and Solid mechanics equation. Methodologies are rigorously examined, comparing neural operator-based techniques with classical finite element solvers and Tikhonov functional-based approaches. Extensive numerical experiments are conducted under varying noise levels, providing insights into the trade-offs and applicability of different methods for diverse PDE-based challenges.

Another aspect of this thesis is exploring Electrical Impedance Tomography (EIT), a powerful imaging technique with diverse applications, primarily focusing on solving its challenging (PDE-based) inverse problem. A comprehensive examination of deep learning-based and analytic-based strategies is conducted, emphasising their strengths and limitations. Novel variable conductivity scenarios are introduced to mimic real-world complexities, facilitating a nuanced assessment of the methods' robustness and adaptability.

Résumé

L'intersection de l'apprentissage profond et des mathématiques a conduit à de profonds développements dans les deux domaines. D'une part, les chercheurs ont cherché à mieux comprendre les fondements mathématiques de l'apprentissage profond, afin d'en améliorer la robustesse. Simultanément, l'apprentissage profond a été exploité pour relever des défis mathématiques, ouvrant la voie à l'apprentissage automatique scientifique. Cette synergie interdisciplinaire a révolutionné l'informatique scientifique, en particulier dans le contexte des équations différentielles partielles (EDP). Des architectures innovantes de réseaux neuronaux ont vu le jour, conçues sur mesure pour résoudre des classes spécifiques d'EDP, en capitalisant sur les propriétés inhérentes à l'EDP. Ces avancées ont eu un impact significatif sur la modélisation mathématique, où les EDP paramétriques jouent un rôle central dans la représentation des processus naturels et physiques en science et en ingénierie.

Cette thèse examine ces méthodes spécialisées de réseaux neuronaux et les étend aux études paramétriques et à la résolution de problèmes inverses connexes. La pertinence de ces approches est démontrée dans diverses applications industrielles, notamment dans un problème de mécanique des milieux continus rencontré dans l'industrie automobile au cours du développement des véhicules. Sur cette base, la recherche s'étend à des EDP plus complexes rencontrées dans les domaines scientifiques et techniques, notamment l'équation de Navier-Stokes, l'équation de Helmholtz, l'équation d'advection et l'équation de la mécanique des solides. Les méthodologies sont rigoureusement examinées, comparant les techniques basées sur les opérateurs neuronaux aux solveurs classiques par éléments finis et aux approches basées sur les fonctions de Tikhonov. Des expériences numériques approfondies sont menées sous différents niveaux de bruit, fournissant un aperçu des compromis et de l'applicabilité des différentes méthodes pour divers défis basés sur les EDP.

Un autre aspect de cette thèse est l'exploration de la tomographie d'impédance électrique (EIT), une technique d'imagerie puissante avec diverses applications, en se concentrant principalement sur la résolution de son problème inverse (basé sur les EDP). Un examen complet des stratégies basées sur l'apprentissage profond et sur l'analyse est effectué, soulignant leurs forces et leurs limites. De nouveaux scénarios de conductivité variable sont introduits pour imiter les complexités du monde réel, facilitant une évaluation nuancée de la robustesse et de l'adaptabilité des méthodes.

Zusammenfassung

Die Überschneidung von Deep Learning und Mathematik hat zu tiefgreifenden Entwicklungen in beiden Bereichen geführt. Einerseits haben sich die Forscher um ein tieferes Verständnis der mathematischen Grundlagen des Deep Learning bemüht, um dessen Robustheit zu verbessern. Gleichzeitig wurde das Deep Learning genutzt, um mathematische Herausforderungen zu lösen und den Weg für wissenschaftliches maschinelles Lernen zu ebnen. Diese interdisziplinäre Synergie hat das wissenschaftliche Rechnen revolutioniert, insbesondere im Zusammenhang mit partiellen Differentialgleichungen (PDEs). Es sind innovative neuronale Netzwerkarchitekturen entstanden, die für die Lösung spezifischer Klassen von PDEs maßgeschneidert sind und die inhärenten Eigenschaften von PDEs ausnutzen. Diese Fortschritte haben sich erheblich auf die mathematische Modellierung ausgewirkt, wo parametrische PDEs eine zentrale Rolle bei der Darstellung natürlicher und physikalischer Prozesse in Wissenschaft und Technik spielen.

Die vorliegende Arbeit befasst sich mit diesen spezialisierten neuronalen Netzwerkmethoden und erweitert sie für parametrische Studien und die Lösung damit verbundener inverser Probleme. Die Relevanz dieser Ansätze wird anhand verschiedener industrieller Anwendungen aufgezeigt, insbesondere an einem kontinuumsmechanischen Problem, das in der Automobilindustrie bei der Fahrzeugentwicklung auftritt. Aufbauend auf dieser Grundlage wird die Forschung auf kompliziertere PDEs ausgedehnt, die in wissenschaftlichen und technischen Bereichen anzutreffen sind, darunter die Navier-Stokes-Gleichung, die Helmholtz-Gleichung, die Advektionsgleichung und die Gleichung der Festkörpermechanik. Die Methoden werden eingehend untersucht, wobei auf neuronalen Operatoren basierende Verfahren mit klassischen Finite-Elemente-Lösern und auf Tichonov-Funktionen basierenden Ansätzen verglichen werden. Es werden umfangreiche numerische Experimente unter verschiedenen Rauschpegeln durchgeführt, die Einblicke in die Kompromisse und die Anwendbarkeit der verschiedenen Methoden für verschiedene PDE-basierte Herausforderungen geben.

Ein weiterer Aspekt dieser Arbeit ist die Untersuchung der elektrischen Impedanztomographie (EIT), einer leistungsstarken bildgebenden Technik mit vielfältigen Anwendungen, wobei der Schwerpunkt auf der Lösung des anspruchsvollen (PDE-basierten) inversen Problems liegt. Es wird eine umfassende Untersuchung von Deep Learning-basierten und analytischen Strategien durchgeführt, wobei deren Stärken und Grenzen hervorgehoben werden. Neuartige Szenarien mit variabler Leitfähigkeit werden eingeführt, um die Komplexität der realen Welt zu imitieren und eine nuancierte Bewertung der Robustheit und Anpassungsfähigkeit der Methoden zu ermöglichen.

Contents

It is the glory of God to conceal a matter; to search out a matter is the glory of kings.

— Proverbs 25:2, NIV-The Holy Bible

1 Introduction

1.1 Motivation

The buzz around artificial intelligence over the recent years is mostly due to its successes in end-user applications ranging from health care, telecommunications, speech recognition, self-driving cars, and many more areas in public life. The recent popularity has been due to large language models, which have been applied successfully in intelligent chatbots. Hardly, the mathematical aspects of it are mentioned. Truly, deep learning has driven these successes, and at the back of these are mathematical concepts. Specifically, in mathematics, one identifies deep learning in two major facets. On one face is the field of mathematics of deep learning, which seeks to understand the concepts of deep learning from a mathematical point of view with the possible aim of making them more robust, and on the other face, the field of deep learning for mathematics, where the objective is to solve problems in mathematics with deep learning. The latter has recorded some great successes in imaging science (edge and feature detection, classification, inpainting, denoising, etc.), inverse problems (computed tomography, magnetic resonance imaging, electrical impedance tomography, etc), and recently partial differential equations (PDEs).

In this work, we focus on deep learning for mathematics. Specifically, we look at the intersection of inverse problems and partial differential equations also known as pde-based inverse problems or parameter identification problems (PIP). We therefore seek to solve not only the PDE (forward) problem but also, we are mostly interested in solving the inverse problem in PDEs, using Deep learning. But why PDEs? Even better, why parameter identification problems in PDEs?

Indeed, one could make the case that partial differential equations (PDEs) are the go-to choice for modelling a wide range of issues in the natural sciences, engineering, and industry. As these models become increasingly intricate, especially in advanced industrial applications

like digital twins, there is a growing demand for highly effective solvers to tackle them. These models encompass the entire product life cycle, starting with traditional simulation and optimisation during the development stage and extending to process monitoring and control during production. Typically, these models depend on the precise calibration of crucial parameters, which could range from single numerical values to complex spatial and temporal parameter functions. The calibration procedure necessitates multiple iterations of the model and further underscores the importance of efficient solving techniques.

It's unsurprising that data-driven ideas, particularly neural network approaches, have been extensively examined in recent years. They hold the potential to overcome three significant challenges in Partial Differential Equation (PDE) simulations:

- Firstly, it's a well-known fact that no mathematical-physical model is ever entirely comprehensive. However, a sufficiently large dataset can capture even the most intricate details or challenging nonlinearities.
- Secondly, in many cases, the parameters that need to be determined do not follow arbitrary patterns but instead adhere to unknown, application-specific distributions. These distributions can be uncovered and harnessed through training data.
- Thirdly, as previously mentioned, the complexity of PDE-based modelling has reached a point where traditional and widely used methods like finite elements or finite differences demand computational times beyond reasonable limits. In contrast, neural network concepts prove to be highly efficient once trained.

Our own experiences over the past few years align with this trend. An increasing number of our industrial and engineering collaboration partners have begun to explore the application of deep learning (DL) concepts based on neural networks for solving Partial Differential Equation (PDE) problems. However, they and we have encountered a bewildering array of diverse DL approaches tailored to these tasks. These approaches range from general methods suitable for various problems to highly specialised ones tailored for individual equations.

To be more precise, we are considering second-order partial differential equations defined in a bounded domain $\Omega \subset \quad ^d$, which depend on a parameter function λ:

$$\mathcal{N}(u, \nabla u, \Delta u, u_t; \lambda) = 0 \tag{1.1}$$

$$\lambda : \Omega \to$$

In this general notation, $\mathcal{N}$ encodes the differential equation as well as boundary conditions. We always assume, that the parameter-to-state operator F, which maps a given parameter λ to the solution of the PDE, is well-posed. I.e., we assume a function space setting such that the solution u of the PDE is unique and depends continuously on the parameter λ. We will consider three related classes of problems:

1. Forward problem: solving a single PDE: given λ, compute $u = F(\lambda)$
2. Parametric studies: given many parameters, $\lambda_i, i = 1, \ldots, n$, compute the corresponding u_i
3. Parameter identification (inverse problem): given a measured u^δ or its values Pu^δ under a measurement operator P, determine a corresponding λ, e.g., solve $F(\lambda) \sim u^\delta$.

Our main target is the third class of problems, i.e., inverse problems stated as parameter identification problems for PDEs. These problems are typically non-linear and ill-posed, even if the differential equation is linear and the forward operator is well-posed. Nevertheless, we will start with the first problem and review the most common DL approaches for solving the forward problem. We then discuss their potential for parametric studies and parameter identification problems. As an underlying motivation for using DL concepts in this setting, we assume that evaluating the forward operator F by classical methods, e.g., finite difference schemes or finite elements, is computationally expensive and unsuitable for large-scale parametric studies.

There are two remarks for clarifying the scope of the present work. Foremost, parameter identification problems are inverse problems and can be attacked by well-established general regularisation schemes for operator equations. For a recent overview of such data-driven concepts for inverse problems and their regularisation properties, see [10]. However, these concepts, e.g., unrolled iteration schemes, typically involve an evaluation of the forward operator or its adjoint. This will not lead to efficient schemes in our framework, where the evaluation of F is assumed to be too costly for large-scale parametric studies. We will remark shortly on that in our section on the state of the art. Hence, in the present paper, we only consider DL concepts, where the forward operator itself, i.e., the parameter-to-state operator, or its inverse are replaced by a neural network. Secondly, a growing number of highly optimised DL concepts exist for very specific PDE problems, e.g., for coupled physics systems, molecular dynamics, or complex fluid dynamic problems. These approaches have a limited potential for transfer to other problems, and we will not address these concepts. We will rather focus on general classes of DL concepts that apply to various PDE-based problems. However, we will highlight some of those successful but specialised approaches in the state-of-the-art section. All the codes and datasets used for the numerical experiment will be available online on GitHub.

1.2 Some remarks on concepts not covered in this work

This section aims at drawing attention to research directions outside this work's scope.

Firstly, several specialised and powerful DL concepts exist for complex but specific PDEs. Research on neural networks for such specialised cases of PDEs, as well as related parametric studies and inverse problems, is exploding, and it is impossible to give an exhaustive list of

references. Hence, we only seek to highlight areas of research that have reached a certain level of maturity in terms of experimental success and mathematical rigour.

A particularly successful area for DL applications is coupled-physics systems, which lead to some of the most complex inverse problems. Typically, those inverse problems cannot be solved by standard concepts, neither in the classical analytical regularisation setting nor in data-driven frameworks. Hence, such problems, e.g., optoacoustic tomography, require specialised concepts, which typically integrate domain-specific expert knowledge or at least partially draw their motivation from analytic reconstruction formulae, see [9, 63, 132, 301]. While we are at it, we also mention successes of DL methods in other sub-fields of tomography such as electrical impedance tomography [128, 137], diffuse optical tomography [239] and computed tomography [204, 205]. Another class of papers takes numerical schemes for PDEs as a starting point for developing network architectures, see [122, 275]. In these papers, the different layers of a neural network are regarded as time stepping in a discretised scheme. In particular, this allows to specifically mimic and improve numerical schemes for parabolic equations, which, e.g., are the basis for many imaging problems based on diffusion processes. Simulation of turbulent flows also is an area, where DL concepts have shown good success. We reference some papers, which in our opinion together with the cited literature therein allow an overview of this topic [76, 233, 309]. One should also mention the spectacular results of simulating molecular dynamics [34, 172] and many other fields of applications, where DL simulations lead to groundbreaking novel insights.

We also do not cover the topic of DL for stochastic PDEs and recent results, which we regard to be more on the side of PDE theory and simulation. Hence, we do not cover recent developments as e.g., described in [40, 74, 79].

Secondly, we want to remark on general regularisation schemes for inverse problems using a data-driven approach, which can be applied to but are not particularly tailored for PDE-based parameter identification problems. Typically, these general data-driven regularisation schemes for parameter identification problems do need an evaluation of F or any form of its adjoints. For this type of scheme, we refer to [10] or to [1] for a recent paper, which embeds the learning of the network together with the parameter identification into a regularisation scheme. We would like to remark that unrolled iteration schemes, such as LISTA or unrolled primal-dual, are somewhat hybrid methods and could be included in our work. However, LISTA is predominately successful for linear forward operators A and still needs the adjoint A^* for initialising the network's input. One could extend such DL schemes, which are derived from classical regularisation theory, so that the evaluation F or its adjoint would be done by training an embedded network for every operator evaluation. However, this leads to a rather complex training task and this extension is not part of the scope of the original papers for e.g., primal-dual, NETT or DeepPrior or similar [2, 74, 207].

1.3 Published articles and contributions of the author

Each chapter within this thesis draws from articles the author and collaborators authored, as explicitly stated at the outset of each chapter. Some aspects of these articles are generally referenced rather than fully presented within this thesis. This omission is primarily because of their lack of direct relevance to the central focus of this thesis. Additionally, there are instances where the research contained within the omitted sections was not conducted by the author, resulting in its inclusion here solely for review and reference purposes. The first authorship is also underlined for each of the articles presented below.

To enhance comprehension and organisation, the contents of some articles are divided into different chapters, with each section contributing to a cohesive whole. Below each article's representation in this thesis is a brief description of the role played by the author in the publication.

- Derick Nganyu Tanyu, Jianfeng Ning, Tom Freudenberg, Nick Heilenkötter, Andreas Rademacher, Uwe Iben and Peter Maass. "Deep learning methods for partial differential equations and related parameter identification problems". In: *Inverse Problems 39.10 (Aug. 2023), p. 103001.*

  Derick Nganyu Tanyu contributed to all aspects of this work and is the corresponding author. He did the organisation, presentation of the results and two-third of the experiments. The section on theoretical results on deep learning for PDEs was the focus of another co-author. This work is the basis of chapter 3 and the numerical results presented in chapter 4. A part of this paper is also briefly showcased in chapters 1 and 2.

- Derick Nganyu Tanyu, Isabel Michel I, Andreas Rademacher, Peter Maass and Jörg Kuhnert. "Parameter identification by deep learning of a material model for granular media". In: *Manuscript submitted for publication (Jul. 2023). Preprint available at (arXiv:2307.04166).*

  Derick Nganyu Tanyu contributed to all aspects of this work. As the main author and corresponding author, he performed all the coding and writing. Chapter 5 is based on this work.

- Derick Nganyu Tanyu, Jianfeng Ning, Andreas Hauptmann, Bangti Jin and Peter Maass. "Electrical Impedance Tomography: A Fair Comparative Study on Deep Learning and Analytic-based Approaches". In: *Manuscript submitted for publication (Nov. 2023). Preprint to be made available on arXiv.*

  Derick Nganyu Tanyu contributed to all aspects of this work except for the experiments involving neural network architectures. He produced the dataset needed for this work, as well as the simulations. As the first and corresponding author, he worked equally to organise, present and discuss the results. This work forms the basis of chapter 6.

- Derick Nganyu Tanyu. "From Neural Operators to Complex Partial Differential Equations based Inverse Problems: Comparative Numerical Methods for Problem-Solving". In: *In preparation. Preprint to be made available on arXiv.*

  Derick Nganyu Tanyu contributed to all aspects of this work, as the sole author. The numerical results present in this work are the basis of chapter 4. A part of chapter 3 is also from this work.

- Terence Kibula Lukong, Derick Nganyu Tanyu, Thomas Tamo Tatietse, and Detlef Schulz. "Long Term Electricity Load Forecast Based on Machine Learning for Cameroon's Power System". In: *Energy and Environment Research 12.1 (2022): 1-45.*.

  Derick Nganyu Tanyu contributed to all aspects of this work. and performed all the coding leading to the results. This work is not included in this thesis due to its divergence from the central theme of the thesis. It is only listed here as part of the work conducted by the author during the PhD period.

1.4 Structure of the thesis

In Chapter 2 we provide some theoretical foundations and basic knowledge on which this thesis builds. We then proceed in Chapter 3, with an overview of deep learning for PDEs, which mainly focuses on solving the forward problem. We equally describe extending these methods to solve the respective inverse or parameter identification problem. Chapter 4 takes a step further, and there, we numerically demonstrate the strengths of methods described in Chapter 3. We start with basic 'academic' problems and build to more complex ones. Chapter 5, takes us to industry, specifically in the automotive industry, where we look at a parameter identification problem in continuum mechanics. This PDE is vital in the vehicle development process. In chapter 6, we take a look at the Calderón problem: a famous PDE-based inverse problem, which forms the basis of electrical impedance tomography and has great potential in medical imaging among many others.

It is not knowledge, but the act of learning, not possession but the act of getting there, which grants the greatest enjoyment.

— *Carl Friedrich Gauss*

2 Preliminaries and Foundations

2.1 Motivation

The focus of this work being the intersection of deep learning, PDEs and inverse problems, a background on the respective subjects is thus warranted. In this chapter, we aim to provide some foundations in these areas. Starting from the growing field of Deep Learning, by stating its most fundamental architectures; through Inverse problems, highlighting the two major approaches in the area, and finally to PDEs, categorising and describing how they are widely solved.

2.2 Neural Networks and Deep Learning

The recent years have seen a boost in artificial intelligence. These successes have mostly been fuelled by the advances made in Deep learning. The early groundbreaking works can be found in the well-written books [133] [26] [245]; only some basics serving as a foundation of this work are highlighted at this point.

Deep learning is commonly seen as a sub-field of machine learning, which itself is a sub-field of artificial intelligence. The specificity of deep learning is the existence of the so-called neural networks. Neural networks have as main building blocks artificial neurons, also known as perceptrons or artificial neural network units, which try to achieve intelligence (thus artificial intelligence). by mimicking the functionality of biological neurons, as illustrated in Figure 2.1. For this reason, perceptrons are designed to have the following elements.

- **Activation**: Both artificial neurons and biological neurons can be in an active or inactive state. In the biological case, this is associated with the action potentials of "firing" or "not firing", while in artificial neurons, it's looked upon as the neuron's output being either in an "on" or "off" state"

- **Inputs**: Both types of neurons receive inputs from other neurons or sources. In biological neurons, these inputs come from dendrites, while in artificial neurons, they are usually mathematical values. A common practice is to normalise or standardise the inputs of the ANN so that they are within a certain range or in a particular distribution.
- **Weights**: Artificial neurons assign weights to their inputs, which determine the importance or influence of each input on the neuron's output. A larger weight usually implies the particular input has a greater influence on the output.
- **Summation**: Both artificial and biological neurons perform a weighted sum of their inputs. In artificial neurons, it's a mathematical operation that sums the products of inputs and weights. Usually, a bias is added to the result of the sum of the products.
- **Activation Function**: Both types of neurons apply an activation function to the weighted sum of inputs. While this is a complex electrochemical process in biological neurons, in artificial neurons, it's typically a simple mathematical function like a step function, sigmoid, or rectified linear unit (ReLU) [250].
- **Output**: Both types of neurons produce an output. This output is usually the application of the activation function to the weighted sum of inputs with the bias.
- **Learning**: Artificial neurons are usually conceived to learn from data via techniques like gradient descent and backpropagation. As a result of this learning process, artificial neural networks are able to adapt and improve their performance with time.

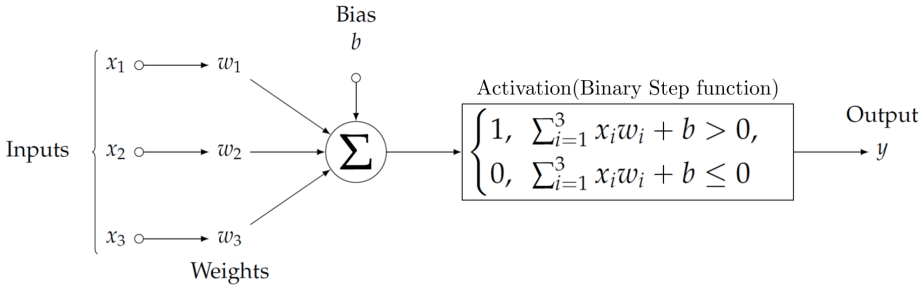


Figure 2.1: The perceptron-the building unit of artificial neural networks.

While the biological neurons inspire the functionality of the perceptrons, they remain a simplified model of the latter and, as a result, do not embody the former's complexity, adaptability as well as their functionality.

We now provide the definition of a perceptron.

Definition 2.2.1 (Perceptron or Artificial neuron)**.** *A perceptron is the function* $f : \ ^n \rightarrow$ *given by*

$$f(x_1, \ldots, x_n) = \phi\left(\sum_{i=1}^{n} x_i w_i + b\right) = \phi(\langle \boldsymbol{x}, \boldsymbol{w} \rangle + b) \tag{2.1}$$

where $\boldsymbol{x} = (x_1, \ldots, x_n) \in \ ^n$ *are the inputs,* $\boldsymbol{w} = (x_1, \ldots, x_n) \in \ ^n$ *are the weights,* $b \in$ *is the bias and* $\phi : \ \rightarrow \$ *is the non-linear activation function.*

In practice, many artificial neurons are made to work on a single input, resulting in a possibly higher dimension output. This usually constitutes a layer.

Definition 2.2.2 (Neural network layer)**.** *A neural network layer is made of* N_ℓ *neurons (usually called the width of the layer), all acting on an input* $\boldsymbol{y} \in \ ^{N_{\ell-1}}$*. It is the function* $f_\ell : \ ^{N_{\ell-1}} \rightarrow \ ^{N_\ell}$ *given by*

$$f_\ell(\boldsymbol{y}) = \phi\left(W^{(\ell)}\boldsymbol{y} + b^{(\ell)}\right), \tag{2.2}$$

where $b^{(\ell)} \in \ ^{N_\ell}$*, and* $W^{(\ell)} \in \ ^{N_{\ell-1} \times N_\ell}$*.*

The above concepts on deep learning have been broadly covered over the past years. Meanwhile, [274] discusses the backpropagation algorithm, which is at the backbone of the learning process in artificial neural networks (ANN). The main idea is the update of the ANN's weights based on a function of the network's output (called the cost or loss function) with respect to their input. For this, optimisation algorithms such as gradient descent and its variants, stochastic gradient descent and more are used. See [113] for a review on optimisation algorithms. For a general but more in-depth understanding of the learning process of ANNs, from its history, through architectures, techniques and perspectives, we refer to [201] [103]. For a comparison and analysis of some popular activation functions, as well as their impact on the ANN's performance, one could have a look at [250].

2.2.1 Neural Networks Architectures

Over the years, a good number of neural network architectures have been developed. However, in most of these famous neural network architectures, it is possible to always identify either a feedforward neural network (FFN), a convolutional neural network (CNN) or a recurrent neural network (RNN).

Feed-forward Neural networks

Artificial neural networks are referred to as being "deep" because they are made up of a good number of neural network layers. Feed-forward neural networks, or multi-layer perceptrons (MLP), are one of the earliest breakthrough in the field. The concepts presented at the start of 2.2 directly apply to them.

Definition 2.2.3 (Feed-forward Neural Networks)**.** *Consider the function* $\mathcal{A} = f_L \circ f_{L-1} \circ \ldots \circ f_2 \circ f_1 : \ ^{N_0} \rightarrow \ ^{N_L}$ *, a composition of* L *neural network layers, with each layer* f_ℓ*, having a width of* $N_\ell, \ell = 1, \ldots, L$*. The function*

$$\mathcal{A}(\boldsymbol{x}) = f_L\left(f_{L-1}\left(\ldots\left(f_1(\boldsymbol{x})\right)\right)\right) \tag{2.3}$$

is called an artificial or deep neural network. where $\boldsymbol{x} \in \ ^{N_0}$*,* N_0 *is the dimension of the input layer and is defined as in Definition 2.2.2*

Convolutional neural networks

While feed-forward neural networks perform relatively well on 'basic' tasks, they are however limited in computer vision tasks such as image classification object detection and other tasks involving grid-like data. Convolutional neural networks were a breakthrough in this aspect. The pioneer work that used CNNs is probably the LeNet-5 architecture [202], which used CNNs for the task of handwritten digit recognition. However, its breakthrough was only later with the AlexNet [296]. Since then, subsequent developments of neural network have been made.

The fundamental building block of a CNN is a convolutional layer, used for feature extraction from input data. In addition to a convolutional layer, CNNs usually have activation layers, pooling layers (used to reduce the spatial dimensions of the feature maps while extracting dominant features), padding layers as well as fully connected layers (usually at the end of the network).

Recurrent Neural Networks

It is usually necessary to capture previous information for sequential data, such as time series data. In other words, there is a need for the network to have some memory. This is achieved with residual connections to previous layers. RNNs [273, 295] have since been improved and their shortcomings such as short-term memory have been improved in Long-short term memory (LSTM) [142] and gated recurrent unit (GRU) [58] networks.

2.2.2 Principal Component Analysis

Though not a neural network architecture per se, this concept is used in many aspects of this thesis. It is thus briefly explained here to lay some foundation. For a detailed overview, we refer to [171] Principal component analysis (PCA) or proper orthogonal decomposition (POD), as called in physics, is mostly viewed in machine learning as a dimensionality reduction technique. The starting point of PCA is high-dimensional data (with many features/variables). By using the dependencies between the variables in this data, PCA, reduces the dimension to a lower dimension (lesser features). In essence, the features with less variance, are considered to be less useful, while those with high variance are preferred. Mathematically, the PCA algorithm makes use of the concepts of mean, covariance matrices, eigenvector and eigenvalues as described in the below steps.

- centering of the data, i.e. subtraction of the mean.
- calculation of the covariance matrix
- computation of the eigenvectors and corresponding eigenvalues of the covariance matrix.
- identification of the principal components, i.e. eigenvectors with the highest eigenvalues.

- transformation of the data by using the eigenvectors.

Definition 2.2.4 (Principal Component Analysis)**.** *Given n samples of a certain data $x_i \in \ ^m$, with $i = 1, \ldots, n$, we define the matrix $X := (x_i)_{i=1}^n \in \ ^{n \times m}$ with mean μ. The matrix $X_\mu := X - \mu \in \ ^{n \times m}$ has covariance matrix in $^{m \times m}$ with corresponding eigenvector matrix $V \in \ ^{m \times m}$. The matrix $V_p \in \ ^{m \times p}$ is obtained from V by selecting the p principal components. The function $\mathcal{P}_X : X \mapsto X_u * V_p$ is the PCA, where $*$ is a matrix multiplication.*

In order to reconstruct X from $Y = \mathcal{P}_X$, one applies the operation $Y * V_p^T + \mu$, where V_p^T is the transpose of the matrix V_p

2.3 Inverse Problems

Simply put, inverse problems are concerned with recovering original information/data from a transformed version, or in other words, determine the causes of a certain desired or observed effect. In this section, only the basics on the subject are discussed. [10] provides a more exhaustive covering on the subject. For a Bayesian view on the subject, see [289].

Definition 2.3.1 (Inverse problem)**.** *The task of recovering a model parameter $\lambda \in \Lambda$, the parameter space from measured data $u \in \mathcal{U}$, the solution space, where*

$$u = F(\lambda) + \eta \tag{2.4}$$

is called an inverse problem. Additionally, $F : \Lambda \to \mathcal{U}$ is the forward operator while $\eta \in \mathcal{U}$ models the noise. (In the rest of the discussion, we ignore η)

One inherent property of inverse problems encountered in science and engineering is their ill-posedness. This makes inverse problems particularly challenging, as finding numerically or analytically stable methods for solving are then difficult. Particularly, it is worth mentioning at this point that PDE-based inverse problems tend to be highly non-linear. This further introduces another challenge, which is however not discussed at this point. [154] studies some PDE-based inverse problems such as the inverse gravimetry, inverse conductivity, inverse scattering, inverse spectral as well tomography and inverse scattering problems.

Definition 2.3.2 (Well/Ill-posedness of an inverse problem)**.** *An inverse problem is **well-posed** in the Hadamard sense [123] if the below conditions are fulfilled:*

- *Existence: for each $u \in \mathcal{U}$, there exists some $\lambda \in \Lambda$ such that $F(\lambda) = u$.*
- *Uniqueness: λ as defined above, is unique. λ is therefore the solution of the inverse problem.*
- *Stability: The inverse mapping $F^{-1} : \mathcal{U} \to \Lambda$ is continuous. Or better still, the error in the solution should depend stably on noise in the data.*

*When any of the conditions is not met, the problem is deemed **ill-posed**.*

Generally, speaking, there exist workarounds on the question of existence and uniqueness. Usually, on one hand, to deal with the problem of non-existence, a solution whose image in the range is closest to the data (in a suitable distance) is usually chosen, and on another hand, to deal with the problem of non-uniqueness, a solution is with zero projection in the Null-space of F is chosen. The challenge is thus usually at the level of instability with noise. Unfortunately, for the majority of interesting but challenging inverse problems, instability has been shown to be an intrinsic property. This formed the basis of the theory of inverse problems, with the aim to develop stable schemes for the estimation of the parameter λ from the data u [10]. Methods, leading to stable approximations of an inverse problem, are called **regularisation methods**.

2.3.1 Regularisation and Inverse Problems

On the subject of regularisation, a good number of algorithms have been developed for the obtention of stable approximations of inverse problems. Most of them can be classified into four main groups [10]:

- Approximation of the analytic inverse, which aims at stabilising a closed form of the inverses F^{-1}, usually by reconstruction operators.
- Iterative methods, applying early stopping. Usually based on the gradient descent, with objective to minimise the term $\|F(\lambda) - u\|^2_{\mathcal{U}}$, where $\|\cdot\|_{\mathcal{U}}$ is the norm in the space $\mathcal{U}$. The ill-posedness of the problem usually leads to a decrease in the error up to a certain level, after which a divergence is observed, thus the need for an early stopping criterion.
- Discretisation usually controls the approximation of the forward operator but also has an effect on the inverse.
- Variational methods minimise the data misfit with an added penalty term. The cost function is, thus, of the nature.

$$\lambda_\alpha := \underset{\lambda\in\Lambda}{\operatorname{argmin}} \left(\mathcal{L}(F(\lambda), u) + R_\alpha(\lambda)\right) \tag{2.5}$$

  Popular choices of the penalty term are total variation (TV) regularisation and Ridge or Tikhonov regularisation. This term usually controls desirable features of the parameter, and its choice is therefore motivated by this.

A special case and a rather popular version of the variational method is the classical Tikhonov regularisation, where the expression in 2.5 is now.

$$\lambda_\alpha := \underset{\lambda\in\Lambda}{\operatorname{argmin}} \left(\frac{1}{2}\|F(\lambda) - u\|^2_{\mathcal{U}} + \alpha R(\lambda)\right) \tag{2.6}$$

A popular choice of the penalty term, in literature is $R(\lambda) = \frac{1}{2}\|\lambda - m_0\|^2_\Lambda$, where the point $m_0 \in (\Lambda_0, \|\cdot\|_\Lambda) \subseteq (\Lambda, \|\cdot\|_\Lambda)$, so that we now have

$$\lambda_\alpha := \underset{\lambda\in\Lambda}{\operatorname{argmin}} \left(\frac{1}{2} \|F(\lambda) - u\|^2_{\mathcal{U}} + \alpha \frac{1}{2} \|\lambda - m_0\|^2_\Lambda \right) \tag{2.7}$$

Specifically when $R(\lambda) = \frac{1}{2}\|\lambda\|^2_\Lambda$, and F is linear, with an adjoint F^*, minimising the expression in 2.6, yields

$$\lambda_\alpha = (F^* \circ F + \alpha I)^{-1} \circ F^*, \tag{2.8}$$

where I, is the identity matrix.

2.3.2 The Bayesian approach

Up to now, we have mostly discussed the case of functional analytic regularisation. However, the question of the choice of the norms $\|\cdot\|_{\mathcal{U}}$ and $\|\cdot\|_\Lambda$, and the choice of the point m_0 remain unaddressed, as they are chosen rather *arbitrarily* in the functional setting, with no further modelling assumptions. Also, the questions of certainty that the prediction made by the model lies in a certain desired regime; and that of the relative probability that the local minimisers of 2.7 determine the parameter λ. These questions are addressed by the Bayesian approach to inverse problems and makes them *richer*. This motivates its mention at this point of this work. However, Bayesian/probabilistic approaches are not used in the problems studied in this work but show a promising direction. We refer the reader to [289] for a more exhaustive discussion on the subject. Worth mentioning, however, is the fact that the Bayesian approach seeks to find a probability measure μ^u on Λ which contains information on the states of λ given the measurement u. Also, we highlight the similarity of the minimisation problem in 2.7 to the probability measure of the posterior μ^u, with density function $\pi^u(\lambda)$, given by

$$\pi^u(\lambda) \propto \exp\left(-\frac{1}{2}\|F(\lambda) - u\|^2_{\mathcal{U}} - \frac{1}{2}\|\lambda - m_0\|^2_\Lambda \right) \tag{2.9}$$

for the specific case where Λ and $\mathcal{U}$ are finite-dimensional, with additive Gaussian noise and a Gaussian prior measure μ_0 (with density function π_0). The density $\pi^u(\lambda)$ is thus larger at the minimisers of 2.7. Additionally, Baye's formula establishes a relationship of direct proportionality between π^u and, π_0 as demonstrated in [289].

In order to obtain information from the expression in 2.9 in high-dimensional settings, the procedure is usually either through *variational* methods where a maximum a posteriori (MAP) estimator, which maximises π^u, is found or through *sampling* methods–such as Markov Chain Monte Carlo (MCMC)–where set of points distributed according to π^u are generated. In problems where the prior $\eta \sim \mathcal{N}(0, \Sigma_\eta)$ and $\mu_0 = \mathcal{N}(m_0, \Sigma_{\mu_0})$, 2.9 then becomes

$$\begin{aligned} \pi^u(\lambda) &\propto \exp\left(-\frac{1}{2}\left|\Sigma_\eta^{-1/2}(F(\lambda) - u)\right|^2 - \frac{1}{2}\left|\Sigma_{\mu_0}^{-1/2}(\lambda - m_0)\right|^2 \right) \\ &= \exp\left(-\frac{1}{2}|(F(\lambda) - u)|^2_{\Sigma_\eta} - \frac{1}{2}|(\lambda - m_0)|^2_{\Sigma_{\mu_0}} \right), \end{aligned}$$

with MAP estimator

$$\lambda^* = \underset{\lambda \in \Lambda}{\operatorname{argmin}} \left(\frac{1}{2} |(F(\lambda) - u)|^2_{\Sigma_\eta} + \frac{1}{2} |(\lambda - m_0)|^2_{\Sigma_{\mu_0}} \right),$$

which is similar to 2.7, but the choice of the norms and m_0 are clear, thus answering the question earlier highlighted.

2.4 Partial Differential Equations

As earlier mentioned, partial differential equations (PDEs) are an essential tool for modelling natural phenomena. The focus on this section is a discussion on the major (and basic) aspects of PDEs as well as numerical methods for solving PDEs. For a more complete introduction on the subject of PDES, one could refer to [83] and [287]. An expression of a second-order PDE, which we mostly focus on in this work, has already been introduced in Equation 1.1. Generally speaking, a second-order PDE has the highest derivative of order two in its expression.

2.4.1 Categorising PDEs

Partial Differential Equations (PDEs) can be categorised into several major types based on their characteristics and properties. As a rule of thumb, categorising PDEs helps in understanding the nature of the PDEs and usually serves as a guide in the choice of appropriate solution techniques and mathematical methods for solving them. Some of the major categorisations of PDEs include:

- **Elliptic** PDEs have smooth and continuous solutions. They are often used to describe steady-state problems where the solution does not depend on time. Examples include the Laplace equation and the Poisson
- **Parabolic** PDEs describe problems that evolve over time, typically involving a diffusion or heat conduction process. They have one independent variable (e.g., time) and are second-order in the spatial variables. Examples include the heat equation and the diffusion equation.
- **Hyperbolic** PDEs are used to model wave-like phenomena and problems that involve information propagation. They are characterised by two independent variables (space and time) and are second-order in both. Examples include the wave equation and the transport equation.
- **Linear** PDEs: are characterised by the linear appearance of the unknown function and its derivatives. They thus have well-understood properties and are thus generally easier to solve.
- **Nonlinear** PDEs are the opposite of linear PDEs and are often more challenging to solve

2.4.2 Classical methods for Solving PDEs numerically

The finite difference (FDM), finite element (FEM), boundary element (BEM), finite volume (FVM) and particle methods are the most established concepts for numeric solutions of PDEs. These methods are well-researched and reliable, and are generally employed whenever possible within the given restrictions of data and computer power.

The finite difference method (FDM) is based on the replacement of the occurring derivatives by difference quotients. In this process, the function values of the solution are approximated at individual discrete grid points. The result is a so-called grid function. We refer to [288] for a more detailed description. The great advantage of the FDM is its straightforward realisation. However, its extension to more complex problems is limited and rather unrealistic smoothness assumptions are necessary for its analysis.

The finite element method (FEM) overcomes the mentioned disadvantages of FDM methods to a large extent. In contrast to the FDM, it is based on the weak formulation of the PDE under consideration and searches for the discrete solution over a finite-dimensional subspace of the underlying function space, see [36] for more details. Especially in connection with adaptive and domain decomposition methods, it shows its full potential. An interesting variant is discontinuous Galerkin (dG) methods, in which the smoothness assumptions for the discrete solution, which stems from the weak formulation of the PDE, are only enforced by penalty terms.

The boundary element method (BEM) follows a different idea. In this concept, the PDE is transformed into a boundary integral equation, which, however, is only possible in special cases. This reduces the dimension of the problem. The boundary integral equation is then discretised by means of FEM, for example. However, the resulting system of equations is dense, and its numerical solution is often tricky and numerically very elaborate. Nevertheless, the BEM offers great advantages for exterior space problems in which an unbounded area is considered. We refer, for instance, to [120] for more information.

The Finite Volume Method (FVM) brings together ideas from the FEM and the FDM and combines them with arbitrary control volumes, see [206]. The FVM is particularly well suited for the discretisation of hyperbolic equations and convection-dominated problems. However, its theoretical analysis is not yet that advanced.

Another approach are particle methods or discrete element methods (DEM), see [46], which are based on a completely different approach. Here, individual particles and their interaction with each other are considered. By using different approaches to describe the interaction, different materials and processes can be simulated. They thus achieve very good accuracy in many applications, e.g., in the simulation of sand, which is difficult to simulate with the other methods.

The mentioned concepts are well-established for solving PDE-based forward problems and all of them have been adapted to inverse problems, in particular in the complex tasks of optimal control or numerical parameter identification. They lead to accurate results

in this context. However, two main difficulties have to be mentioned: Firstly, and as mentioned already in the previous section, solving these inverse problems requires solving the forward problem multiple times, which leads to unacceptable overall computing times at least for large problems. Furthermore, the optimisation algorithms for solving the above-mentioned parameter identification tasks require the first and, if possible, also the second derivative with respect to the unknown parameters. At first, it is often unclear whether the mentioned derivatives exist at all. In addition, their calculation is algorithmically very complex and time-consuming, especially for time-dependent problems. We refer to [303] for a detailed discussion.

My work always tried to unite the true with the beautiful, but when I had to choose one or the other, I usually chose the beautiful.

— *Hermann Weyl*

3 Deep Learning for Partial Differential Equations 1: Concepts and State of Art

This chapter reviews the literature and provides the start of art of DL methods for solving PDEs. They are then extended for the purpose of parametric studies and inverse problems. It is based on the following article:

> Derick Nganyu Tanyu, Jianfeng Ning, Tom Freudenberg, Nick Heilenkötter, Andreas Rademacher, Uwe Iben and Peter Maass. "Deep learning methods for partial differential equations and related parameter identification problems". In: *Inverse Problems 39.10 (Aug. 2023), p. 103001.*

3.1 Motivation

The main purpose of this section is to list the DL methods considered in this present survey. In the subsequent sections, we will then analyse their potential for parametric studies and inverse problems.

As already mentioned, mesh-based methods for solving PDEs such as finite element methods, finite difference methods, and finite volume methods are the dominant techniques for obtaining numerical solutions of PDEs. When implementing these methods, the computational domain of interest should be first discretised using a set of mesh points and the solution is then approximated at the chosen grid points. The solutions are usually obtained from a finite linear space by solving a linear or non-linear system of equations. The classical methods are very stable and efficient for low-dimensional problems and regular geometries. They are well understood in terms of existence, stability and error

estimate, and we can usually achieve the desired accuracy by increasing the resolution of the discretisation. In addition to the more general remarks above, the classical methods have several additional drawbacks and limitations. Firstly, the curse of dimensionality has limited the use of mesh-based methods, since the number of discretisation points will increase exponentially with the dimension, and there is no straightforward way to discretise irregular domains in high-dimensional spaces. Secondly, traditional methods are designed to model one specific instance of the PDE system, not the parameter-to-state operator. Thus, for any given new instance of the PDE system, the problem has to be solved again. Thirdly, the numerical solution is only computed at the mesh points and evaluation of the solution at any other point requires interpolation or some other reconstruction method. Fourthly, classical methods require the knowledge of the underlying analytic system of differential equations, but sometimes the exact physics is unknown. Lastly, solving inverse problems by classical problems is often prohibitively expensive and requires complex formulations, repeated computation of forward problems, new algorithms and elaborate computer coding.

Deep learning methods have shown great power in addressing the above difficulties. Since there are many different deep learning methods for PDEs, and each method has its particular properties, we will discuss them one by one. We should add, that the theoretical investigation of deep learning concepts for PDEs, beyond more or less direct applications of the universal approximation theorem, is still scarce. This is in contrast to the classical PDE theory, where exhaustive literature exists on the analytic properties of different types of PDEs. The theoretical characterisation of PDEs, as well as the type of theoretical background needed for proving solvability and uniqueness, differentiates between linear and non-linear elliptic, parabolic or hyperbolic equations in terms of the given boundary conditions. Analytic tools are different for these different classes; hence it is somewhat surprising that DL concepts for PDEs hardly ever use this classical classification but rather use a classification based on whether the PDE (no matter which type of PDE) is known, which type of data is given for training and which type of network architecture as well as which loss function for training is used.

We follow this data-driven classification scheme. Our first level of characterisation is based on the input and output structure of the network. A first class of networks takes x or (t, x) as input and outputs a one- or low-dimensional scalar value or vector $u_\Theta(\cdot)$. During training the weights Θ of the network are optimised such that $u_\Theta(\cdot) = \hat{u}(\cdot)$ is an approximation to the solution u of the PDE at the specified point x or (t, x), i.e., a regression problem for the function $u_\Theta \approx u$ is solved. These networks aim at a function approximation. Otherwise, one aims at an operator approximation for the parameter-to-state map and uses networks which take a parameter function, either in discretised form or as a vector of precomputed feature values, as the input and outputs the full solution function u. The solution u is again delivered either in discrete form or as a related feature vector, that can be turned into a proper function representation in a post-processing step. The general idea of these two characterisations is summarised in the following Figures 3.1-3.2:

- Function evaluation, neural network $u_\Theta(t,x) \approx u(t,x)$ based on the PINN concept, [261]

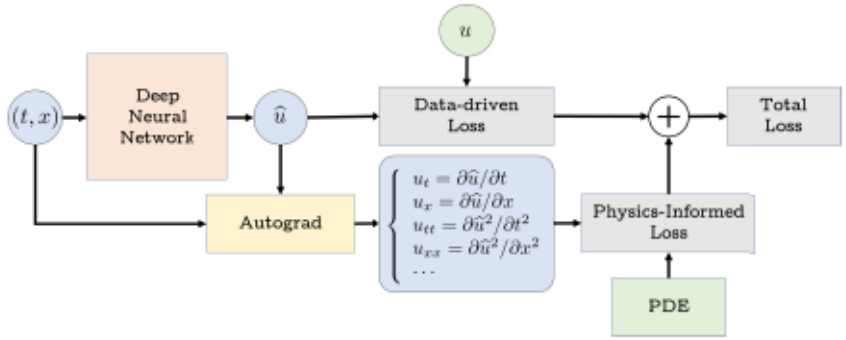


Figure 3.1: A network with a low input dimension for (t,x), which outputs a scalar or low-dimensional $\hat{u}(t,x) = u_\Theta(t,x)$. Hence, the neural network is trained to be a function approximation of the solution $\hat{u}(t,x) \approx u(t,x)$. During training, automatic differentiation by backpropagation is used to compute all necessary derivatives for checking, whether the PDE is satisfied. In addition, the loss function typically sums up over several values of (t,x) and their corresponding function values. After training, the Deep Neural Network is then used and $\hat{u}$ can be evaluated at arbitrary points (t,x).

- Operator evaluation, neural network $\Phi_\Theta(\lambda)$ following the PCANN concept [24]

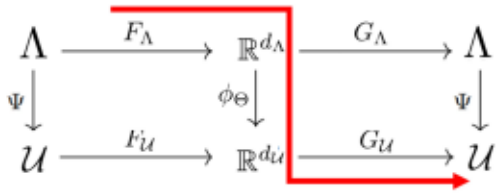


Figure 3.2: A typical setup for an operator approximation scheme. Here, an order reduction model (see [22, 57, 276] for an overview of reduction models) in combination with a neural network is employed. First, the parameter $\lambda \in \Lambda$, the parameter space, is reduced to a feature vector in $\mathbb{R}^{d_\Lambda}$ by determining its scalar products with a pre-computed reduced basis, i.e., $c_k = \langle \lambda, b_k \rangle$ (which in case of an orthonormal system is equivalent to $\lambda \sim \sum c_k b_k \in \Lambda$). Only the feature vectors are mapped by the neural network $\phi_\Theta : \mathbb{R}^{d_\Lambda} \ni c \mapsto \bar{c} \in \mathbb{R}^{d_\mathcal{U}}$. The output features $\bar{c}$ are then used to determine an expansion of the sought-for solution $\hat{u} = \sum_\ell \bar{c}_\ell \bar{b}_\ell$ in terms of a basis $\{\bar{b}_\ell\}$ in the solution space, $\mathcal{U}$. In this way, we then approximate the parameter to solution function $\Psi : \Lambda \ni \lambda \mapsto u \in \mathcal{U}$ by $\Phi_\Theta = G_\mathcal{U} \circ \phi_\Theta \circ F_\Lambda$ as indicated by the red arrow. F_Λ and $F_\mathcal{U}$ are encoder functions (PCA maps) in the parameter and solution spaces respectively, while G_Λ and $G_\mathcal{U}$ are their respective decoder functions. (see 3.5.1 and [24]).

Typical examples of DL concepts aiming for a function approximation are Deep Ritz method [215, 327], Physics-Informed Neural Network (PINN) [261], Deep Galerkin Method (DGM) [284], Weak Adversarial Networks [329], and Deep Splitting Approximation methods [19]. In all these concepts, information of the underlying physics must be used when training the neural network, i.e., the PDE must be known analytically. These concepts are usually mesh-independent and accurate, while they require knowledge of the governing laws of the PDEs, and the optimisation problem needs to be solved for every new instance. This is similar to classical methods.

Networks	PDE usage	Data usage	Neural network approximation	Discretisation parameter	Discretisation Solution	Reference
Deep Ritz	yes	no	function	no	no	[215, 327]
PINN	yes	no	function	no	no	[261]
WAN	yes	no	function	no	no	[14, 329]
PCANN	no	many	operator	free grids	free grids	[24]
FNO	no	many	operator	free grids	free grids	[211]
UFNO	no	many	operator	free grids	free grids	[316]
MWT	no	many	operator	free grids	free grids	[119]
DeepONet	no	many	operator	fixed	no	[222]
PINO	yes	if needed	operator	free grids	free grids	[214]
PI-DeepONet	yes	if needed	operator	fixed	no	[310]

Table 3.1: The list of DL concepts considered in this survey is based on a pre-selection mirroring our own interest in inverse problems. Hence, concepts for operator approximation are preferred. In their original papers, most of these concepts were designed for Level 1 problems only. Hence, we need to specify in later sections how to extend them for Level 2 or Level 3 problems. Also, we will later refine the features characterising the different methods in terms of limitations, e.g., which PDE equations are most suitable and which are not.

Examples for the second type, i.e., methods aiming at an operator approximation, include model reduction neural network (PCANN) [24], Deep Operator Network (DeepONet) [222] and its extensions [106, 310], Fourier neural operator (FNO) [211] and its physics-informed extension–Physics Informed Neural Operator (PINO) [214], Graph Kernel Network [212, 213], Bayesian deep convolutional encoder-decoder networks [334], Wavelet Neural Operators [302], Multiwavelet based operator (MWT) [119] and many more. These operator methods usually learn the neural network with some paired input-output (parameter-solution) observations with or without the knowledge of the physical system (using the physics information could sometimes alleviate the need for much data as in the case of PINO and PI-DeepONet) as highlighted in Table 3.1. Thus, the neural network only needs to be trained once, and a new parameter identification task can be directly solved by a forward pass of the network. Some of these deep learning methods still need to discretise the domain of interest, and the solutions are sought in a finite linear space [334]. A recently published paper [68] aims at comparing some of these operator approximation concepts for solving various PDEs. The primary criterion for comparison in this well-written paper is the cost-to-accuracy curve. Accordingly, the authors compare results obtained with different network sizes. However, their comparison includes the basic methods PCANN, DeepONet and FNO, which in our tests did not yield the best results. Also, the numerical test stays in a range - when compared with FEM methods - of rather large approximation errors. This matches our experience; it is hard to get to high precision for forward solvers with DL concepts. However, our focus is on inverse problems, which are not covered in the mentioned paper.

In terms of the involved computational load, the computational cost of classical methods for solving linear PDEs is mainly determined by the need to solve a large linear system. For deep learning methods, it is the training process (optimisation) of the neural network. As a general rule of thumb for low dimensional problems with regular geometries, classical methods are in general more powerful and efficient than deep learning methods. Ignoring rounding errors, the main error of the classical methods arises from the step

size of the discretisation. There exists a trade-off on the resolution: coarse grids are fast but less accurate, fine grids are more accurate but slower. On the other hand, for DL approaches one usually has to carefully choose the neural network, optimisation method and hyperparameters in consideration of the underlying differential equation. Additionally, the error of deep learning methods is usually difficult to estimate and consists of several factors. The accuracy can be characterised by dividing the total error into three main types: approximation, optimisation, and generalisation errors. More specifically, the approximation error measures the smallest difference between the neural networks and the underlying function or operator, which we aim to approximate. This error is influenced by the size and architecture of the neural networks. The optimisation error arises from the non-convexity property of the loss function and the limited number of iterations used for its minimisation. Stochastic Gradient Descent is an efficient algorithm for escaping local minima and reducing the optimisation error; nevertheless, in real-life applications, we never achieve a global minimum. The generalisation error refers to how the trained networks perform on unseen data and is mainly affected by the amount of training data, the modelling information of the PDEs, and the architecture of the neural networks.

The selection of DL-based PDE solvers, which are further investigated in this survey, is based on our endeavour to cover a most complete range of different concepts. Hence, we included the basic concepts of learned DL solvers, as well as some recent improvements of these.

A more detailed description of these approaches, their algorithmic implementation and their analytical properties will be given in the subsequent sections. We will also highlight, how these methods, which were typically designed as PDE-forward solvers, can be extended to parameter identification problems.

3.2 Characterisation of DL concepts for PDEs

Having specified the restricted scope of the present survey, we will now list the concepts under consideration in the following sections. Before starting the introduction of the individual methods, we include some general remarks on the characterisation of deep learning methods for PDEs. As already mentioned, the characterisation of DL concepts for solving PDEs differs considerably from the analytic characterisation of PDEs. They are mainly characterised in terms of the information needed for applying them. The two main classes of information are data and physical models, where 'physical models' refers to the mathematical formulation of the differential equations and their boundary conditions.

3.2.1 Data

Data-driven methods rely on the availability of sufficiently large data sets of good quality. The data provided can be sampled values of single or multiple solutions, where sampling can be done either on a predefined grid or with arbitrary sampling points. Some of the

most advanced theoretical investigations do start from assuming a sampling in function space, which opens the path for incorporating functional analytical machinery in their analysis, see [1, 24, 260].

However, the volume of available data is usually rather scarce,e.g., useful experimental data is generally limited or even intractable for many practical scenarios and high-fidelity numerical simulations are often computationally expensive. It is sometimes challenging to extract interpretable information and knowledge from the data deluge. Moreover, purely data-driven methods may fit observations very well, but predictions may be physically inconsistent or implausible. Theoretically, finite data cannot fully and correctly determine a mapping which maps an infinite set to an infinite set. However, we can still see the success of data-driven methods, which is mainly due to two reasons: firstly, the search for a solution or parameter typically follows a certain prior distribution, which we might be able to approximate well with few samples; and secondly most problems are continuous, which in connection with suitable regularisation schemes leads to good generalisation properties, even for limited sets of training data, [261]

In addition to sampled data, a wealth of domain-specific expert knowledge exists for most applications. This information may come in the form of observational, physical or mathematical understanding of the system under consideration. Integrating data and this prior knowledge can yield more interpretable machine learning methods and can improve the accuracy and generalisation performances without large amounts of data.

In our context, information from physics may come in different forms: most commonly, the system of partial differential equations as well as boundary conditions are specified. Otherwise, some energy functional or a weak formulation may be given. Also, certain approaches [277] incorporate conservation laws in the network architecture. Whenever such information is used, the concept is typically called 'physics informed'.

Different from data-driven methods, the physics information may uniquely determine the systems, however, only within the limitations of the chosen model. For example, given the PDE formula and some initial and boundary equations, the solution is uniquely determined if the system is well-posed. In other words, mathematical formulae may inherently contain all information of the physical systems, while data are just observations and reflections of the physical systems and thus cannot fully reveal the whole information of the systems.

The most common approach is to use this physics information in the loss function during the training of the network. Hence, the type of physics information determines the loss function.

3.2.2 Network training and application

Training the resulting networks typically leads to additional problems. This may require extensive hyperparameter search, data pre-processing or subtle parameter settings for

controlling the convergence of the training scheme. For the respective training concepts for each method, refer to [244].

As pointed out in [175], specific to physics-informed learning, there are currently three pathways that can be followed separately or in tandem to accelerate training and enhance the generalisation of ML models by embedding physics in them. The first one is **observational biases** from observational data, which is the foundation of data-driven methods in machine learning. The second is **inductive biases** by designing special NN architectures that can automatically and exactly satisfy some properties and conditions. The most common example are convolutional NNs which can be designed to respect properties, such as symmetry, rotations and reflections. Other examples are graph neural networks [212] which are adapted to graph-structured data. Another concept is based on quadratic residual networks(QRES) [39], which increases the potential of the neural networks to model non-linearities by adding a quadratic residual term to the weighted sum of inputs before applying activation functions. They were shown to have good performance in learning high-frequency patterns. The invertible neural network [8] makes the inverse of the neural network well-defined. It has applications in inverse problems and generative models. Specific architectures can also be designed to satisfy the initial/boundary conditions. The last pathway is **learning biases**, where the physics constraints are imposed in a soft manner by appropriately penalising the loss function of conventional NN approximations. Representative examples include the Deep Ritz method and PINN.

We also want to comment on the arguably most important tools for physics-informed machine learning, namely automatic differentiation [18]. In principle, this general concept allows us to compute exactly the derivatives of the network output with respect to the input variables. Hence, we can conveniently incorporate weak or strong formulations of the underlying PDE. However, compared to conventional numerical gradients such as finite difference, the automatic differentiation method is usually slower and requires more memory.

Moreover, after training, the methods will again differ substantially in their scope of applicability. Methods might allow the sampling of a solution of a PDE at arbitrary points. In this case, the method is called meshfree, which is one of the most desirable properties of most DL concepts for PDEs. Another important property, in particular for Level 2 and Level 3 problems, is the computational load needed for applying the network to a different parameter of the PDEs. This is most efficient for most operator learning schemes, but might require re-training or the computation of specific feature vectors.

In conclusion, different deep learning methods for PDEs are mainly different in terms of their neural network architecture and the particular choice of the loss function for training.

3.2.3 General DL concepts for PDE-based inverse problems

So far, we have sketched two main approaches for solving PDEs (forward problems), namely function or operator approximation schemes. In the following sections, we will introduce

these methods and their baseline implementation in more detail. We will also include, whenever appropriate, the related concepts for the PDE-based inverse problems.

The extension of function approximations $u_\Theta(t, x)$ for solving inverse problems is not straightforward and differs from method to method. In contrast to that, the application of learned operator approximation schemes $\Phi_\Theta(\lambda)$ to inverse problems is always based on one of the two following concepts:

- The direct (or backward) method, where the network is trained with reversed input-output. In other words, the solution of the PDE is the input of a neural network which outputs the parameter function. This method is similar to the forward problem, but we learn the backward operator.
- Tikhonov Regularisation, for this case, the forward operator $\Phi_\Theta(\lambda)$ is trained as usual. For an inverse problem with given noisy data, u^δ we then approximate a suitable parameter λ by minimising a Tikhonov functional with respect to λ

$$\|\Phi_\Theta(\lambda) - u^\delta\|^2 + \alpha R(\lambda).$$

  The choice of the penalty term R adds to the flexibility of the method and can encode further properties of the solution such as smoothness, sparsity or piece-wise constant structures. Minimisation is done iteratively via gradient descent, where the differentiation with respect to λ can be implemented using backpropagation with respect to the input, but with fixed weights Θ. We exemplify the procedure for the model reduction concept PCANN in Algorithm 5. A similar idea can be applied to the other solution operators mentioned in Sections 3.5.2 and 3.5.3.

3.3 Theoretical Results of Deep Learning for PDEs

One reason for the success of finite element methods is that the functions in Sobolev space can be approximated by piece-wise polynomials with a certain convergence rate depending on the mesh size. However, classical methods require degrees of freedom of the order of ε^{-d} for a desired accuracy ε in d dimensional problems, which limits their applications to high-dimensional problems. The task of deep learning for PDEs is to employ neural networks to approximate the solution or parameter-to-state operator with a limited amount of data or physical information. A fundamental question is therefore whether the neural network can effectively approximate the solution/parametric operator of a PDE system, and if so, how to estimate the complexity of a neural network in terms of the dimensionality growth and accuracy requirements. In this section, we will briefly review some approximation theoretical aspects of deep learning for PDEs.

3.3.1 Approximation Results for Solution Learning in PDEs

Neural networks are universal approximators [65, 145] in the sense that continuous functions on compact domains can be uniformly approximated by neural networks with arbitrary accuracy, provided the number of neurons is sufficiently large. A lower bound on the size of the neural network for achieving the desired accuracy was not given in these papers. With some assumptions on the activation functions and the space of functions to be considered, more refined results on the relationship between the size and the approximation accuracy of neural networks have been reported in [16, 29, 72, 107, 114, 115, 221, 230, 280]. The universal approximation theorem has also been generalised to convolutional neural networks in [332] and complex-valued neural networks in [308]. These results concern general function approximations and do not refer to PDE solutions.

Neural Networks Approximation for High Dimensional PDEs

We now shortly discuss the theoretical foundations for approximating solutions of PDEs by neural networks. Recent years have witnessed great empirical success of various deep learning methods [130, 149, 327, 329] in particular for solving high-dimensional PDEs. To explain these successes, several mathematical results have been proven showing that neural networks have the ability to approximate solutions of some PDEs without the curse of dimensionality (CoD). In particular, in [108] it is shown that the solutions of linear Black-Scholes PDEs can be approximated by neural networks with the size increasing at most polynomially with respect to both the reciprocal of the prescribed approximation accuracy and the PDE dimension d. A number of articles have appeared that significantly extend the results in [108] to more classes of PDEs [21, 102, 146, 151, 161, 263]. In particular, in [151] it is proven that neural networks have the expressiveness to overcome the CoD for semilinear heat PDEs.

On the other hand, there are also some articles, [109, 111, 254, 324] that derive some lower bounds on the complexity of neural networks with ReLU activation function to achieve a certain accuracy, showing that there are natural classes of functions where deep neural networks with ReLU activation cannot escape the CoD.

Barron Spaces

Typical classical approximation results for PDE solutions use a setting in Sobolev or Besov spaces. These spaces can be defined by their approximation properties using piece-wise polynomials or wavelets. In the same sense, the Barron space [53, 80] is the analogue when we consider approximation by two-layer neural networks. Roughly speaking, the Barron space consists of functions that can be approximated by two-layer neural networks, and the approximation error is governed by the norm of the Barron space. A nice property of Sobolev/Besov-type spaces is that solutions to partial differential equations lie in these spaces. This is the core of the regularity theory for PDEs. When introducing Barron spaces

in PDEs, a natural corollary and fundamental question is whether the solutions of the dimensional partial differential equations (PDEs) we are interested in belong to a Barron space. We will briefly review the definition of the Barron space and its approximation theorem, as well as an example of its use in escaping CoD in high-dimensional PDEs.

Consider functions $g \subset \quad ^d : X \to R$ which allow the below representation:

$$g(x;\Theta) = \int_\Omega a\sigma(w^T x + b)\rho(da, dw, db), \quad x \in \quad ^d, \tag{3.1}$$

where ρ is a probability distribution on $\Omega = \quad \times \quad ^d \times \quad$ and $\sigma : \quad \to \quad$ is some activation function. This representation can be thought of as a continuum realisation of two-layer neural networks, which are given as

$$g_n(x,\Theta) = \frac{1}{n}\sum_{k=1}^{n} a_k\sigma(w_k^T x + b_k), \quad x \in \quad ^d \ . \tag{3.2}$$

We present the following definition from [53] for Barron functions defined in a domain $\Omega \subset \quad ^d$. This definition is an adaptation of the definition given in [80], with some crucial modifications, for the purpose of analysing PDEs.

Definition 3.3.1. *For a domain $\Omega \subset \quad ^d$ and a fixed radius $R \in [0, +\infty]$, the corresponding Barron space with index p is defined as*

$$\mathcal{B}_R^p(\Omega) = \{f : \|f\|_{\mathcal{B}_R^p(\Omega)} < \infty\}, \tag{3.3}$$

where

$$\|f\|_{\mathcal{B}_R^p(\Omega)} := \inf_{\rho\in\mathcal{P}_R}\left\{\left(\int |a|^p\rho(da, dw, db)\right)^{1/p} : f = \int_\Omega a\sigma(w^T x + b)\rho(da, dw, db)\right\} \tag{3.4}$$

and $\mathcal{P}_R := \left\{\rho : \rho \quad \text{is supported on} \quad \times \{x \in \quad ^d : \|x\| \le R\} \times \quad\right\}$.

The most essential feature that distinguishes Barron function spaces from Sobolev or Besov spaces is that elements of the former can be approximated by a two-layer neural network with an approximation rate that is independent of the dimension, as shown in the next theorem [53]:

Theorem 1. *Suppose that the activation function σ is smooth with $C_0 := \sup_{z\in} \ |\sigma(z)| \le \infty, C_1 := \sup_{z\in} \ |\sigma'(z)| \le \infty$ and $\sup_{z\in} \ |\sigma''(z)| \le \infty$, and that $g \in \mathcal{B}_R^1(\Omega)$. Then for any open bounded subset $\Omega_0 \subset \Omega$ and any $n \in \quad _+$, there exists $\{(a_k, w_k, b_k)\}_{k=1}^n$ satisfying*

$$\left\|\frac{1}{n}\sum_{k=1}^{n} a_k\sigma(w_k^T x + b_k) - f\right\|_{H^1(\Omega_0)} \le \frac{2(C_0^2 + R^2C_1^2)m(\Omega_0)\|g\|_{\mathcal{B}_R^p(\Omega)}}{n} \ , \tag{3.5}$$

where $m(\Omega_0)$ is the Lebesgue measure of Ω_0.

The above theorem gives an H^1-approximation rate for the Barron space defined by its integral representation. Furthermore, for a family of second-order elliptic PDEs with some assumptions on the coefficients and the source term, it is proved in [53] that for any exact solution u and $\varepsilon \in (0, 1/2)$, there exists $u^* \in \mathcal{B}_R^1(\ \ ^d)$ with $R \leq c_1(\frac{1}{\varepsilon})^{c_2}$ and $\|u^*\|_{\mathcal{B}_R^1(\ \ ^d)} \leq c_3(\frac{d}{\varepsilon})^{c_4|\ln\varepsilon|}$ so that $\|u - u^*\|_{H^1(\ \ ^d)} \leq \varepsilon$. Thus, it is easy to conclude that there is a two-layer neural network $u_m(x;\Theta)$ with $m \leq c_5(\frac{d}{\varepsilon})^{c_6|\ln\varepsilon|}$ such that $\|u_m - u\| \leq \varepsilon$, where c_1, c_2, c_3, c_4 are constants independent of u^*, ε and d. Therefore, these results prove that even a simple two-layer neural network with a single activation function has sufficient representational power to approximate the solution of an elliptic PDE without CoD.

In [54], the regularity theory of solutions to the static Schrödinger equation in spectral Barron spaces was studied. In [80] the flow-induced function space was introduced and analysed by considering the residual neural networks. And the analysis in [161] shows that the solutions of some linear parabolic PDEs belong to a close analogue of the flow-induced spaces.

The Monte Carlo Approach

The approximation power of neural networks is the basic requirement for escaping CoD in high-dimensional PDEs, while the algorithms for finding the optimal parameters of the neural networks are also equivalently important when considering computational complexity. The approximation of high-dimensional integrals plays an important role in such high-dimensional problems. Let $f : D = [0, 1]^d \to \quad$ be a function in $L^2(D)$and let

$$I(f) = \int_D f(x)dx. \tag{3.6}$$

To approximate the integral (3.6), the Monte Carlo algorithm [79] randomly samples independent, continuous, uniformly distributed points $\{x_i\}_{i=1}^N$ from D and let

$$\mathcal{I}_N(f) = \frac{1}{N}\sum_i^N f(x_i). \tag{3.7}$$

Then by a simple calculation, we have

$$[|I(f) - \mathcal{I}(g)|^2] = \frac{\mathrm{Var}(f)}{N} \quad \text{and} \quad \mathrm{Var}(f) = \int_X |f(x)|^2 dx - \left[\int_x f(x)dx\right]^2. \tag{3.8}$$

Thus, the Monte Carlo algorithm, which approximates the integral in high-dimensional spaces, can escape the CoD in a probabilistic sense. The convergence rate of the Monte Carlo algorithm plays an important role in the theory of machine learning for high-dimensional PDEs. As we can see, the deep Ritz and WAN loss functions discussed later for PDEs all involve the evaluation of an integral over the domain of interest, and the PINN loss function can also be viewed as the integral of the squared residual of the PDE over its domain of definition. In addition, the paper [110] proved that if a function can be approximated by a

proper discrete approximation algorithm without CoD, and if there are neural networks that satisfy certain regularity properties and approximate this discrete approximation algorithm without CoD, then the function itself can be approximated by neural networks without suffering the CoD. Full history recursive multilevel Picard approximation methods(MLP) [79, 150] are some recursive nonlinear extensions of the classical Monte-Carlo approximation methods, and it has been shown that such approximation schemes also escape the CoD in the numerical computation of semilinear PDEs with general time horizons.

In summary, there exist a growing number of strict mathematical results which prove that deep neural networks have the expressive capacity to approximate the solutions of some specific PDEs without the CoD and that the Monte Carlo algorithm can provide an efficient method for computing the associated loss functions. Nevertheless, the conjecture that there is a deep learning-based approximation method that overcomes the CoD in terms of computational complexity in the numerical approximation of PDEs has not yet been fully analysed, for example, the convergence rate of the optimisation procedure to learn the solution is also required to overcome the CoD.

3.3.2 Neural Network Approximations for Parametric PDEs

Operator learning methods aim to approximate the parametric map connecting a parameter space with a solution state space (parameter-to-state map). In the literature, the proposed operator learning methods usually consider low-dimensional PDEs. However, the inputs and outputs of neural networks for operator learning are usually high-dimensional vectors, which requires the neural networks to have sufficient expressiveness to approximate the parametric map. The first successful use of neural networks in the context of operator learning appeared in [47], where the authors designed a novel learning architecture based on neural networks and proved that these neural networks yield a surprising universal approximation property for infinite-dimensional nonlinear operators. This was later extended to deeper networks, see [255].

A Theoretical Analysis of Neural Networks for Parametric PDEs with Reduced Basis Assumption

In this subsection, we will briefly review a theoretical result [194] for parametric PDEs, which is based on the assumption of an inherent low dimensionality of the solution manifold. We will present a brief overview of the arguments that lead to the approximation theorem obtained in [194]. The lemmas in this paper and the arguments used for proving them are constructive and have important value in the analysis of neural networks.

In [194], the authors consider parameter-dependent equations in their variational form:

$$\mathcal{A}_\lambda(u_\lambda, v) = f_\lambda(v), \quad \text{for all} \quad \lambda \in \mathcal{Y}, v \in \mathcal{H}, \tag{3.9}$$

where the parameter set $\mathcal{Y}$ is a compact subset of ^{p} with a fixed and potentially large p. $\mathcal{H}$ is a Hilbert space and $\mathcal{A}_\lambda$ is a symmetric, uniformly continuous and coercive bilinear

form, $f_\lambda \in \mathcal{H}^*$ is bounded by a constant C for all $\lambda \in \mathcal{Y}$, $u_\lambda \in \mathcal{H}$ is the solution and the solution manifold is assumed to be compact in $\mathcal{A}$.

The following is a simplified outline of the arguments of [194], which derived upper bounds on the size of neural networks with activation ReLU approximating the solution operator of parametric partial differential equations under the assumption that the solution space is inherently lying in a low-dimensional space.

1. Firstly, it recalled the result shown in [324] that the scalar multiplication $f(x, y) = xy$ for $x, y \in [0, 1]$ can be constructed by a ReLU NN of size $\mathcal{O}(\log_2(1/\varepsilon))$ up to an error of $\varepsilon > 0$.

2. As a next step, the approximate scalar multiplication is used to show that a matrix multiplication of two matrices of size $d \times d$ with entries bounded by 1 can be performed by NN of size $\mathcal{O}(d^3 \log_2(1/\varepsilon))$ up to an error of $\varepsilon > 0$.

3. For $A \in \quad^{d\times d}$ such that $\|A\|_2 \leq 1 - \delta$ for some $\delta \in (0, 1)$, the map $A \to \sum_{s=0}^{n} A^s$ can be approximated by a ReLU NN with an accuracy of $\varepsilon > 0$ and having a size of $\mathcal{O}(n \log_2^2(m) d^3 \cdot (\log_2(1/\varepsilon) + \log_2(n)))$. Furthermore, with the fact that the Neumann series $\sum_{s=0}^{n} A^s$ converges exponentially fast to $(\mathrm{Id} - A)^{-1}$, a ReLU NN can be constructed that approximates the inversion operator $B \to B^{-1}$ to accuracy $\varepsilon > 0$ under suitable conditions on the matrix B. This NN has the size $\mathcal{O}(d^3 \log_2^q(1/\varepsilon))$ for a constant $q > 0$.

4. Next, two assumptions are required, which are satisfied in many applications. The first is that the map from the parameter space to the associated stiffness matrices of the Galerkin discretisation of the parametric PDE with respect to a reduced basis can be well approximated by neural networks. The second is that the map from the parameters to the right-hand side of the variational form of the parametric PDEs discretised based on the reduced basis can be effectively approximated by neural networks. Then there exists a neural network that approximates the operator from parameters to the corresponding discretised solution with respect to the reduced basis. If the reduced basis is of size d and the implementations of the map obtaining the stiffness matrix and the right-hand side are adequately efficient, then the corresponding NN is of size $\mathcal{O}(d^3 \log_2^q(1/\varepsilon))$. Finally, if D is the size of the high-fidelity basis, then one can approximate a base change by applying a linear map $\mathbf{V} \in \quad^{D\times d}$ to a vector with respect to the reduced basis. This procedure increases the size of the NN to $\mathcal{O}(d^3 \log_2^q(1/\varepsilon) + dD)$.

Approximation Results for Operator Learning Methods with State-of-the-Art Architectures

In contrast to neural networks aiming at function approximations, several operator learning methods for PDEs aiming at learning the full parameter-to-state map have been proposed in recent years. These operator learning methods have their own specific and somewhat complex architectures. For example, the networks of Fourier neural operators are mainly defined in Fourier space or the DeepONets consist of a branch network that approximates the map and a trunk network that approximates the solution basis. Thus, the network approximation result in (3.3.2) cannot be directly applied to these state-of-the-art operator learning methods. However, there are also some approximation results available for some operator learning methods.

In [186] it is shown that in the worst case, the network size of the Fourier neural operator can grow exponentially in terms of accuracy when approximating general operators. However, the author also proves that, under suitable hypotheses, the size of the network in FNOs for approximating the parametric map for a Darcy-type elliptic equation or for the incompressible Navier-Stokes equations of fluid dynamics scales polynomially in the error bound.

For PCANN, under some assumptions on the probability measure of the parameter and solution spaces, in [24] it is proven that for any error level $\varepsilon > 0$ there exist dimensions for the reduced basis of the parameter and solution spaces, and a network with maximum layers and widths, such that the neural network can approximate the operator up to a certain error ε associated with the probability distribution.

In DeepONet [222], the basis functions of the solution space are represented by the trunk net. Thus, not only is the branch net required to have approximation capability to learn the map connecting the parameter functions and the coefficients of the solutions with respect to the basis of the trunk net, but the trunk net should also efficiently approximate the basis of the solutions. A first answer to this question lies in a remarkable universal approximation theorem for the operator network first proven in [47]. More refined results are given in [197, 222]. In particular, in [197] the authors extended the universal approximation theorem from continuous to measurable functions, while removing the compactness requirements. Upper and lower bounds on the DeepONet error are given in [197] with respect to the number of sensors, the number of branch and trunk networks p, and the sizes and depths of the networks.

Some approximation results for operator learning with convolutional neural networks have also been derived. In [87], the authors established and verified theoretical error bounds for the approximation of nonlinear operators using convolutional neural networks. The results shed light on the role of convolutional layers and their hyperparameters, such as input and output channels, depth and others.

3.4 DL concepts based on function approximation

In this section, we list the most common DL concepts based on function approximation for PDE forward solvers and their related inverse problems.

3.4.1 Deep Ritz Method

Motivation

The Deep Ritz method [327] is a function evaluation concept based on a combination of the classical Ritz method (variational method) and deep learning. The Deep Ritz method assumes that a variational formulation of a PDE as stated in Equation (1.1) exists, i.e., there exists a $\Psi : \quad \rightarrow \quad$ such that the unique solution u of the PDE is given by

$$u = \text{argmin}_{v \in H} I(v) \tag{3.10}$$

where

$$I(u) = \int_{\Omega} \Psi(u(x)) dx \tag{3.11}$$

and H is the set of admissible functions. Hence, the Deep Ritz method is a physics-informed method.

The idea of the Deep Ritz method is to replace u with a neural network u_Θ which has a scalar or vectorial x as input. The network is trained by choosing suitable collocation points $\{x_i\}_{i=1}^N \subset \Omega$ and by replacing the integral in the variational formulation by a finite sum

$$\min_{\Theta} \sum_{i=1}^{N} \Psi(u_\Theta(x_i)) \ . \tag{3.12}$$

After training, u_Θ can be evaluated efficiently at arbitrary points in the domain of definition.

Network architecture

For the numerical examples for our standard test problems, see Section 4.1, we used the network shown in Figure 3.3. The main building block of the Deep Ritz neural network consists of two stacked fully connected (FC) layers which are each followed by a non-linear somewhat "smooth" activation function. A residual connection, which can help to avoid the vanishing gradient, links the input of the first FC layer to the output of the second FC layer. Several blocks are then stacked to complete the architecture of the network.

Consider the i-th block of the neural net. Let $W_{i,1}, W_{i,2} \in \quad^{m \times m}$ and $b_{i,1}, b_{i,2} \in \quad^{m}$ be the respective weight and bias of the first and second FC layers. A block which receives an input s, performs the operation

$$B_i(s) = \sigma\left(W_{i,2} \cdot \sigma\left(W_{i,1} s + b_{i,1}\right) + b_{i,2}\right) + s, \tag{3.13}$$

where σ is the activation function. From experience,

$$\sigma(x) = \max\left\{x^3, 0\right\}, \tag{3.14}$$

provides a good balance between accuracy and simplicity. Its smoothness contributes to the accuracy of the Deep Ritz method. To match the input, resp. output, dimensions of the considered problem, an appropriate FC layer is used at the entrance (resp. exit) of the first (resp. last) layer of the first (resp. last) block of the neural network. Figure 3.3 shows the architecture of the Deep Ritz network.

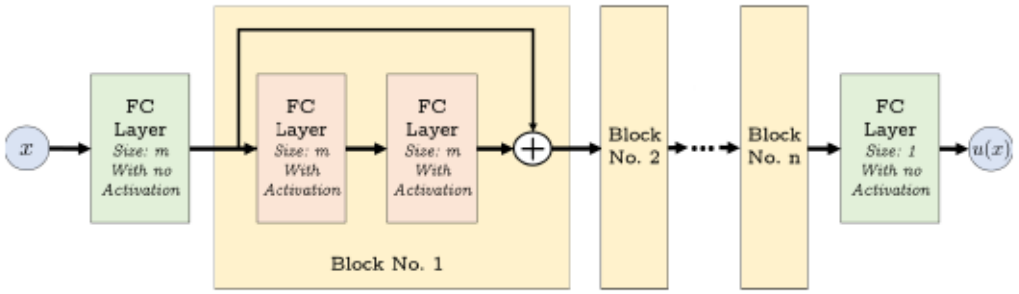


Figure 3.3: Deep Ritz network with n blocks and with appropriate input and output FC linear layers. The first FC linear layer can be replaced by zero padding

The algorithm

The training of the network does not need any information or data of the solution u, hence, with this respect, it is equivalent to classical numerical PDE solvers. However, it needs the PDE in its variational formulation, hence, it is a physics-informed concept. The network architecture will be of the type 'function approximation', i.e., a rather small network with two or three nodes in the input level and a single value for output. This can be adjusted for higher-dimensional PDEs accordingly. The training of the network uses a loss function, which is simply a discretisation of the variational formulation. A second term is added for ensuring the boundary conditions. Hence, first of all, it requires sampling a suitable set of evaluation points $x_i, i = 1, .., N$ in the domain of definition and on the boundary. This can be done via stochastic sampling of a uniform distribution over the domain of definition (Monte Carlo integration) on a fixed grid. The training itself is then done by classical stochastic gradient descent. The sampling points may be changed during training after a fixed number of epochs. We specify the approach for the classical Poisson problem

$$\begin{aligned} \Delta u &= \lambda \text{ on } \Omega \subset \mathbb{R}^2 \\ u &= g \text{ on } \partial\Omega \end{aligned} \tag{3.15}$$

This has the variational form: $I(u) = \int_\Omega \left(\frac{1}{2}|\nabla u(x)|^2 - \lambda(x)u(x)\right) dx$, see Algorithm 1.

It has been numerically shown that the Deep Ritz has the power to solve high-dimensional problems, this is mainly due to the use of the Monte Carlo algorithm for approximating the integral. In addition, Deep Ritz is potentially a naturally adaptive algorithm that can solve problems with corner singularities. One drawback of the Deep Ritz method is that the neural network parameterises the solution function instead of the parameter-solution

Algorithm 1: Deep Ritz Method

Input:
- $N_{\mathcal{L}}/N_{\mathcal{B}}$: number of inner/boundary collocation points
- τ: learning rate

Initialise the network architecture u_Θ
while not converged **do**
  Sample collocation points $\{x^i_{\mathcal{L}} \in \Omega : i = 1, \dots, N_{\mathcal{L}}]\}$ and $\{x^j_{\mathcal{B}} \in \partial\Omega : j = 1, \dots, N_{\mathcal{B}}\}$
  Compute $E_{\mathcal{L}}(u_\Theta) = \frac{1}{N_{\mathcal{L}}} \sum_{i=1}^{N_{\mathcal{L}}} \left(\frac{1}{2}|\nabla u_\Theta(x^i_{\mathcal{L}})|^2 - \lambda(x^i_{\mathcal{L}}) u_\Theta(x^i_{\mathcal{L}})\right)$, via backpropagation
  Compute $E_{\mathcal{B}}(u_\Theta) = \frac{1}{N_{\mathcal{B}}} \sum_{j=1}^{N_{\mathcal{B}}} \left(u_\Theta(x^j_{\mathcal{B}}) - g(x^j_{\mathcal{B}})\right)^2$
  Add loss values $L = E_{\mathcal{L}}(u_\Theta) + E_{\mathcal{B}}(u_\Theta)$
  Optimise Loss, L using the appropriate optimisation algorithm.
  Update $\Theta \leftarrow \Theta - \tau \nabla_\Theta L$
end

operator. Hence, the Deep Ritz method is less suited for parametric studies, since it requires a new training of the full network for every new parameter. Note that the loss function of the Deep Ritz method can be negative and the minimal value is usually unknown, this also causes some challenges in the training process. In addition, the minimisation problem that results from Deep Ritz is usually not convex even when the original problem is. The treatment of the essential boundary condition is not as simple as for traditional methods.

Theoretical Background

The convergence properties of the Deep Ritz methods have been analysed intensively over the last years [75, 77, 162, 241]. The most far-reaching results - to the best of our knowledge - are described in [75]. This paper also contains a nicely written survey on the state of the art of other results concerning convergence properties. This paper uses techniques from Γ-convergence for proving that, under rather mild assumptions on the network architectures, the loss function of the network training Γ-converges to the true variational formulation and the minimisers also converge weakly to the true solution of the PDE, see Theorem 7 [75].

3.4.2 Physics-informed Neural Network (PINN)

Motivation

The notion of 'physics-informed neural networks' is now used in more general terms. In this section, however, we discuss the baseline version of a PINN as introduced in the original paper [261]. Similar to the previously described Deep Ritz method, this original PINN concept parameterises the solution function as a neural network and requires knowledge of the underlying In contrast to the Deep Ritz method, PINNs use the strong form of the PDE, thus their application to general PDEs is straightforward. To describe the method, we split up the operator $\mathcal{N}$ in equation (1.1) into a differential operator $\mathcal{L}$ encoding the PDE and the initial/boundary operator $\mathcal{B}$. The problem then reads

$$\begin{aligned} \mathcal{L}(u;\lambda) &= 0 \quad \text{in} \quad \Omega \\ \mathcal{B}(u;\lambda) &= 0 \quad \text{on} \quad \partial\Omega. \end{aligned} \tag{3.16}$$

The neural network u_Θ is again substituted into the PDE, where the differential operator $\mathcal{L}$ can be applied to u_Θ via automatic differentiation (backpropagation). We note that $\mathcal{L}(u_\Theta;\cdot)$ and $\mathcal{B}(u_\Theta;\cdot)$ rely on the same set of parameters Θ as u_Θ, which is crucial for the optimisation of PINNs. To fit u_Θ to the PDE, we penalise the residuals of both operators by their mean squared error on previously sampled points. This leads to the training objective

$$\min_\Theta \left(MSE_\mathcal{L}(u_\Theta) + MSE_\mathcal{B}(u_\Theta) \right), \tag{3.17}$$

where

$$MSE_\mathcal{L}(u_\Theta) = \frac{1}{N_\mathcal{L}} \sum_{i=1}^{N_\mathcal{L}} |\mathcal{L}(u_\Theta;\lambda)(x_\mathcal{L}^i)|^2 \quad \text{and} \quad MSE_\mathcal{B}(u_\Theta) = \frac{1}{N_\mathcal{B}} \sum_{j=1}^{N_\mathcal{B}} |\mathcal{B}(u_\Theta;\lambda)(x_\mathcal{B}^j)|^2. \tag{3.18}$$

During optimisation, $MSE_\mathcal{B}$ enforces the initial or boundary conditions of a given problem and $MSE_\mathcal{L}$ checks the differential equation at the collocation points. A visual representation of the general idea is shown in Figure 3.4.

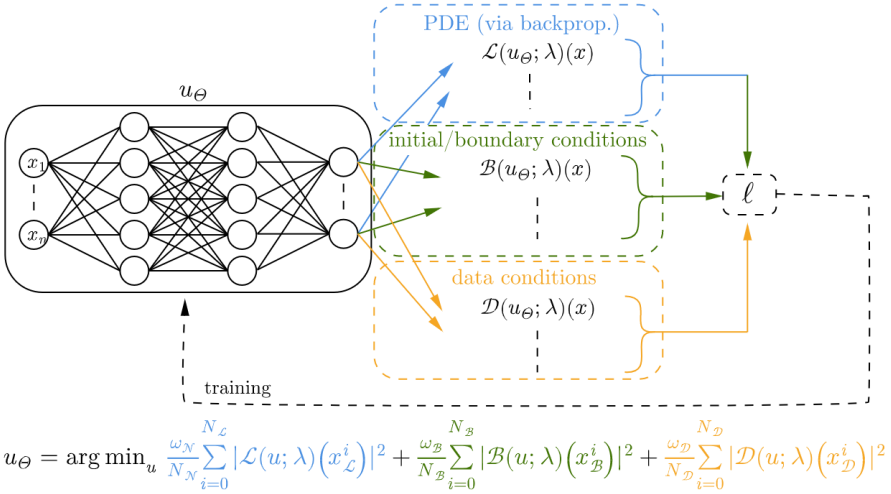


Figure 3.4: General training procedure of the PINN approach. As compared with the conditions in (3.18), we included a data condition in this figure that refers to observations of the solution on some points.

Network architecture

The architecture of the neural network u_Θ allows a lot of freedom, the original paper [261] uses a simple fully-connected architecture. Depending on the underlying differential equation, different network structures can lead to better approximation results. This promotes the existence of a great variety of extensions of the PINN approach, either focused on individual problems or designed for broader classes of equations. We will mention a few of these methods in Section 8.

The algorithm

The general training procedure is comparable to the Deep Ritz algorithm 3.4.1. Again, the network itself is optimised to approximate the solution as a function, therefore the number of input nodes corresponds to the dimensionality of Ω and the amount of output nodes denotes the dimensionality of the possibly vector-valued solution. The training requires the underlying PDE and some evaluation points $x^i_{\mathcal{L}}, x^j_{\mathcal{B}}$, $i = 1, \dots, N_{\mathcal{L}}$, $j = 1, \dots, N_{\mathcal{B}}$ to compute the residuals. The sampling strategy and therefore the distribution of these points can be chosen freely and may be changed throughout the training. Summarising the approach for the Poisson problem of Equation 3.15, the general algorithm is shown in 2.

Algorithm 2: Physics-Informed Neural Networks

Input: $N_{\mathcal{L}}/N_{\mathcal{B}}$: number of inner/boundary collocation points,
τ: learning rate

Initialise the network architecture u_Θ
while not converged **do**
  Sample collocation points $\{x^i_{\mathcal{L}} \in \Omega : i = 1, \dots, N_{\mathcal{L}}]\}$ and $\{x^j_{\mathcal{B}} \in \partial\Omega : j = 1, \dots, N_{\mathcal{B}}\}$
  Compute $MSE_{\mathcal{L}}(u_\Theta) = \frac{1}{N_{\mathcal{L}}} \sum_{i=1}^{N_{\mathcal{L}}} |\Delta u_\Theta(x^i_{\mathcal{L}}) - \lambda(x^i_{\mathcal{L}}))|^2$, via backpropagation
  Compute $MSE_{\mathcal{B}}(u_\Theta) = \frac{1}{N_{\mathcal{B}}} \sum_{j=1}^{N_{\mathcal{B}}} |u_\Theta(x^j_{\mathcal{B}}) - g(x^i_{\mathcal{B}}))|^2$
  Add loss values $L = MSE_{\mathcal{L}}(u_\Theta) + MSE_{\mathcal{B}}(u_\Theta)$
  Update $\Theta \leftarrow \Theta - \tau \nabla_\Theta L$
end

The generality of PINN allows the extension of the loss function to include several additional conditions. For example, if some observed data is available, it can be seamlessly incorporated into the loss function. In addition, PINN can be extended to solve integro-differential equations [225], fractional PDEs [252] and stochastic PDEs [330]. Furthermore, by using a discrete-time model (Runge-Kutta methods) to link two distinct temporal snapshots, the number of collocation points for time-dependent PDEs can be reduced significantly.

Application to parameter identification

Notably, PINNs are also suitable for the parameter identification of PDEs. Given a set of observed data points, the unknown parameters of the PDE can be optimised alongside the parameters of the solution network u_Θ. From an implementation point of view, PINNs can therefore be applied to parameter identification problems requiring minimal additional work, see [48, 159, 226, 260, 261, 328].

While the identification/learning of scalar parameters is quite straightforward, that of function-valued (e.g., space-dependent and/or time-dependent) parameters requires some additional effort. For function-valued parameters such as the right side, λ in equation 3.15 or the coefficient λ in 4.3 of a certain PDE, one usually introduces an additional neural network λ_Θ for the searched functions. The idea behind these networks is comparable to u_Θ which parameterises the PDE solution as described in 2, i.e., we input the space and/or time

variables and the network should output an approximation of the searched function at these values. The training process then consists of a combination of physics-informed loss and data loss. Via the data loss, the network u_Θ learns an interpolation of the given (measured) data. A physics-informed loss, like the $MSE_\mathcal{L}$ term in 3.18, then enables us to learn the searched functions, by using the learned interpolation u_Θ. During the minimisation of the sum of both loss terms, the corresponding weights of all appearing networks then are optimised simultaneously via automatic differentiation. The resulting loss function which is optimised, e.g., in the case of the Poisson problem 3.15 is thus

$$L = \underbrace{\frac{1}{N_\mathcal{P}} \sum_{i=1}^{N_\mathcal{P}} |\Delta u_\Theta(x^i_\mathcal{P}) - \lambda_\Theta(x^i_\mathcal{P}))|^2}_{\text{Physics loss}} + \underbrace{\frac{1}{N_\mathcal{D}} \sum_{j=1}^{N_\mathcal{D}} |u_\Theta(x^j_\mathcal{D}) - u(x^j_\mathcal{D}))|^2}_{\text{Data loss}}, \tag{3.19}$$

where $N_\mathcal{P}$ is the number of collocation points, and $N_\mathcal{D}$ is the number of known data points, such that $\{x^i_\mathcal{P} \in \Omega : i = 1, \dots, N_\mathcal{P}]\}$ and $\{x^j_\mathcal{D} \in \partial\Omega : j = 1, \dots, N_\mathcal{D}\}$.

In the case of noisy data, the physics can also act as a regularisation for the learned interpolation, since the loss also influences u_Θ. Even when only discrete data points of the solution are available, the physics loss may be evaluated on arbitrary points, because of the interpolation properties of u_Θ. Moreover, the flexible framework of PINNs allows for further inclusion of a-priori knowledge on the solution or parameter functions, such as their regime or boundary values, by similar incorporation into the loss.

In the numerical experiments in Section 4.1, we will use simple fully connected neural networks (FCNNs), consisting of multiple FC layers. As an activation function, we will use $\phi(x) = \tanh(x)$.

Theoretical Background

In contrast to the large number of applications of PINNs, they still lack a rigorous theoretical background. First consistency results under additional assumptions for linear elliptic and parabolic PDEs, can be found in [325]. Some estimates on the generalisation error of PINNs approximating solutions for forward and inverse problems are proven in [234, 235]. The authors derive an error estimation in terms of training error and training points by exploiting the stability properties of the underlying equation. More general convergence results of the PINN approach still have to be found.

Some studies of the convergence rate were carried out in [311, 313]. There the Neural Tangent Kernel (NTK) theory [157] was applied to the PINN framework, which gave the first insight into the convergence behaviour of the different loss terms and spectral bias of PINNs. With the NTK theory special weights for the different terms can be found, to achieve more robust and accurate results. In [312], the same approach was also applied to the DeepONet architecture, which will be introduced in Section 3.5.3.

Generalisations and extensions of PINN

Lastly, we want to give a short overview of possible extensions and generalisations for PINNs. Since the literature in this regard is vast, we cannot represent every approach.

- **Quadratic Residual Networks (QRES)** [39]: Extends PINNs by including quadratic terms in each layer. The architecture used consists of FC layers with additional weight matrices, see Figure 3.5. In each layer, the output of the FC layer matrix and additional matrix are point-wise multiplicated and added to the FC layer output, to create a quadratic term. This may lead to faster convergence and better parameter efficiency with respect to network width and depth.

- **Residual-based adaptive refinement** [225]: The idea of RAR is to add more residual points in the locations where the PDE residual is large during the training process, conceptually similar to FEM refinement methods.

- **Gradient-enhanced PINN (gPINN)** [328]: If the PDE residual $\mathcal{L}u$ is zero, then it is clear that the gradient of the PDE residual, i.e., $\nabla\mathcal{L}u$, should also be zero. Thus, the gradient-enhanced PINN (gPINN) uses a new type of loss function by leveraging the gradient information of the PDE residual to improve the accuracy and training efficiency of PINNs.

- **XPINN and cPINN** [158, 159]: The XPINN and cPINN apply domain decomposition, and then different subdomains use independent neural networks to approximate the solutions. In addition to the loss function used in PINN, interface conditions are introduced to stitch the decomposed subdomains together in order to obtain a solution for the governing PDEs over the complete domain. These extensions efficiently lend themselves to paralleled computation and are able to solve more general PDEs, e.g., PDEs with jump parameter functions.

For further details on the PINN approach, we suggest the paper [64], which contains a comprehensive and nicely written survey for the current state of the art of the PINNs. It mentions different applications, extensions, theories, general challenges and currently open questions.

3.5 DL concepts based on operator approximation

In this section, we introduce the basic concepts for DL approaches which aim at approximating the parameter-to-state operator. By construction, these methods are perfectly suited for parametric studies and can be adapted directly to inverse problems. The two basic concepts for adaptation to inverse problems have already been described in Section 3.2.3.

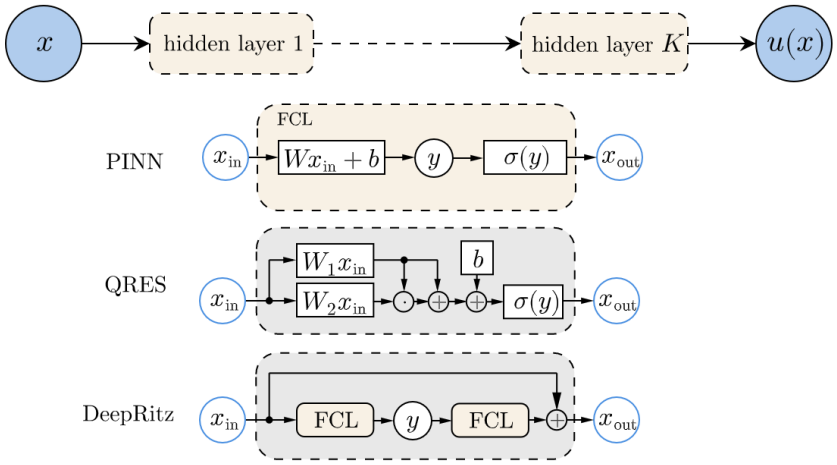


Figure 3.5: Overview and comparison of network architectures used in PINN, QRES and Deep Ritz. With weight matrix W, bias vector b and activation function σ.

3.5.1 Model Reduction and Neural Networks for Parametric PDEs (PCANN)

Motivation

PCANN, seeks to provide a meshless operator for the evaluation of the solution of a PDE by combining ideas of model order reduction with deep learning. For given training data (λ_i, u_i) one first obtains a model reduction by use of the principal component analysis (PCA) for both the input (parameter λ) and output (solution u) functions. Only the coefficients of a finite number of PCA components are retained. The PCA thus reduces the dimensions of both the input and output spaces to finite latent dimensional spaces. A neural network then maps the coefficients of the respective representations in these latent spaces as shown in Figure 3.2. The evaluation of this operator approximation for a novel parameter λ is then most efficient: one only needs to compute the scalar products with the specified finite number of PCA components, the neural network then maps these coefficients to the latent coefficients of the output space and an expansion using these coefficients and the PCA on the output side gives a function approximation to the solution of the PDE.

The formulation of this approach is in a function space setting and hence mesh-free. For implementation purposes, however, one has to specify how to compute the scalar products with the PCA components. These functions are only given numerically, usually by their values on a specified grid. The red arrow in Figure 3.2 shows the flow during testing. The overall PCANN can be used to evaluate the solution of the PDE for any given grid size as illustrated in Algorithm 3. In line 2-3 of Algorithm 3, it is possible to use one of the numerous PCA algorithms available [319]. For the implementations in this work, we make use of randomised PCA as proposed in [125, 126].

In a situation where the input-to-output operator is linear, like in the case of the Poisson problem considered, it might be beneficial to use a simple linear map for the

mapping of the reduced/latent dimensions. Section 4.1, equally shows results for this case, PCALin, where a single linear layer with no activation functions is used to map the latent dimensions. PCALin is similar to the linear method proposed in [24], however, we use a neural network instead of the normal equations. The results in both cases are similar as we later show in section 4.1. The combination of model order reduction with deep learning has recorded success in a number of application problems ranging from cardiac electrophysiology [91, 95], where the use of proper orthogonal decomposition (POD) further improves the results [94], to fluid flow [93] and non-linear models [61, 92]. Specifically, the PCANN operator has been used in [188, 217] in a multiscale plasticity problem to map strain to stress. Worth highlighting is an earlier work [139], which equally combines model order reduction with neural networks to solve PDEs.

Network Architecture

In our numerical examples, we followed the outline of the original paper [24] and used a fully connected feed-forward neural network for the mapping of the latent spaces. The number of nodes per layer is as follows $d_{\mathcal{X}}, 500, 1000, 2000, 1000, 500, d_{\mathcal{Y}}$. $d_{\mathcal{X}}$ and $d_{\mathcal{Y}}$ are considered to be hyperparameters, and for convenience, are chosen to be equal. As activation function we use the Scaled Exponential Linear Unit (SELU) activation function which that induce self-normalizing properties in feed-forward neural networks [178]. Figure 3.6 shows a simplified schematic of the overall architecture.

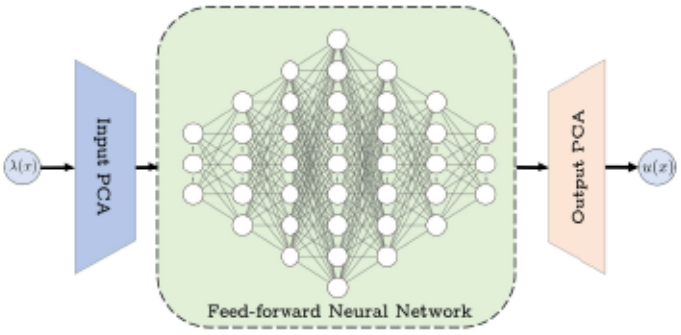


Figure 3.6: Architecture of the PCANN.

The Algorithm

As a purely data-driven method, no physics or PDE is needed in the training of the model. However, the data used for the training of the network is obtained from a classical FDM for solving the underlying By training the network with these input-output (numerically-given) function pairs, we obtain a neural operator which solves the PDE for various instances irrespective of the mesh size.

Once again, we specify the algorithm for the Poisson problem in Equation 3.15. The training data is the pair (λ_i, u_i), with each $\lambda_i, u_i \in \mathbb{R}^{s^2}$. Originally, λ_i, u_i are functions

given in a (square) grid of dimensions $s \times s$. By flattening these functions, we now have them in $^{s^2}$. Training then proceeds as in Algorithm 3, while testing/usage of the trained network proceeds as in Algorithm 4.

Theoretical Background

As the PCANN method combines both the PCA and Neural network (NN) to achieve the task of operator learning, the strength of its approximation, therefore, comes from that of both the PCA and NNs. For a good approximation, one thus needs not only a good neural network but also, appropriate PCA truncation parameters (resulting from choosing sufficient amounts of data). The existence of all these factors is shown in Theorem 3.1 of [24] which is a consequence of Theorem 3.5 of the same work. A recent work [195] further extends the approximation theory of PCANN. Notably, it derives novel approximation results, with minimal assumptions [195].

Algorithm 3: PCANN

Input:
- (λ_i, u_i): Training Data pair with $i = 0 \dots N_{\text{train}}$
- τ: learning rate

Initialise the network Φ_Θ
Compute PCA of (λ_i), store PCA $(a_k), k = 1, \dots, d_{\mathcal{X}}$
Compute PCA of (u_i), store PCA $(b_\ell), \ell = 1, \dots, d_{\mathcal{Y}}$
for $i \leftarrow 0, \dots, N_{\text{train}}$ **do**
    compute $c_i^k = \langle \lambda_i, a_k \rangle \in R^{d_{\mathcal{X}}}$
    compute $d_i^\ell = \langle u_i, b_\ell \rangle \in R^{d_{\mathcal{Y}}}$
end
while not converged **do**
    `/* Train network */`
    for $i \leftarrow 0, \dots, N_{\text{train}}$ **do**
        Compute Loss, $L_i = \sum_{\ell=1}^{d_{\mathcal{Y}}} \left\| \frac{\Phi_\Theta\left(c_i^k\right) - d_i^\ell}{d_i^\ell} \right\|_2$
    end
    Compute the Loss, $L = \sum_{i=1}^{N_{\text{train}}} L_i$
    Optimise L using the appropriate/chosen optimisation algorithm.
    Update $\Theta \leftarrow \Theta - \tau \nabla_\Theta L$
end

Algorithm 4: Using/Testing the PCANN

Input:
- Input parameter functions λ_i with $i = 0 \dots N_{\text{test}}$
- Input PCA basis $(a_k)_{k=1,\dots,d_{\mathcal{X}}}$
- Output PCA basis $(b_\ell)_{\ell=1,\dots,d_{\mathcal{Y}}}$
- Trained network Φ_Θ.

Result: Output solution functions $\tilde{u_i}$, with $i = 0 \dots N_{\text{test}}$

for $i \leftarrow 0, \dots, N_{\text{test}}$ **do**
    $\tilde{u_i} = \sum_\ell \Phi_\Theta\left(\langle \lambda_i, a_k \rangle\right)_\ell b_\ell$
end

PCANN adaptation for inverse problems

Following the general concepts as discussed in Section 3.2.3 we have two principal options. We either train a PCANN network $\Psi_\Theta(u)$ with training data (u_i^δ, λ_i), where we reverse the input-output order. I.e., the (u_i^δ) are used as input and (λ_i) as output. After training we can directly use a new measurement (u^δ) and compute a corresponding parameter $\hat{\lambda} = \Psi_\Theta(u^\delta)$.

The other option is to use a trained network for the forward problem Φ_Θ and include this into a Tikhonov functional. The resulting algorithm is summarized as follows:

Algorithm 5: Using trained forward PCANN for Inverse Problem

Input:
- Solution function u^δ
- Input PCA basis $(a_k)_{k=1,\dots,d_\mathcal{X}}$
- Output PCA basis $(b_\ell)_{\ell=1,\dots,d_\mathcal{Y}}$
- Trained network $\Phi_\Theta : c = (\langle \lambda, a_k \rangle) \mapsto d = (\langle u, b_\ell \rangle)$.

Result: Output parameter function $\hat{\lambda}$

Initialise c^0
for $m \leftarrow 0, \dots, N$ *(large integer)* **do**
  Compute $J_\alpha(c) = \left\| \Phi_\Theta(c) - \langle u^\delta, b_\ell \rangle \right\|_2$
  Compute the loss $J_\alpha(c) := J_\alpha(c) + \alpha R(c)$.
  Optimise Loss $J_\alpha(c)$ using gradient descent
  Update $c^{m+1} \leftarrow (c^m - \sigma_m \partial J / \partial c\,(c^m))$
end
Compute $\hat{\lambda} = \sum_k c_k^N a_k$

3.5.2 Fourier Neural Operator

Motivation

Fourier neural operators (FNO) [211] are designed as deep learning architectures for learning mappings between infinite-dimensional function spaces. The Fourier neural network is formulated as an iterative architecture, where each iteration (hidden layer), inspired by the convolution theorem, is a Fourier integral operator defined in Fourier space. The main network parameters are therefore defined and learned in Fourier space rather than in physical space, i.e., the coefficients of the Fourier series of the output function are learned from the data.

While standard feed-forward neural networks (FNN) and CNNs combine linear multiplications with non-linear activations in order to learn highly nonlinear functions, FNOs combine linear integral operators with non-linear activations in each layer to learn non-linear operators. The fast Fourier transform (FFT) makes the implementation even more efficient.

Different from DeepONet, and similar to the PCANN methods, FNO discretises both the input function $\lambda(x)$ and the output function $u(x)$ by using point-wise evaluations in an equispaced mesh. In addition, FNO requires that λ and u be defined on the same domain and are discretised by the same discretisation. The function space formulation of the approach allows training an FNO on a dataset with low resolution and applying it directly on a dataset with higher resolution, thus achieving the so-called zero-shot super-resolution.

Theoretical Background

The idea of FNO, stems from an earlier paper by the same team, see [212], which also seeks to approximate a neural operator by concatenating multiple hidden layers. While standard artificial neural networks (FCN, CNN) have affine functions (weights and biases) with scalar nonlinear activation as hidden layers, neural operators instead have affine operators (usually, in addition to affine functions with weights and biases [186] [224]), and with scalar nonlinear activation functions. A hidden layer numbered $j+1$ thus performs the update of input v_j as follows

$$v_{j+1}(x) := \sigma\left(Wv_j(x) + \left(\mathcal{K}(\lambda;\theta)v_j\right)(x)\right), \quad \forall x \in \Omega, \tag{3.20}$$

where W and $\mathcal{K}$ are the respective weights, biases and neural operators. These neural operators are chosen to be integral operators of the form

$$(\mathcal{K}(\lambda;\theta)v)(x) = \int_\Omega \kappa_\theta(x, y; \lambda(x), \lambda(y))v(y)dy, \quad \forall x \in \Omega. \tag{3.21}$$

These operators can be realised by different concepts such as graph kernels [212], multipole expansions [213], non-local kernels [326], low-rank kernels [187], wavelet transforms [302], multiwavelet transforms [119], or Laplace transforms [45]. A recent work even uses some famous convolutional neural networks architectures [262]. In the special case where $\kappa_\theta = \kappa_\theta(x-y)$, Equation 3.21 becomes

$$\begin{aligned}
(\mathcal{K}(\theta)v)(x) &= \int_\Omega \kappa_\theta(x-y)v(y)dy, \quad \forall x \in \Omega \\
&= (\kappa_\theta * v)(x), \quad \forall x \in \Omega \\
&= \left(\mathcal{F}^{-1}(\mathcal{F}(\kappa_\theta)\cdot\mathcal{F}(v))\right)(x), \quad \forall x \in \Omega \\
&= \left(\mathcal{F}^{-1}(K_\theta \cdot \mathcal{F}(v))\right)(x), \quad \forall x \in \Omega,
\end{aligned}$$

by the use of the convolution theorem. This then leads to the parameterisation of the neural network given by κ_θ directly in the Fourier space K_θ. [186] provides a good theoretical background of FNOs and [187] offers a good overview of neural operators using different integral operators along with some theoretical justification.

It is worth mentioning the recent work in [196], which introduces the Nonlocal Neural Operator (NNO), that generalises over arbitrary geometries. FNO is a special case of the NNO, and this work highlights how increasing the number of channels in FNO as opposed to increasing the number of Fourier modes benefits the FNO. The same work also introduces the averaging neural operator which is a subclass of NNO but also happens to be at the core of many neural operator frameworks [196].

The Algorithm

The classical FNO is only possible with a uniform grid (cartesian domain), but can be extended for any mesh. We note that, unlike the PCANN method, the input and output functions are inputted in their unflattened states. Once again, as an example, we consider the problem in Equation 3.15 and we seek to learn the operator

$$F : \Lambda(\Omega;\ ^{d_\lambda}) \ni \lambda(x) \mapsto u(x) \in \mathcal{U}(\Omega;\ ^{d_u}), \tag{3.22}$$

where d_λ and d_u depend on the discretisation used ($d_\lambda = d_u = 513^2$ if we use functions with a resolution of 513×513), Ω is a subset of while Λ and $\mathcal{U}$ are Banach spaces of functions taking values in .

Network Architecture

The architecture of the FNO follows from the algorithm described in Section 3.5.2. For the special case in Equation 3.15, we consider a mesh of resolution $N \times M$. We denote each point in the domain Ω as x_k^l, with input $\lambda(x_k^l)$, and output $u(x_k^l)$, with $k \in \{1, \ldots, N\}$ and $l \in \{1, \ldots, M\}$. We equally denote $\lambda(x)$ and $u(x)$ as the input and output for the whole domain.

- The FCN P_θ takes each $\lambda(x_k^l) \in$ and maps it to a higher dimension d_{v_0}, i.e.,

$$\begin{aligned} v_0(x_k^l) &= P_\theta(\lambda(x_k^l)) \in\ ^{d_{v_0}}, \\ v_0(x) &= P_\theta(\lambda(x)) \in\ ^{d_{v_0} \times N \times M}. \end{aligned}$$

  $v_0(x)$ is thus an $N \times M$ 'image' with d_{v_0} channels.

- To obtain $v_1(x)$ from $v_0(x)$, the following operations are applied.

  - The (2D) Fourier transform $\mathcal{F}$ is applied to each channel of $v_0(x)$, and the first $k = k_1 \cdot k_2$ modes are kept (k_1, k_2, being the number of modes to be kept in the first and second dimensions respectively). As a result, we obtain $\mathcal{F}(v_0(x)) \in\ ^{d_{v_0} \times k}$.
  - The linear transform $\mathcal{R}_\phi$, which is realised as a weight matrix in $^{k \times d_{v_0} \times d_{v_0}}$ is applied to $\mathcal{F}(v_0(x))$. The result is $\mathcal{R}_\phi \cdot \mathcal{F}(v_0(x)) \in\ ^{d_{v_0} \times k}$.
  - Apply the inverse FFT to $\mathcal{R}_\phi \cdot \mathcal{F}(v_0(x))$, (appended with zeros to make up for the truncated/unselected modes). The outcome is $\mathcal{F}^{-1}(\mathcal{R}_\phi \cdot \mathcal{F}(v_0(x))) \in\ ^{N \times M \times d_{v_0}}$
  - Finally, $v_1(x)$ is obtained from the above together with a residual connection with a weight matrix $W_\Phi \in\ ^{d_{v_0} \times d_{v_0}}$, as follows

$$v_1(x) = \sigma(W_\Phi \cdot v_0(x) + \mathcal{F}^{-1}(\mathcal{R}_\phi \cdot \mathcal{F}(v_0(x)))) \in\ ^{N \times M \times d_{v_0}}$$

- $v_2, v_3, \ldots, v_T$ are obtained from $v_1, v_2, \ldots, v_{T-1}$ respectively in a similar manner as above. In this way, we have T Fourier layers.

- At the end of the last Fourier layer another FCN Q_ψ is applied, which projects $v_T(x)$ to the output dimension. We therefore obtain the output function

$$u(x) = Q_\psi(v_T(x)) \in \mathbb{R}^{N \times M}$$

In the FNO architecture, we use the Gaussian Error Linear Units (GELUs) [134, 135] as activation functions.

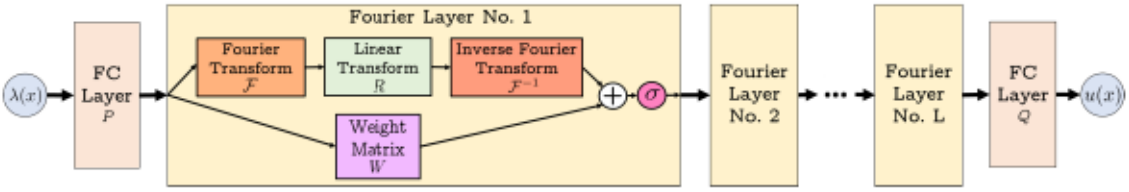


Figure 3.7: Architecture of the FNO.

Algorithm 6: FNO

Input:
- (λ_i, u_i): Training data pair with $i = 0 \ldots N_{\text{train}}$
- τ: learning rate

Initialise the fully connected networks (FCN) P_θ and Q_ψ
Initialise the linear transforms W_Φ and R_ϕ
`/* Section 3.5.2 defines these FCN and linear transform networks */`
Lift λ_i to a higher dimension by computing $v_0 = P_\theta(\lambda_i)$
for $j \leftarrow 0, \ldots, T-1$ **do**
  $v_{j+1} := \sigma\left(W_\Phi v_j + [\mathcal{F}^{-1}\left(R_\phi \cdot (\mathcal{F} v_j)\right)]\right)$, output of Fourier layers
end
Obtain the target dimension by using Q_ψ to get $\bar{u}_i = Q_\psi(v_T)$
while not converged **do**
  `/* Train network */`
  for $i \leftarrow 0, \ldots, N_{\text{train}}$ **do**
    Compute loss $L_i = \left\| \frac{\bar{u}_i - u_i}{u_i} \right\|_2$
  end
  Compute the Loss, $L = \sum_{i=1}^{N_{\text{train}}} L_i$
  Optimise L using the appropriate/chosen optimisation algorithm.
  Update $\Theta \leftarrow \Theta - \tau \nabla_\Theta L$ where $\Theta = (\theta, \psi)$.
end

Generalisation and Extensions of FNOs

A detailed guide for implementation and an illustration of how to extend FNO to operators with inputs and outputs defined on different domains can be found in [224]. There, two cases of generalisation are also considered:

C 1: As motivation, take a parabolic PDE, where the initial condition at time $t = 0$ is the input for the parameter-to-state operator. The output is the solution for all times $t \in [0, T]$. This leads to the general setting where the output space is that of functions defined on a product space, where the first component is the same as for the input space and the second component is arbitrary. i.e.

$$F : \Lambda(\Omega; \quad ^{d_\lambda}) \ni \lambda(x) = u(x, 0) \mapsto u(x, t) \in \mathcal{U}(\Omega \times [0, t]; \quad ^{d_u}),$$

where Ω is a subset of while Λ and $\mathcal{U}$ are Banach spaces taking values in $^{d_\lambda}$ and ^{d_u} (with $d_\lambda = d_u$, depending on the discretisation of the functions) respectively. In order to match dimensions before applying an FNO, the output domain could be shrunk by use of a Recurrent Neural Network (RNN) or the input domain could be extended to incorporate the time, t as an extra coordinate.

C 1: The input space is a subset of the output space, like in the case of the operator F, mapping the boundary conditions of the domain to the solution of the PDE in the whole domain.

$$F : \mathcal{G}(\partial\Omega; \quad ^{d_g}) \ni g(x) = u|_{\delta\Omega} \mapsto u(x) \in \mathcal{U}(\Omega; \quad ^{d^u}),$$

where $\partial\Omega \subset \Omega \subset$, while $\mathcal{G}$ and $\mathcal{U}$ are Banach spaces taking values in ^{d_g} and ^{d_u} (defined in a similar way as above in the first case) respectively. For this situation, zero padding could be applied or better still, an appropriate transformation could be used to map the boundary condition to a lower dimension. This then brings us back to the previous case.

It is worth noting that the use of the FFT in the algorithm of the FNO restricts its applicability to situations where the input and output are defined on a cartesian domain (square, rectangle domain in 2D and cube, cuboid in 3D). This introduces a challenge when dealing with complex geometries. A practical workaround is described as follows.

W 1: Define a new domain which is the minimum bounding box of the complex domain, and perform a zero padding for the area within this bounding box, but out of the complex domain. However, in practice, this makes the resulting function discontinuous in the 'rectangle' and usually yields larger errors.

W 1: Use the same minimum bounding of the complex domain, as previously described, but padding is done with the nearest function values within the complex domain, instead of zeros.

Though somewhat successful, the extensions provided in [224] do not seem to favour FNO. A better approach to complex domains is by transforming/deforming this complex domain to a cartesian domain by use of neural networks (NN), positioned before and after

the FNO as proposed in [210]. These neural networks could either be fixed or learned together with the FNO parameters. The resulting NN-FNO-NN network is said to be 'geometry aware' and is called Geometry-Aware FNO (Geo-FNO).

Besides these structural considerations, the following generalisations have recently gained increasing attention:

From conv-FNO to U-FNO: Possibly the most recent variant of the FNO, the U-FNO [316] introduces the famous U-Net to the architecture in the so-called U-Fourier layer. The U-FNO thus has both Fourier and U-net layers; it starts off with the former and ends with the latter. Essentially, a U-Fourier layer is similar to the Fourier layer, but has both a weight matrix and a two-step U-Net layer in the residual as illustrated in Figure 3.8. In contrast to the update performed by the Fourier layer in Equation 3.20, the U-Fourier layer performs the update in Equation 3.23, with U being the two-step U-Net.

$$v_{t+1}(x) := \sigma\left(Uv_t(x) + Wv_t(x) + \left(\mathcal{K}(a;\theta)v_t\right)(x)\right), \quad \forall x \in \Omega. \tag{3.23}$$

As compared with methods based on CNNs, the baseline FNO achieves a good accuracy for single-phase flow problems, but it doesn't seem to do so well with multiphase flows [315]. On the other hand, conv-FNO does better than CNN-based methods, and the U-FNO even improves on that. Conv-FNO is implemented by using a standard convolution in place of the U-Net.

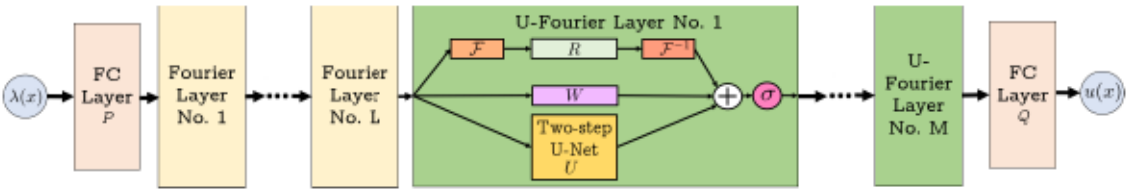


Figure 3.8: Architecture of the U-FNO.

Multiwavelet based operator (MWT): In a more recent work [119], a neural operator which evaluates the kernel operator in (3.21) using a different approach is introduced. The idea here is to leverage the successes in signal processes of both orthogonal polynomials and wavelet basis (notably, vanishing moments and orthogonality). Multiwavelets, therefore, do not just project the function onto a single wavelet function as wavelets do, instead they project the function onto a subspace of degree-restricted polynomials.

For an understanding of the concept, the space of piece-wise polynomials of degree up to $k \in \mathbb{N}$ with $n \in \mathbb{Z} \cup \{0\}$ is defined in $[0, 1]$:

$$\mathbf{V}_n^k = \bigcup_{l=0}^{2^n-1} \left\{ f \mid \deg(f) < k \text{ for } x \in \left(2^{-n}l, 2^{-n}(l+1)\right) \wedge 0, \text{ elsewhere} \right\}.$$

For each $k \in$ with $n \in \cup \{0\}$, we have that $\dim(\mathbf{V}_n^k) = 2^n k$ and

$$\mathbf{V}_{n-1}^k \subset \mathbf{V}_n^k. \tag{3.24}$$

We note that, as n increases, so does l, and $\mathbf{V}_n^k$ is defined from a lower(coarser) to a higher(finer) resolution; these resolutions being each time a power of 2. As a result, the method is restricted to discretisations which are a power of 2. Appropriate padding or interpolation could be applied to the function if its resolution is not a power of 2.

Given the basis $\phi_j, j = 0, 1 \ldots, k-1$ w.r.t, a measure μ_0 of $\mathbf{V}_0^k$, it is possible to obtain the basis of subsequent $\mathbf{V}_n^k, n > 0$ by appropriate shifts and scales of ϕ_j :

$$\phi_{jl}^n(x) = 2^{n/2}\phi_j\left(2^n x - l\right), \quad j = 0, 1, \ldots, k-1, \quad l = 0, 1, \ldots, 2^n - 1, \text{ w.r.t. } \mu_n \tag{3.25}$$

One then defines the multiwavelet subspace $\mathbf{W}_n^k$, which is related to the spaces of orthogonal polynomials as below:

$$\mathbf{V}_{n+1}^k = \mathbf{V}_n^k \bigoplus \mathbf{W}_n^k, \quad \mathbf{V}_n^k \perp \mathbf{W}_n^k. \tag{3.26}$$

Similarly, the basis of $\mathbf{W}_n^k, n > 0$ can be obtained by appropriate shifts and scales if the basis $\psi_j, j = 0, 1 \ldots, k-1$ w.r.t, a measure μ_0 of $\mathbf{W}_0^k$, is known. A similar expression as in Equation 3.25 can thus be obtained.

Equations 3.24 and 3.26 inform us that the basis of $\mathbf{V}_n^k$ and W_n^k can be written as linear combinations of that of $\mathbf{V}_{n+1}^k$. As a result, for a given function it is possible to obtain a relationship between its coefficients in these bases. Specifically, the function $f \in {}^d$ has coefficients in the space of orthogonal polynomials (multiscale coefficients) $\mathbf{s}_l^n = [\langle f, \phi_{il}^n \rangle_{\mu_n}]_{i=0}^{k-1} \in {}^{k^d \times 2^n}$ and coefficients in the multiwavelet subspace (multiwavelet coefficients) $\mathbf{d}_l^n = [\langle f, \psi_{il}^n \rangle_{\mu_n}]_{i=0}^{k-1} \in {}^{k^d \times 2^n}$, which are related by the *decomposition* equations:

$$\mathbf{s}_l^n = H^{(0)}\mathbf{s}_{2l}^{n+1} + H^{(1)}\mathbf{s}_{2l+1}^{n+1}, \tag{3.27}$$

$$\mathbf{d}_l^n = G^{(0)}\mathbf{s}_{2l}^{n+1} + G^{(1)}\mathbf{s}_{2l+1}^{n+1}, \tag{3.28}$$

and *reconstruction* equations

$$\mathbf{s}_{2l}^{n+1} = \Sigma^{(0)}(H^{(0)T}\mathbf{s}_l^n + G^{(0)T}\mathbf{d}_l^n), \tag{3.29}$$

$$\mathbf{s}_{2l+1}^{n+1} = \Sigma^{(1)}(H^{(1)T}\mathbf{s}_l^n + G^{(1)T}\mathbf{d}_l^n). \tag{3.30}$$

$H^{(0)}, H^{(1)}, G^{(0)}, G^{(1)} \in {}^{k^d \times k^d}$ are the reconstruction filters while $\Sigma^{(0)}, \Sigma^{(1)}$ are the correction terms. The theory and derivation of these terms are well covered in [119], we only highlight it here for better explanation of the steps involved in the method. In Equations 3.27-3.28, $H^{(0)}$ and $G^{(0)}$ act on the even terms of the multiscale coefficient while $H^{(1)}$ and $G^{(1)}$ act on the odd terms. The result of these operations is visibly a down-sampling from the higher to a lower resolution (half the original resolution), thus

the term decomposition. On the other hand, equations 3.29-3.30 perform the reverse of this operation, leading to the recovery of the higher resolution.

So far, we walked through multiwavelet representation of a function $f \in \mathbb{R}^d$. This notion will be applied to the input and output functions of the neural network/operator. A different notion, known as the Non-Standard Form is used to obtain the multiwavelet representation of the kernel function. This method essentially reduces the operator kernel in Equation 3.21 $\mathcal{K}v = w$ to the set of equations.

$$\begin{aligned} U_{dl}^n &= A_n d_l^n + B_n s_l^n, \\ U_{sl}^n &= C_n d_l^n, \\ U_{sl}^L &= \bar{T} s_l^L, \end{aligned}$$

with U_{dl} and U_{sl} (respectively d_l^n and s_l^n) being the respective multiscale and multiwavelet coefficients of w (respectively v). L here is the index corresponding to the coefficients of the lowest resolution (output of the last decomposition). A_n, B_n, C_n and $\bar{T}$ are then parameterised with a CNN (which could be done in Fourier space as in FNO or not) network followed by a ReLU activation then linear layer which we learn during training. In practice we use same neural networks for all n: thus $A_n = A_{\theta_A}, B_n := B_{\theta_B}, C_n := C_{\theta_C}$ and $\bar{T} := \bar{T}_{\theta_T}$. For a given input, the multiwavelet operator performs a series of operations as shown in Figure 3.9.

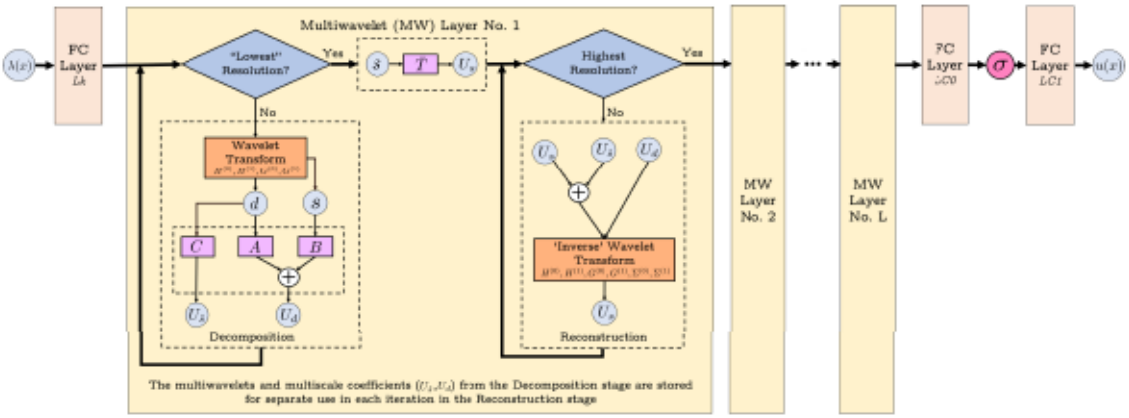


Figure 3.9: Architecture of the MWT Operator.

Physics Informed (Fourier) Neural Operator - PINO This method combines the operator learning of FNOs and function learning PINNs, in a quest to alleviate the challenges faced in the individual cases [214]. While PINNs face a challenge in optimisation when dealing with complex problems as in multiscale dynamics, FNO needs a wealth of data which might not be readily available. PINO combines both concepts by introducing two stages in the learning process:

- Firstly, the operator is learned using either/both a **data loss** (same loss as in FNO) and/or a **PDE loss** (physics-informed loss), in a phase termed Pre-training. This case reduces to the normal FNO method if no PDE loss is used. As an example, the PDE loss for Poisson's equation in 3.15 is given by Equation 3.31 where α is a hyperparameter.

$$L_{\text{PDE}} = \underbrace{\sum_{i=1}^{N_{\text{train}}} \|\Delta u_{i\Theta} - \lambda_{i\Theta}\|^2_{L^2(\Omega)}}_{\text{Domain } (\Omega) \text{ loss}} + \underbrace{\alpha \sum_{i=1}^{N_{\text{train}}} \|u_{i\Theta} - g_i\|^2_{L^2(\partial\Omega)}}_{\text{Boundary } (\partial\Omega) \text{ loss}}, \tag{3.31}$$

- Secondly, for a specific instance of the input (parameter, for forward problem), the learned operator from the first stage is further fine-tuned, using the **PDE loss** and an **operator loss**. The latter minimises the difference between the initial network, F_{Θ_0} resulting from the pre-training phase and the further optimised networks $F_{\Theta_i}, i > 0$. This phase is the test-time optimisation stage. For a specific parameter instance λ, the operator Loss is thus given by

$$L_{\text{OP}} = \|F_{\Theta_i} - F_{\Theta_0}\|^2. \tag{3.32}$$

A strength of the PINO is in its ability to achieve competitive errors with fewer data as demonstrated in [214]. This is mainly due to the introduction of the PDE loss. Notably, this PDE loss is evaluated in a not-so-usual way as the input of the network is the function and not a set of collocation points. Automatic differentiation as we know it is thus not possible. Three methods for gradient descent are outlined in [214]. One option is numerical differentiation, i.e., a finite difference method (FDM), which we use for our experiments in later sections.

FNO usage for inverse problems

FNO is an operator learning concept, hence both general concepts as outlined in Section 3.2.3 can be used for solving inverse problems. We will report on the achieved results in our section on numerical examples. FNO does remarkably well, despite its rather linear and Fourier-centric approach. However, it allows the incorporation of fine details related to high frequencies in a consistent way, nevertheless understanding the success of FNO methods for solving inverse problems should be an important direction for future research.

3.5.3 DeepONet

Motivation

It is widely known that deep neural networks are universal approximators, i.e., they can approximate any finite-dimensional function to arbitrary accuracy [65, 145]. This has been extended to a universal approximation theorem for operators [47], which states that a neural network with a single hidden layer can accurately approximate any nonlinear continuous operator. This theorem and its extension to multi-layer networks, see [222], provides the motivation for the concept of DeepONet.

Network Architecture

The DeepONet [222] mirrors the structure of the universal approximation theorem of operators with a novel network architecture. Let us consider an operator $\mathcal{F}$ that maps an input function λ to an output function u, i.e., $u = \mathcal{F}(\lambda)$. A DeepONet G_Θ takes an input function λ, which is sampled at fixed predefined collocation points, and provides an approximation for $u(x)$ at arbitrary points x by the combination of a trunk and branch net in the following way:

$$u(x) \approx G_\Theta(\lambda)(x) = \sum_{k=1}^{p} \underbrace{b_k(\lambda(s_1), \lambda(s_2), ..., \lambda(s_m))}_{\text{branch}} \underbrace{t_k(x)}_{\text{trunk}} = \sum_{k=1}^{p} b_k(\lambda) t_k(x). \tag{3.33}$$

The trunk net $\mathbf{t} = \mathbf{T}(x; \theta_t)$ takes the continuous coordinates x as the input, and outputs a feature vector $\mathbf{t} = [t_1, ..., t_p] \in \quad ^p$, which can be considered as p functions of x. The branch net $\mathbf{b} = \mathbf{B}(\lambda; \Theta_b)$ takes $\lambda = [\lambda(s_1), \lambda(s_2), ..., \lambda(s_m)]$, the discretisation of the input function, as input and returns a feature vector $\mathbf{b} = [b_1, ..., b_p] \in \quad ^p$ as output. The branch and trunk nets are then combined by an inner product to approximate the underlying operator.

This DeepONet is called "unstacked DeepONet", while it is called a "stacked DeepONet" if each $b_k(\lambda)$, $k = 1, ..., p$ is an individual neural network, i.e., $[b_1, ..., b_m] = [\mathbf{B}(\lambda; \Theta_b^1), ..., \mathbf{B}(\lambda; \Theta_b^m)]$. Several numerical results have shown that unstacked DeepONets typically have larger training errors as compared with stacked DeepONets, but the test error is smaller and unstacked DeepONets lead to smaller generalisation errors. Both branch net and trunk net can have general architectures, e.g., fully connected neural network(FCN), recurrent neural network(RNN), and convolutional neural network(CNN). However, as x is usually in low dimensional space, a standard FNN is commonly used as the trunk net. A bias can also be added to the last stage of the DeepONet to improve performance.

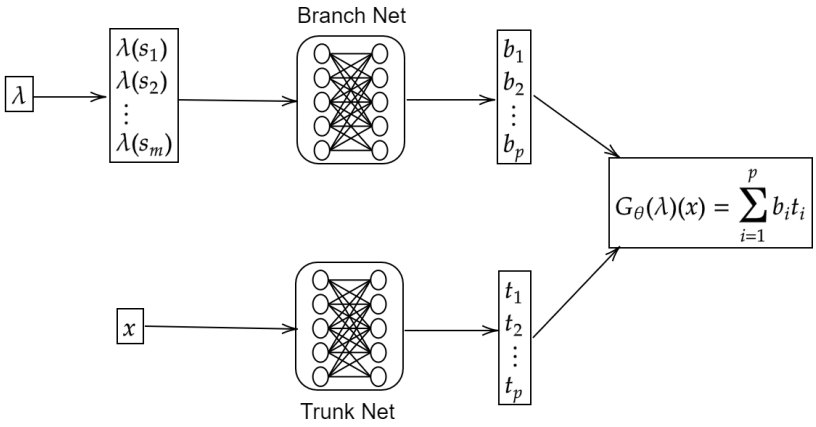


Figure 3.10: Architecture of the DeepONet

The novelty of the DeepONet is that its network architecture is composed of these two sub-networks, which treat the input λ and x differently, thus it is consistent with prior knowledge. In addition, there is no need to discretise the domain to approximate

the solution space, and the evaluated solution functions are defined in the whole domain. The only condition required is that the sensor locations $[s_1, s_2, ..., s_m]$ should be consistent for the training dataset. There also are some generalisations which encode the input functions to the branch net by a feature vector, for example, we can use the coefficients of λ with respect to some chosen basis as input.

Once the DeepONet is trained, it is easy to see that the numerical solutions will lie in the linear space $\text{Span}\{t_1(x), ..., t_p(x)\}$, i.e., $\{t_k(x)\}_{k=1}^p$ are actually the trained basis to approximate the solution space, and $\{b_k(\lambda)\}_{k=1}^p$ are the corresponding coefficients. Since usually, p is not large, e.g., approximately 100 for our 2 dimensional problems, the DeepONet can be regarded as a model reduction method in which the reduced basis is obtained by training.

The Algorithm

Based on the architecture of the DeepONet, the parameters can be optimised by minimising the following mean square error loss:

$$\begin{aligned} L(\Theta) &= \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\left|\mathcal{F}\left(\lambda_i\right)\left(x_{i,j}\right) - G_\Theta\left(\lambda_i\right)\left(x_{i,j}\right)\right|^2 \\ &= \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\left|u_i\left(x_{i,j}\right) - \sum_{k=1}^{p} b_k\left(\lambda_i\right) t_k\left(x_{i,j}\right)\right|^2 \end{aligned}$$

There are two important hyper-parameters which should be determined before training: the positive integers m and p, i.e., the number of sensors for encoding the input functions and the number of the bases used to approximate solutions. Larger m and p means smaller encoding error and reconstruction error, respectively. However, increasing them usually does not necessarily reduce the total error due to the increased complexity of the optimisation problem. The locations of the sensors are not necessarily equispaced. However, they should be consistent with the training dataset.

Note that a pair of the training data in DeepONet is $\{\lambda_i, x_{i,j}, u_i(x_{i,j})\}$, thus it is slightly different from most neural operator methods, i.e., the full field observation of solutions is not necessary for the DeepONet and the DeepONet can work with only partially observed solutions. This is of particular importance for real applications where more often than not the available data is not complete. In addition, for different types of data sets, we can use different implementations for DeepONet to dramatically reduce the computational cost and memory usage by orders of magnitudes. For example, different branch input λ may share the same trunk net input x, and different input x may share the same input λ, see [224] for details. This computation technique can also be applied to the extension PI-DeepONet [310] when computing the derivatives of the output functions.

Algorithm 7: Fast implementation of DeepONet

Input: data: $\{(\lambda_i, x_j, u_i(x_j))|i = 1, 2, ..., N; j = 1, 2, ..., M\}$
Number of sensors: m, and the locations of the sensors: $S = (s_1, ..., s_m)$
Let $\lambda = [\lambda_1(S), ..., \lambda_N(S)] \in {}^{N\times m}; X = [x_1, ..., x_M] \in R^M, U = [u_1, u_2, ..., u_M] \in {}^{N\times M}$
p: the number of basis to be learned.
τ_b/τ_t : learning rates of the branch net and trunk net

Initialise the branch net B_{Θ_b} and trunk net T_{Θ_t}
while not converged **do**
  $B = B_{\Theta_b}(\lambda) \in {}^{N\times p}$ and $T = T_{\Theta_t}(X) \in {}^{M\times p}$
  Output $= BT^T \in {}^{N\times M}$
  $L(\Theta) = \frac{1}{MN}\|U - \text{Output}\|$
  Update $\Theta \leftarrow \Theta - \tau\nabla_\Theta L$
  `/* The fast algorithm can be easily extended to the mini-batch case and the computation of the derivatives in PI-DeepONet. */`
end

Generalisations of DeepONet

Several extensions of DeepONet have also been developed.

- In [224], a feature expansion for the trunk net input is proposed in order to satisfy some desirable properties of the output function, e.g., oscillating structures or decay properties. Feature expansion for the branch net can also be used to incorporate a feature which is a function of x. E.g., the POD-DeepONet, as proposed in [224], precomputes a basis by performing proper orthogonal decomposition (POD) on the training data. Thus, POD-DeepONet shares the same idea with PCANN for the output space. Employing such feature vectors might be particularly advantageous for non-smooth parameters or solutions, i.e., discontinuous or highly oscillatory functions.

- The Bayesian B-DeepONet [216] uses the Bayesian framework of replica-exchange Langevin diffusion to enable DeepONets training with noisy data. The B-DeepONet and Prob-DeepONet, as proposed in [238], were shown to have good predictive power along with uncertainty quantification. In [96], a further generalization 'Variational Bayes DeepONet' was introduced, in which the weights and biases of the neural network are treated as probability distributions instead of point estimates, and their prior distributions are updated by Bayesian inference.

- A universal approximation theorem of continuous multiple-input operators was proved and the corresponding MIONet was proposed to learn multiple-input operators in [170]. Similarly, the authors in [298] proposed a new enhanced DeepONet, in which multiple input functions are represented by multiple branch DNN sub-networks, which are then combined with an output trunk network via inner products to generate the output of the whole neural network.

- Several extensions for dealing with particular function properties were recently proposed. E.g., a multi-scale DeepONet [220] was proposed to approximate a nonlinear operator between Banach spaces of highly oscillatory continuous functions. The shift-DeepONets proposed in [124] extends Deep Operator Networks for discontinuous output functions by elevating the linear basis expansion of the classical DeepONet architecture to a non-linear combination. This can make the basis functions themselves dependent on the input function by exposing explicit shift and scale parameters. In the paper [294], the authors introduce the Deep Graph Operator Network, a combination of DeepONet and Graph neural networks, by using GNN Branch Net to exploit spatially correlated graph information. The authors in [237] developed a framework named Fed-DeepONet to allow multiple clients to train DeepONets collaboratively under the coordination of a centralised server.
- The Variable-input Deep Operator Network (VIDON) [258], is peculiar in that it allows for sensor points to be queried from any point in the domain during training. In this way, no prior discretisation of the domain is needed.

Theoretical Background

In addition to the universal theorem, several analytic results have been published for DeepONet. In the original paper, a theoretical analysis was presented which allows estimation of the approximation properties for ODE operators with respect to an underlying probability distribution. The analysis depends on the number of sensors and the related approximation of the input functions.

In [197], the universal approximation property of DeepONets was extended to include measurable mappings in non-compact spaces. By decomposition of the error into encoding, approximation and reconstruction errors, both lower and upper bounds on the total error were derived, relating it to the spectral decay properties of the covariance operators associated with the underlying measures. For four prototypical examples of nonlinear operators, it was proved that DeepONets can break the curse of dimensionality.

In [70], the convergence rates of the DeepONet were considered for both linear and non-linear advection-diffusion equations with or without reaction. The conclusion is that the convergence rates depend on the architecture of the branch network as well as on the smoothness of the inputs and outputs of the operators. The paper [104] gives a bound on the Rademacher complexity for a large class of DeepONets.

3.6 Conclusion

This chapter outlined some major methods in the context of deep learning for PDEs, and how these methods can be extended for the respective parameter identification tasks. In the next chapter, these methods, and the extensions are evaluated on both basic and complex problems, as well as both linear and non-linear problems.

Mathematics is, in its way, the poetry of logical ideas.

— Albert Einstein

4

Deep Learning for Partial Differential Equations 2: Numerical Results

This chapter presents the results of DL methods for PDEs, and their extensions in parametric studies and inverse problems. It is based on the following articles:

> Derick Nganyu Tanyu, Jianfeng Ning, Tom Freudenberg, Nick Heilenkötter, Andreas Rademacher, Uwe Iben and Peter Maass. "Deep learning methods for partial differential equations and related parameter identification problems". In: *Inverse Problems 39.10 (Aug. 2023), p. 103001.*
>
> Derick Nganyu Tanyu. "From Neural Operators to Complex Partial Differential Equations based Inverse Problems: Comparative Numerical Methods for Problem-Solving". In: *In preparation. Preprint to be made available on arXiv.*

4.1 Introduction

The core of this chapter is a numerical comparison of the methods described in chapter 3. The aim is to develop a guideline for scientists who want to start working with DL methods for PDE-based problems and who face the problem of determining suitable methods.

We will perform numerical experiments on different levels. Firstly, we will investigate the performance for solving the forward Poisson problem. This linear problem is most commonly used and, despite its limited value for generalisations to non-linear problems - already yields some insight into the performance of the chosen methods. We then perform tests for the inverse Poisson problem, where the source term is the sought-after parameter.

After that, we will turn to analysing the behaviour of the chosen methods for forward and inverse Darcy flows. A particular emphasis is on evaluating and comparing the respective numerical schemes in terms of accuracy, but also computational time and storage needed for parametric studies.

With the insight gained from studying these two rather *basic* problems, we proceed to investigate more *complex* problems, where we equally show that the results obtained for these methods are comparable.

We want to acknowledge that a growing list of publications devoted to testing DL concepts for the solution of PDEs already exists, see [105, 112, 224]. However, these studies focus on PDE-based forward problems; in contrast, our experiments for comparing forward solvers are a preliminary step towards our main goal, namely a comparison of DL methods for PDE-based parameter identification problems.

4.2 Part 1: Basic Problems

We start this section, focused on the Poisson and Darcy flow problems, with an outline of our data generation procedure, which follows [24, 212, 213]. Moving on, for both the Poisson and Darcy flow problems, we proceed as follows:

- Forward problem
- Inverse problem trained with exact data
- Inverse problem trained with noisy data

4.2.1 Generating Training and Test Data

The outline of our numerical experiments for the linear differential equations $\Delta u = \lambda$ on Ω, $u = g$ on $\partial\Omega$ (Poisson problem), as well as simulations for the Darcy flow, follows the approach of [24]. Hence, we run a performance analysis in terms of accuracy, as well as computational load, and the parameter λ is randomly generated as a Gaussian random field. As a gold standard for comparison, we use a standard FDM code, which is tuned to high precision and also provides the ground truth data for training the networks.

Learning operators implies learning mappings between function spaces. For a supervised learning task, we need input-output data pairs for training, which in our case are the parameter and solution functions. We start with Gaussian Random Fields, commonly used in the stochastic modelling of physical phenomena as described in [24, 212, 213, 224]; more precisely, we use the Gaussian distribution as base measure

$$\mu_{\mathrm{G}} = \mathcal{N}\left(0, (-\Delta + 9I)^{-2}\right),$$

with a zero Neumann boundary condition on the operator Δ, which yields random but smooth test data for the Poisson problem. Other interesting measures are μ_{L} and μ_{P}

which are the push-forwards of μ_{G} under the exponential and piece-wise constant maps respectively, so that $\mu_{\mathrm{L}} = \exp_\sharp \mu_{\mathrm{G}}$ and $\mu_{\mathrm{P}} = T_\sharp \mu_{\mathrm{G}}$, where $\exp_\sharp$ and $T_\sharp$ represent the respective push-forward functions, with

$$T(s) = \begin{cases} 12 & s \geq 0 \\ 3 & s < 0 \end{cases}$$

For our experiments with the Darcy flow, we used piece-wise constant parameters. Some examples are shown in Figure 4.1.

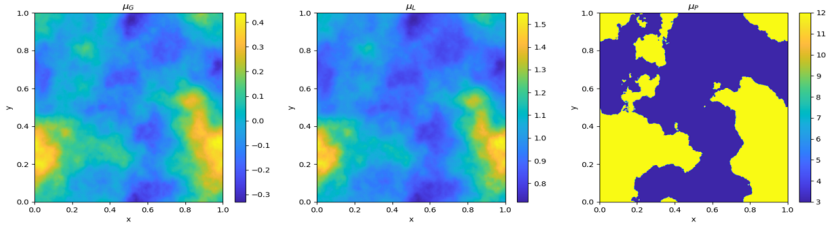


Figure 4.1: Example of samples from the GRF probability measures μ_G, μ_{L} and μ_{P}.

To complete the data pair, we use the generated samples as right-hand sides of the Poisson problem and use a finite difference method (FDM) to get the solution for the considered problem. As an example, we consider the Poisson problem given by

$$\begin{aligned} -\Delta u(s) &= \lambda(s) \quad s \in (0,1)^2 \\ u(s) &= 0 \qquad s \in \partial(0,1)^2, \end{aligned} \tag{4.1}$$

where the forcing term $\lambda(s)$ is sampled from the GRF $\lambda \sim \mu_{\mathrm{G}}$. Then, the domain $(0,1)^2$ is discretised as shown in Figure 4.2, and a second-order FDM scheme is used to evaluate the Laplacian, reducing the Poisson equation to a system of linear equations given by Equation 4.2, where $i = 0, \ldots N_y$ and $j = 0, \ldots N_x$ for a resolution of $(N_x + 1) \times (N_y + 1)$

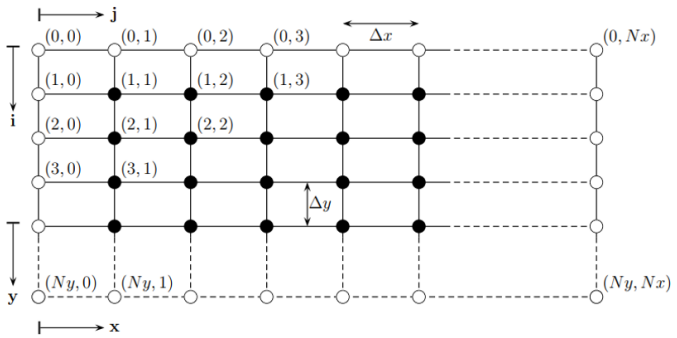


Figure 4.2: Computational grid showing interior grid points (black) and boundary grid points (white).

$$\frac{\Delta y}{\Delta x}\left(-u_{i,j-1} + 2u_{i,j} - u_{i,j+1}\right) + \frac{\Delta x}{\Delta y}\left(-u_{i-1,j} + 2u_{i,j} - u_{i+1,j}\right) = \Delta x \Delta y \lambda\left(x_j, y_i\right) \tag{4.2}$$

A similar discretisation scheme is used for the Darcy Flow equation 4.3 which is equally of interest to us in this work.

$$\begin{aligned} -\nabla \cdot (\lambda(s)\nabla u(s)) &= f(s) \quad s \in (0,1)^2 \\ u(s) &= 0 \qquad\quad s \in \partial(0,1)^2 \end{aligned} \tag{4.3}$$

Our numerical tests are done using baseline implementations of the mentioned DL algorithms, and we did some extensive hyperparameter search for every method. For details, see the appendices in [244]. Nevertheless, hyperparameter search and fine-tuning the optimisation scheme in search of global minima is a never-ending story. Hence, the presented results should be understood as the best we could achieve within the given time and with the available computing capacity (specified in [244]). All methods were treated equally, and we believe that this leads to a fair basis for comparison.

In addition, we are well aware of the extensive body of literature dealing with sometimes rather refined extensions and improvements of these methods, e.g., there are several extensions of the PINN approach and a survey on different Physics-informed neural operator networks variants has been recently published, [105]. However, these extensions most often come with the necessity to fine-tune additional hyperparameters, or they only apply to special cases. Hence, as said before, we focus on a comparison of the basic implementations in this section. To give proper credit to the different approaches, we assigned different co-authors to different concepts and everybody did their best to 'defend' the respective concepts.

So far, we have addressed the general outline of our testing scenario and our data generation for the academic test examples. This is in line with procedures used by our collaborating industrial partners. Research and development departments in the industry also seem to rely mostly on simulated data during the first development cycles for new products. In contrast to simulated data, real-life data for industrial applications typically is very scarce. Almost often, this is only available in very limited numbers. This leads to the problem of data enrichment and data augmentation, which is outside the scope of the present work.

We also clarify the criteria for our evaluation below. There are several obvious categories such as achievable mean accuracy, training time, and time for solving the forward Poisson problem after training, but also degrees of freedom of the networks used, stability concerning hyperparameter tuning, etc. Nevertheless, in the following, we will mainly discuss two criteria, namely accuracy and the potential for parametric studies/inverse problems, i.e., the computational time needed to run the algorithm after training for novel sets of parameters.

4.2.2 Poisson Problem

Forward Problem

Testing different DL concepts to train a **forward solver** for the Poisson problem (4.1) is the most basic academic example, which is used by almost all relevant publications in this

context. This linear problem can be solved by all other methods considered, and the errors are as reported in Table 4.1. These errors do not differ too much, they are below 1% relative error except for DRM, which has an error of approximately $2\% \sim 5\%$. Still, there are some notable differences. The overall winner is PCALin both in terms of achievable accuracy and in terms of complexity of the network. However, none of the methods was able to achieve an accuracy comparable with fine-tuned FDM or FEM methods. The advantage of DL concepts, in this case, lies in the potential for large-scale parametric studies, where the execution time for testing novel parameters has to be minimal.

However, not all DL concepts are suitable for parametric studies. In general, DL concepts aiming for a function approximation require expensive retraining of the network for every additional parameter and have limited value for large-scale parameter variations. Hence, we focus on DL concepts for operator approximations, see Table 3.1. In our experiments, DeepONet and their physics-informed versions performed best in terms of efficiency (run time for testing). The differences in testing time for these methods are below the variance introduced by the random sampling of test data.

The comparison with respect to different resolutions of u shows the expected, but interesting result, that operator approximations, which are based on a functional analytic reasoning in function spaces rather than discrete settings, indeed show an independent accuracy across scales, see Table 4.2. This cannot be matched by function approximations, where the error increases by several orders of magnitude for coarse discretisations.

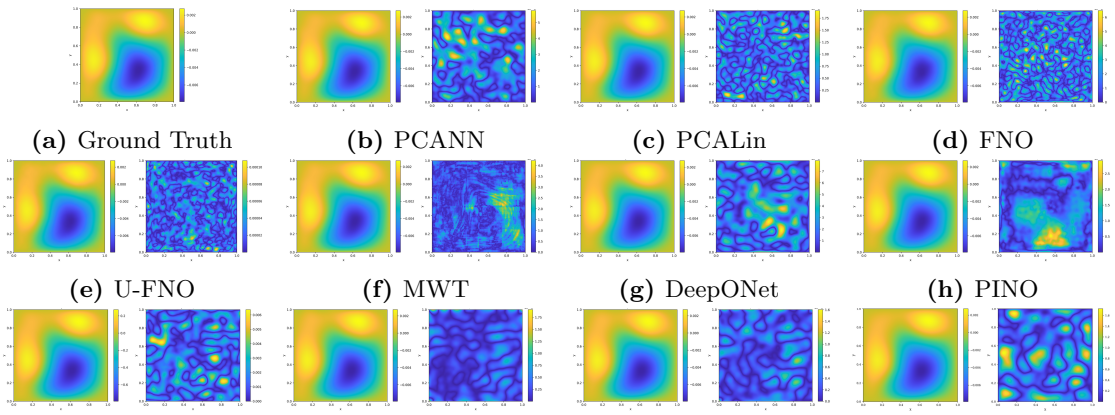
These numerical findings for the linear Poisson problem in a standard domain have little value for generalisations to other PDE problems. Nevertheless, as a punch line for simple linear PDE problems, we would stress the old saying 'keep it simple' and suggest applying 'easy to use' concepts such as PCALin with a suitable, but comparatively small network.

PINNs also offer an easily accessible concept, which is easy to adapt to other PDE, but it requires retraining, i.e., PINNs in their original version are not suitable for parametric studies, and they exhibited a slightly larger L_2-error in our scenario.

As an alternative, MWT requires a more advanced implementation but seems to be more efficient for parametric studies, however, training times are rather high.

Inverse Problem

The investigation of the performance of DL concepts for PDE-based inverse problems for parameter identification is the core of this survey. In this subsection, we summarise the results for the inverse Poisson problem, i.e., determining λ in Equation 3.15 from a measured version of u. Iterative methods for solving such inverse problems, as well as parametric studies, require multiple evaluations of the parameter-to-state map F. As already mentioned, DL concepts based on operator approximations are better suited in this context, and we will focus on these methods. The only exception is the extension to inverse problems of PINNs and QRES, as described in Section 8

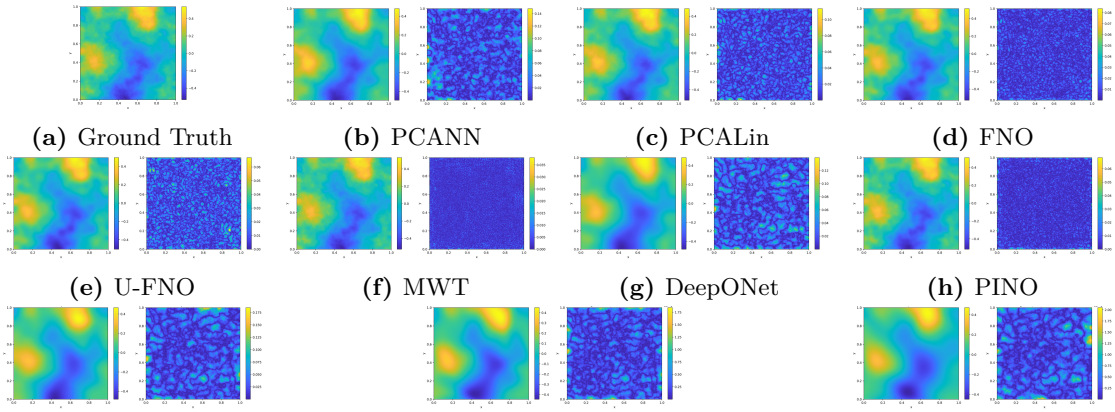


(a) Ground Truth **(b)** PCANN **(c)** PCALin **(d)** FNO

(e) U-FNO **(f)** MWT **(g)** DeepONet **(h)** PINO

(i) PI-Deeponet **(j)** PINN **(k)** QRES **(l)** DRM

Figure 4.3: Test examples for Poisson forward problem using resolution of 513×513. (b)-(l) show the specified neural network's approximation of the solution (left-hand side) and the absolute difference between the Ground truth in (a) with the approximation (right-hand side).



(a) Ground Truth **(b)** PCANN **(c)** PCALin **(d)** FNO

(e) U-FNO **(f)** MWT **(g)** DeepONet **(h)** PINO

(i) PI-Deeponet **(j)** PINN **(k)** QRES

Figure 4.4: Test examples with backward operator training for inverse Poisson problem using a resolution of 513×513. (b)-(k) show the specified neural network's approximation of the solution (left-hand side) and the absolute difference between the Ground Truth in (a) with the approximation (right-hand side). All examples are computed with noise free data.

Our tests for PDE-based inverse problems are organised in terms of how we attack the inverse problem and which data is used for training. Foremost, there are two primary approaches for dealing with inverse problems, see Section 3.2.3. We can either train the inverse problem with a reversed input-output structure, or we can integrate a learned forward solver in a Tikhonov approach. Secondly, we can train the network with either noiseless data or noisy data with different noise levels. Both tests are meaningful for applications, depending on whether clean data obtained in research labs or measured data from field experiments are available. As it is common for inverse problems, these networks will be evaluated for reconstruction problems with noisy data after training. The case of testing with noise-free data is included for the reason of completeness.

Training is always done with 1000 training samples, where λ is computed as a smoothed

random Gaussian field, see Section 4.2.1. The parameter-to-state map, $u = F(\lambda)$, is then computed with a high-precision finite difference scheme. The resulting solution u is then perturbed with normally distributed random noise of different levels, δ.

$$u^\delta(x) = u(x) + \delta \cdot ||u|| \cdot N(0,1)$$

For the evaluation of the methods, we use additional $N = 5000$ samples of u^δ, which are computed by the same procedure, i.e., drawing additional random samples for λ_{true}, computing the corresponding solution u and adding noise.

These perturbed solutions are the input for the inverse problem during evaluation. The resulting estimation of the parameter $\hat{\lambda}$ is then compared with the original, true parameter λ_{true}. As a standard measure of success, we average over the different evaluation samples and take the mean L_2-error, $E(\|\hat{\lambda} - \lambda_{true}\|)$.

For certain applications, i.e., control problems, the output error is also of importance and, for some experiments, we also report the difference between the solution obtained with $\hat{\lambda}$ and the true, unperturbed solution u as shown in Tables 4.7b and 4.7c.

Let us comment on the different test scenarios (inverse learning or Tikhonov, noise-free or noisy data for training) in more detail.

The tests with unperturbed data, i.e., the case where the networks were trained and evaluated with perfect noise-free data are reported in Table 4.1 and Table 4.2 for different resolutions. For operator approximation methods, the inverse problem in these tables is always solved by inverse learning (backward operator training). Also, PINN and QRES can be directly extended to parameter learning by a doubling of the network, see Section 8. This results in a doubling of the network parameters as shown in the second column of both tables. As a general observation, we remark that the achievable accuracy for the inverse problem is considerably below the accuracy of the forward problem, which reflects the ill-posedness of the inverse problem. Also, the baseline implementations of PCANN, PINN and DeepONet perform less reliable as compared with their extensions, such as PCALin, U-FNO, PINO or MWT. After training, the run times of the different methods are the same as for the forward problem.

Due to the linear nature of the problem and the missing noise in the data, one should not overestimate the value of these tests. In particular, testing with noisy data is essential for inverse problems.

To this end, we have done experiments where the networks were trained with noiseless data but the evaluation was done with noisy data, see Table 4.5. We clearly see that the networks trained with noise-free data do not generalise to the case of noisy data. The error exceeds 100% in most cases. The only notable exceptions are surprisingly DeepONet and PI-DeepOnet. Even for high noise levels, these methods still produce errors in the reconstructed parameter which are approx. 4 – times larger than the noise in the data, which is the range of errors one would expect for optimal regularisation schemes for ill-posed problems.

Tables 4.6 and 4.7 are the most meaningful ones for inverse problems. Here, we report results on networks, which are trained and evaluated with noisy data. Table 4.6 reports results obtained by reversed/backward operator training, i.e., the network is trained to take u^δ as input and to directly return an estimate for the parameter λ. Table 4.7 reports results obtained by first training a forward network $\Phi_\Theta(\lambda)$ and then computing an approximation to the parameter by solving a Tikhonov minimisation problem

$$\min_\lambda \ \|\Phi_\Theta(\lambda) - u^\delta\|^2 + \alpha R(\lambda) \ .$$

In these experiments, R was always taken as a discretisation of the L_2-norm and α was optimised by numerical experiments. Depending on the problem, one could add an additional regularisation term as shown in [244].

Both approaches, i.e., inverse learning and embedding a learned forward operator into a Tikhonov scheme do give comparable results, see Tables 4.6 and 4.7. All errors in the reconstructions are naturally larger than the error in the input data, as is to be expected for ill-posed problems. Given the small differences achieved by the different methods, it is challenging to suggest a particular method. However, in our opinion, the range of errors 1% or 5% might be most important for applications and a closer look at the numbers reported in Table 4.6 for inverse training of the inverse Poisson problem shows that FNO and PINO seem to perform best, with PCALin and MWT as runner-ups. For Tikhonov-based training and giving higher importance to larger error rates of 1% or 5%, best results are achieved by MWT with PINO as runner-up. Hence comparing the winners of either approach (FNO/PINO and MWT) for this range of errors yields a slight advantage for using the Tikhonov approach in connection with the MWT concept. However, we should remark that the Tikhonov approach requires solving a minimisation problem for each new data set and is computationally more expensive.

Finally, Tables 4.7b and 4.7c are based on experiments, where the reconstructed parameter $\hat{\lambda}$ was used to compute the corresponding solution $\hat{u} = F(\hat{\lambda})$. This is then compared with either the true solution u, see Table 4.7c), or the noisy data, which was used for training, see Table 4.7b). This is considered as a test for how well these parameter estimation problems can be used to solve control problems, which is however not the focus of this work and this case is therefore not investigated further.

Summary Poisson problem

In summary, for the forward problem, the investigated DL concepts are less accurate as compared with finite difference methods on a fine grid. However, these DL concepts do produce competitive results for solving inverse problems with different noise levels. DL concepts for operator approximation perform best and offer a significant advantage in run time. Hence, either large-scale parametric studies or iterative solvers for parameter identification benefit decisively from well-trained DL concepts.

Networks	# of Parameters	Forward Problem Rel. L2 Error	Forward Problem Training (s/epoch)	Forward Problem Testing (s)	Inverse Problem Rel. L2 Error	Inverse Problem Training (s/epoch)	Inverse Problem Testing (s)
DRM	6,721	0.0251	0.0514	~ 480	-	-	-
PINN	5,301 | 10,602	0.0075	0.1034	$\sim 1,315$	0.1650	0.2368	$\sim 3,370$
QRES	5,509 | 11,018	0.0076	0.1581	$\sim 2,150$	0.1549	0.4186	$\sim 5,730$
PCANN	5,155,150 | 5,205,200	0.0073	0.0142	0.6063	0.0981	0.0161	0.6179
PCALin	62,750	**0.0013**	0.0077	0.7453	**0.0244**	0.0090	0.7640
FNO	2,368,001	0.0066	42.6249	0.0168	0.0415	42.6245	0.0174
U-FNO	3,990,401	0.0053	97.6730	0.0346	0.0254	97.6434	0.0343
MWT	9,807,873	0.0036	114.2043	0.0488	**0.0159**	113.2655	0.0483
DeepONet	640,256 | 768,128	0.0042	0.1098	0.0002	0.1035	0.1201	0.0002
PINO	2,368,001	**0.0031**	42.6289	0.0166	0.0301	42.8172	!0.0143
PI-DeepONet	640,256 | 739,712	0.0061	0.4637	0.0002	0.1068	0.4462	0.0002

Table 4.1: Performance of different methods for the Poisson Problem, using a 513×513 resolution. In the second column, for the forward and inverse problems, if the same number of parameters are used, only one number is specified. If different amounts are used, two numbers are given. The left one then corresponds to the forward problem and the right one to the inverse case. The networks were trained with noiseless data and backward operator training was used for solving the inverse problem.

	Forward Problem Rel. Errors				*Inverse Problem Rel Errors*			
Grid size, s	65	129	257	513	65	129	257	513
DRM	0.0397	0.0289	0.0244	0.0251	-	-	-	-
PINN	0.0245	0.0088	0.0084	0.0075	0.1715	0.1694	0.1653	0.1650
QRES	0.0288	0.0082	0.0088	.0076	0.1598	0.1565	0.1542	0.1549
PCANN	0.0078	0.0075	0.0073	0.0073	0.0989	0.0979	0.0981	0.0981
PCALin	0.0016	0.0013	0.0013	0.0013	0.0303	0.0271	0.0253	0.0244
FNO	0.0066	0.0061	0.0067	0.0066	0.0299	0.0341	0.0366	0.0416
U-FNO	0.0063	0.0056	0.0062	0.0054	0.0292	0.0277	0.0288	0.0254
MWT	0.0047	0.0047	0.0048	0.0036	0.0357	0.0202	0.0198	0.0159
DeepONet	0.0047	0.0041	0.0041	0.0042	0.0983	0.0992	0.1041	0.1044
PINO	0.0031	0.0030	0.0034	0.0031	0.0263	0.0273	0.0319	0.0301
PI-DeepONet	0.0081	0.0057	0.0060	0.0061	0.1074	0.1068	0.1067	0.1068

Table 4.2: Error variation with the resolution for the Poisson problem. The networks were trained with noiseless data and backward operator was used for solving the inverse problem.

For the somewhat simple Poisson problem with vanishing boundary data, where both the forward and inverse operator are linear, there is not much difference between the DL concepts. Overall, we propose using PCALin, PCANN or PINN for first testing, due to their 'easy-to-use' structure. They produce acceptable results for a wide range of hyper-parameters and are comparatively easy to train. For more accurate results, we propose using MWT, which is however somewhat more involved; for implementation details, see the respective appendices.

4.2.3 Darcy Flow

Forward Problem

We now consider the steady-state of the 2-d Darcy flow equation on the unit square which is the second order, linear, elliptic PDE

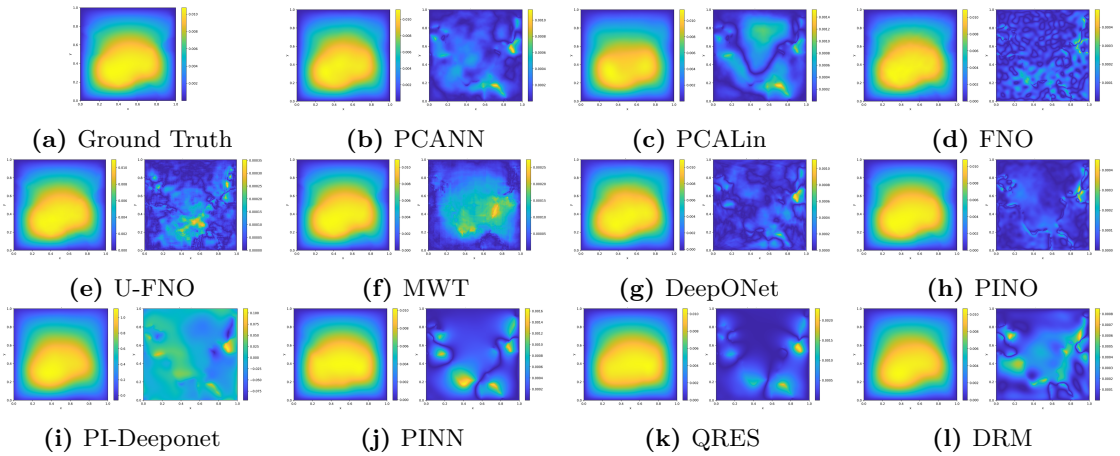
$$\begin{aligned} -\nabla \cdot (\lambda(s)\nabla u(s)) &= f(s) \quad s \in (0,1)^2 \\ u(s) &= 0 \qquad s \in \partial(0,1)^2 \end{aligned} \tag{4.4}$$

with a Dirichlet boundary. In this equation, $\lambda \in L^\infty\left((0,1)^2;\ _+\right)$ is the diffusion coefficient, $f = 1 \in L^2\left((0,1)^2;\ _+\right)$ is the forcing function and $u \in H_0^1((0,1)^2;\ )$ is the unique solution of the PDE.

We start with a discussion of the forward problem, i.e., computing the solution u in Equation 4.3, for a given piece-wise constant diffusion coefficient λ. In all experiments, $\lambda(x) \in \{3, 12\}$, i.e., the domain of definition is segmented randomly into two regions with known values.

Surprisingly, the errors for PINN and QRES are relatively high, even if the network is trained and evaluated with noise-free data, see Table 4.3. All other methods solve this non-linear forward problem reliably. MWT is the overall winner for solving the forward problem with noise-free data. This also holds for most levels of resolution.

Again, the comparison with respect to different resolutions of u shows the expected result that operator approximations, based on functional analytic reasoning in function spaces, indeed show an independent accuracy across scales, see Table 4.4.



(a) Ground Truth **(b)** PCANN **(c)** PCALin **(d)** FNO

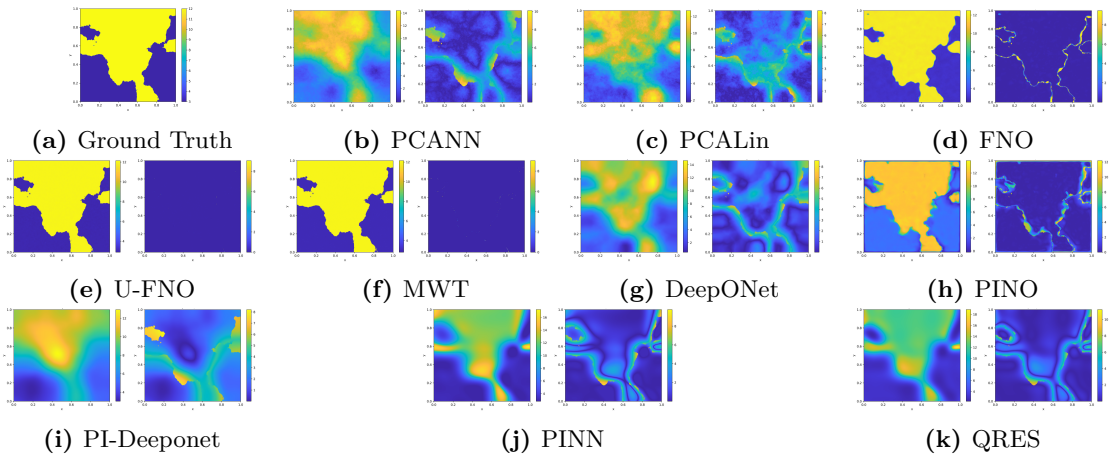
(e) U-FNO **(f)** MWT **(g)** DeepONet **(h)** PINO

(i) PI-Deeponet **(j)** PINN **(k)** QRES **(l)** DRM

Figure 4.5: Test examples for forward Darcy problem using a resolution of 513×513. (b)-(l), shows the specified neural network's approximation of the solution (left-hand side) and the absolute difference between the Ground Truth in (a) with the approximation (right-hand side).

Inverse Problem

The investigation of the performance of DL concepts for inverse Darcy flows again either uses inverse training or embedding of a forward solver into a Tikhonov functional. The



(a) Ground Truth **(b)** PCANN **(c)** PCALin **(d)** FNO

(e) U-FNO **(f)** MWT **(g)** DeepONet **(h)** PINO

(i) PI-Deeponet **(j)** PINN **(k)** QRES

Figure 4.6: Test examples with backward operator training for inverse Darcy Flow problem using a resolution of 513×513. (b)-(k), shows the specified neural network's approximation of the solution (left-hand side) and the absolute difference between the Ground Truth in (a) with the approximation (right-hand side). All examples are computed with noise free data. For reconstructions with noisy data see Figure 4.10.

testing scenarios are similar to those of the Poisson problem, i.e., 1000 random samples were used for training and 5000 samples for evaluation.

In view of different applications, we have either used no further assumption on λ, see Figure 4.6, or we have assumed that we know λ is a segmentation of the domain of definition into two components with a known value, see Table 4.10. This models the problem of mixing two liquids with known densities.

For a closer investigation, we start with a basic experiment for solving the Darcy inverse problems by inverse training with noise-free data. Results using a 513×513 resolution and without knowing the values oλ a priori are shown in Figure 4.6. We note the strength of the FNO, U-FNO, MWT and PINO to identify the position of the discontinuities with backward operator training. PINO, however, computes a mismatch in the values of the densities. The network architecture seems to play a crucial role in these results. For e.g., DeepONet always provides a reconstruction, which is obtained by a linear combination of smooth functions(trunk nets), and in solution learning methods $u_\Theta(x)$ is naturally continuous, hence discontinuities are not in the range of these methods. Other methods based on discretisation, i.e., reconstructing the densities in the last layer of the networks at prescribed positions allow for sharper transitions of the densities.

As before, we have done experiments where the networks were trained with noiseless or noisy data. The results of inverse training with perfect data for training and evaluation are stated in Tables 4.3 and 4.4; they mimic the results of the forward training and do not yield additional insights. As for the Poisson problem, training with noise-free data does not generalise to noisy data, see Table 4.5; even only 1% error in the evaluation data yields reconstruction errors of 30% and larger. We should remark on the sub-optimal results obtained by DeepONet, which shows errors of about 25% for all noise levels tested. Here,

the error margins are considerably larger, but almost constant over different noise levels. This might indicate that the dominant error is not stemming from inversion and noise, but rather from a structural disadvantage of the network architecture or sub-optimal training. We have tested several larger DeepONet network architectures, which however did not cure the problem. A major reason for the large error of DeepONet for this inverse problem is that DeepONet has to learn the basis of the output space via the trunk net. Thus, the trunk net must have sufficient expressiveness to approximate the basis of the output space. In addition, even if the size of the trunk net is large enough, finding the optimal parameters of the trunk net is another challenge. Thus, if the number of reduced basis of the output space is large or the output functions are very complicated (e.g., highly oscillatory or discontinuous), it would be very difficult for the trunk net to learn the basis accurately. For both Poisson and Darcy problems, the parameter spaces are more complicated than the solution spaces, which explains the large error of DeepONet for inverse training.

Tables 4.6 and 4.8 state the results most important for inverse problems. They show some remarkable differences between the considered concepts. Most methods (PINN, QRES, PCANN, PCALin, DeepONet, PI-DeepONet) show approximately the same errors for all noise levels. This indicates that a structural approximation error dominates the influence of the data noise. We have tested different hyperparameter settings for all of these methods, which, however, did not change the result. As a consequence, these methods do not yield competitive results for small levels of data noise. Nevertheless, PCANN exhibits about 10% error in the reconstructions even for 5% data error, which is remarkably good. In our test scenarios, there are clear winners: U-FNO and MWT for small noise levels and PCANN for larger noise levels. While the DeepONet, its physics-informed variant PI-DeepONet as well as PINO, do not seem to be suitable for Darcy inverse problems. As mentioned earlier, PINO is capable of detecting the discontinuities but it fails to provide accurate density estimates.

When comparing the best methods for inverse training (MWT for small noise levels, PCANN for larger noise levels, see Table 4.6) with the best method for Tikhonov minimisation, see Table 4.8, we see an advantage for inverse training, which generally yields better results. That errors are consistently larger for Tikhonov learning might be partially explainable by the difficulty of choosing a suitable regularisation parameter α. We have also included the accuracy metric which measures the segmentation error rather than the functional L_2 error. Again, results are similar for all methods, see Table 4.8a.

We finally estimated the output error by feeding the reconstructed $\hat{\lambda}$ into the respective neural networks and comparing the resulting $\hat{u}$ with the exact data u and its noisy version u^δ, see Figures 4.8c and 4.8.

Summary Darcy Flow

In summary, the investigated DL concepts produce competitive results for solving inverse problems with different noise levels. DL concepts for operator approximation perform best

Networks	# of Parameters	Forward Problem			Inverse Problem		
		Rel. L2 Error	Training (s/epoch)	Testing (s)	Rel. L2 Error	Training (s/epoch)	Testing (s)
DRM	22,201	0.0369	0.0859	$\sim 1,050$	-	-	-
PINN	6,291 | 7,972	0.1995	0.1164	$\sim 2,010$	0.1988	0.2528	$\sim 4,790$
QRES	6,562 | 8,895	0.2017	0.2108	$\sim 3,320$	0.1975	0.4221	$\sim 6,420$
PCANN	5,155,150 | 5,035,030	0.0253	0.0136	0.6108	0.0988	0.0487	0.1548
PCALin	10,100	0.0656	0.0075	0.3290	0.2203	0.0139	0.3108
FNO	2,368,001	0.0109	41.6857	0.0165	0.1490	42.0596	0.0163
U-FNO	3,990,401	0.0095	97.9580	0.0344	**0.0085**	98.8170	0.0345
MWT	9,807,873	**0.0058**	112.4575	0.0637	**0.0156**	112.8750	0.0403
DeepONet	568,320 | 1,047,644	0.0295	0.0606	0.0011	0.2220	0.0659	0.0011
PINO	2,368,001	**0.0084**	42.8685	0.0168	0.2345	42.7417	0.0142
PI-DeepONet	568,320 | 684,288	0.0384	0.2513	0.0011	0.2720	0.4905	0.0011

Table 4.3: Performance of different methods for the Darcy flow problem with piece-wise constant coefficients and using a 513×513 resolution. A similar convention is used for the second column as in Table 4.3.

Grid size, s	*Forward Problem Rel. Errors*				*Inverse Problem Rel Errors*			
	65	129	257	513	65	129	257	513
DRM	0.0501	0.0375	0.0358	0.0369	-	-	-	-
PINN	0.2254	0.2043	0.2071	0.1995	0.2413	0.2014	0.2032	0.1988
QRES	0.2596	0.2077	0.1993	0.2017	0.2371	0.2117	0.2036	0.1975
PCANN	0.0256	0.0254	0.0253	0.0253	0.0983	0.0985	0.0987	0.0988
PCALin	0.0656	0.0656	0.0656	0.0656	0.2191	0.2198	0.2201	0.2203
FNO	0.0113	0.0106	0.0107	0.0109	0.1661	0.1536	0.1501	0.1490
U-FNO	0.0078	0.0071	0.0074	0.0095	0.0928	0.0640	0.0322	0.0085
MWT	0.0080	0.0060	0.0059	0.0058	0.1047	0.0800	0.0461	0.0156
DeepONet	0.0290	0.0292	0.0289	0.0295	0.2219	0.2261	0.2221	0.2220
PINO	0.0078	0.0066	0.0071	0.0084	0.1703	0.1997	0.2216	0.2345
PI-DeepONet	0.0390	0.0382	0.0381	0.0384	0.2706	0.2711	0.2682	0.2720

Table 4.4: Error variation with resolution for the Darcy Flow problem with piece-wise constant coefficients. The networks were trained with noiseless data and backward operator training was used for solving the inverse problem.

and offer a significant advantage in run time. Hence, either large-scale parametric studies or iterative solvers for parameter identification decisively benefit from well-trained DL concepts. This training has to be done with the same noise level, training with noise-free data does not generalise to noisy data. As a general procedure, we would recommend favouring backward operator training over Tikhonov minimisation.

When choosing appropriate methods, we propose to use inverse training in combination with U-FNO and MWT for small noise levels and, the concept PCANN was best for solving the non-linear inverse Darcy flow problem with higher noise levels.

As extensive as our numerical tests have been, they only provide a snapshot of the diverse landscape of PDE problems and testing scenarios. In particular, it might be interesting e.g., to test Darcy inverse problems with continuous parameters and inverse Poisson problems with piece-wise constant functions. However, for the sake of an overview and the already somewhat lengthy list of tables presented, we decided to restrict ourselves to the tests presented in the work.

	Poisson					*Darcy Flow PWC*				
Noise level	0%	0.1%	1%	5%	10%	0%	0.1%	1%	5%	10%
PCANN	0.0988	0.0993	0.1171	0.3525	0.5645	0.0983	0.0983	0.0984	0.0998	0.1039
PCALin	0.0303	0.0379	0.0737	0.3134	0.6228	0.2191	0.2191	0.2192	0.2231	0.2348
FNO	0.0299	0.2966	2.6444	10.5499	19.9952	0.1341	0.1743	0.5100	3.3888	7.5197
U-FNO	0.0292	0.2102	1.1645	4.3941	8.3548	0.0896	0.1202	0.5896	1.4194	2.2052
MWT	0.0316	0.2201	1.2716	4.1969	7.6609	0.0865	0.1123	0.3668	0.6949	0.8620
DeepONet	0.0983	0.0985	0.1048	0.2059	0.3748	0.2219	0.2223	0.2224	0.2262	0.2375
PINO	0.0263	0.3457	3.3960	16.8352	33.8227	0.1703	0.1789	1.0438	6.1264	12.3570
PI-DeepONet	0.1074	0.1075	0.1120	0.1917	0.3352	0.2706	0.2706	0.2706	0.2714	0.2737

Table 4.5: Effects of noise on the solution for the inverse problems on a 65×65 resolution. The network is trained with noise-free data, but evaluated with noisy data. Backward operator training is used for solving the inverse problem.

	Poisson					*Darcy Flow PWC*				
Noise level	0%	0.1%	1%	5%	10%	0%	0.1%	1%	5%	10%
PINN	0.1715	0.1727	0.1734	0.2110	0.2378	0.2413	0.2407	0.2496	0.2564	0.2893
QRES	0.1598	0.1658	0.1714	0.2007	0.2256	0.2371	0.2421	0.2617	0.2789	0.2951
PCANN	0.0988	0.0991	0.1293	0.1876	0.2273	0.0983	0.0987	0.0990	0.1012	0.1093
PCALin	0.0303	0.0318	0.0840	0.1478	0.1846	0.2191	0.2191	0.2191	0.2217	0.2315
FNO	0.0299	0.0554	0.0957	0.1389	0.1678	0.1341	0.1344	0.1449	0.1770	0.2019
U-FNO	0.0292	0.0594	0.1036	0.1633	0.1839	0.0896	0.0902	0.1219	0.1649	0.1932
MWT	0.0316	0.0533	0.0966	0.1525	0.1867	0.0865	0.0893	0.1115	0.1634	0.1956
DeepONet	0.0983	0.1042	0.1084	0.1537	0.1870	0.2219	0.2283	0.2273	0.2379	0.2522
PINO	0.0263	0.0570	0.0957	0.1389	0.1678	0.1703	0.1737	0.2083	0.5147	0.9699
PI-DeepONet	0.1074	0.1086	0.1137	0.1516	0.1834	0.2706	0.2680	0.2703	0.2735	0.2742

Table 4.6: Effects of noise on backward operator inverse problems on a 65×65 resolution. The network is trained with noisy data and datasets with the same noise level are used for testing.

4.3 Part 2: Complex Problems

Following the study performed in section 4.2, which studied less complex PDEs of the Poisson and Darcy flow problems, this section seeks to study how the methods perform with more complex PDEs such as the Navier-Stokes, Helmholtz, Advection equations as well a solid/structural mechanics problem. These problems were extensively studied in [68]. There, the focus was on the forward problem, and the writers comparatively look at the performance of the various neural operators with cost. However, in this work, we focus on using neural operators to solve the respective inverse problems. In the following, we provide the description of the corresponding inverse problems for the forward problems stated in [68]. For consistency, we denote the parameter by $\lambda \in \Lambda$, the parameter space which we aim at identifying, and the solution (or measurement) $u \in \mathcal{U}$, the solution space.

Furthermore, we once again define the Gaussian Random Fields (GRF), commonly used in the stochastic modelling of physical phenomena (and has gained popularity in deep learning methods for solving PDEs literature). Specifically, we use the Gaussian distribution μ_{G} with mean $\bar{\mu}$ and variance $(-\Delta + \tau^2)^{-d}$ as base measure. Thus,

$$\mu_{\mathrm{G}}(\bar{\mu}, \tau, d) = \mathcal{N}\left(\bar{\mu}, (-\Delta + \tau^2)^{-d}\right), \tag{4.5}$$

with a zero Neumann boundary condition on the Laplacian operator Δ, which yields random but smooth test data. τ denotes the inverse length scale of the random field,

	0%	0.1%	1%	5%	10%
PCANN	0.2128	0.2128	0.2137	0.2975	0.4468
PCALin	0.0803	0.0805	0.0918	0.1534	0.2132
FNO	0.0931	0.0936	0.1082	0.1450	0.1693
U-FNO	0.0687	0.0697	0.0909	0.1397	0.1802
MWT	0.0708	0.0714	0.0878	0.1368	0.1796
DeepONet	0.1341	0.1330	0.1350	0.1660	0.2088
PINO	0.0598	0.0623	0.0890	0.1441	0.1915
PI-DeepONet	0.1573	0.1580	0.1727	0.2040	0.2298

(a) λ_{err}, the relative error of the learned parameter

	0%	0.1%	1%	5%	10%
PCANN	0.0133	0.0128	0.0130	0.0123	0.0122
PCALin	0.0010	0.0010	0.0016	0.0074	0.0158
FNO	0.0022	0.0025	0.0097	0.0464	0.0924
U-FNO	0.0009	0.0013	0.0091	0.0457	0.0914
MWT	0.0014	0.0017	0.0093	0.0459	0.0917
DeepONet	0.0053	0.0054	0.0113	0.0497	0.0988
PINO	0.0006	0.0010	0.0088	0.0454	0.0911
PI-DeepONet	0.0045	0.0047	0.0135	0.0520	0.1013

(b) $\tilde{u}_{\text{err}}$, the relative error between the solution of the learned parameter and the noisy solution.

	0%	0.1%	1%	5%	10%
PCANN	0.0133	0.0128	0.0131	0.0150	0.0209
PCALin	0.0010	0.0010	0.0020	0.0074	0.0133
FNO	0.0022	0.0023	0.0040	0.0089	0.0133
U-FNO	0.0009	0.0010	0.0024	0.0072	0.0124
MWT	0.0014	0.0015	0.0025	0.0072	0.0124
DeepONet	0.0053	0.0053	0.0055	0.0085	0.0137
PINO	0.0006	0.0008	0.0023	0.0072	0.0126
PI-DeepONet	0.0045	0.0046	0.0058	0.0147	0.0152

(c) u_{err}, the relative error between the solution of the learned parameter and the noiseless solution.

Table 4.7: Effects of noise on Tikhonov-based inverse problems for the Poisson problem and using datasets with a resolution of 65×65. The errors shown here are averaged over 100 test samples.

	0%	0.1%	1%	5%	10%
PCANN	77.74	77.70	77.83	76.79	74.19
PCALin	90.58	90.58	90.57	90.55	90.42
FNO	96.92	96.92	96.71	95.45	94.30
U-FNO	94.23	93.83	95.20	94.14	92.48
MWT	98.26	98.33	97.84	96.42	95.04
DeepONet	93.88	93.88	93.95	93.49	92.70
PINO	97.56	97.59	97.45	96.34	95.07
PI-DeepONet	92.03	91.93	91.84	91.80	91.60

(a) $\lambda_{\text{acc}}(\%)$, the accuracy of the learned parameter

	0%	0.1%	1%	5%	10%
PCANN	0.4684	0.4692	0.4680	0.4828	0.5148
PCALin	0.3157	0.3157	0.3159	0.3163	0.3183
FNO	0.1764	0.1764	0.1823	0.2143	0.2402
U-FNO	0.2140	0.2205	0.2017	0.2371	0.2734
MWT	0.1287	0.1250	0.1435	0.1880	0.2209
DeepONet	0.2484	0.2483	0.2469	0.2547	0.2691
PINO	0.1559	0.1549	0.1607	0.1932	0.2244
PI-DeepONet	0.2792	0.2801	0.2820	0.2822	0.2861

(b) λ_{err}, the relative error of the learned parameter

	0%	0.1%	1%	5%	10%
PCANN	0.1151	0.1152	0.1144	0.1211	0.1373
PCALin	0.0929	0.0929	0.0929	0.0931	0.0934
FNO	0.0104	0.0105	0.0174	0.0711	0.1362
U-FNO	0.0477	0.0517	0.0419	0.0698	0.1145
MWT	0.0129	0.0125	0.0177	0.0544	0.1032
DeepONet	0.0480	0.0484	0.0495	0.0773	0.1190
PINO	0.0167	0.0164	0.0211	0.0567	0.1065
PI-DeepONet	0.0586	0.0593	0.0604	0.0816	0.1244

(c) $\tilde{u}_{\text{err}}$, the relative error between the solution of the learned parameter and the noisy solution.

	0%	0.1%	1%	5%	10%
PCANN	0.1151	0.1152	0.1144	0.1207	0.1363
PCALin	0.0929	0.0929	0.0929	0.0928	0.0923
FNO	0.0210	0.0212	0.0223	0.0329	0.0421
U-FNO	0.0477	0.0517	0.0395	0.0437	0.0496
MWT	0.0129	0.0125	0.0142	0.0209	0.0276
DeepONet	0.0480	0.0484	0.0483	0.0561	0.0597
PINO	0.0167	0.0164	0.0181	0.0250	0.0349
PI-DeepONet	0.0586	0.0593	0.0593	0.0614	0.0677

(d) u_{err}, the relative error between the solution of the learned parameter and the noiseless solution.

Table 4.8: Effects of noise on Tikhonov-based Inverse Problems on the Darcy Flow problem with PWC coefficients, using a dataset of resolution of 65×65. The errors shown here are averaged over 100 test samples.

69

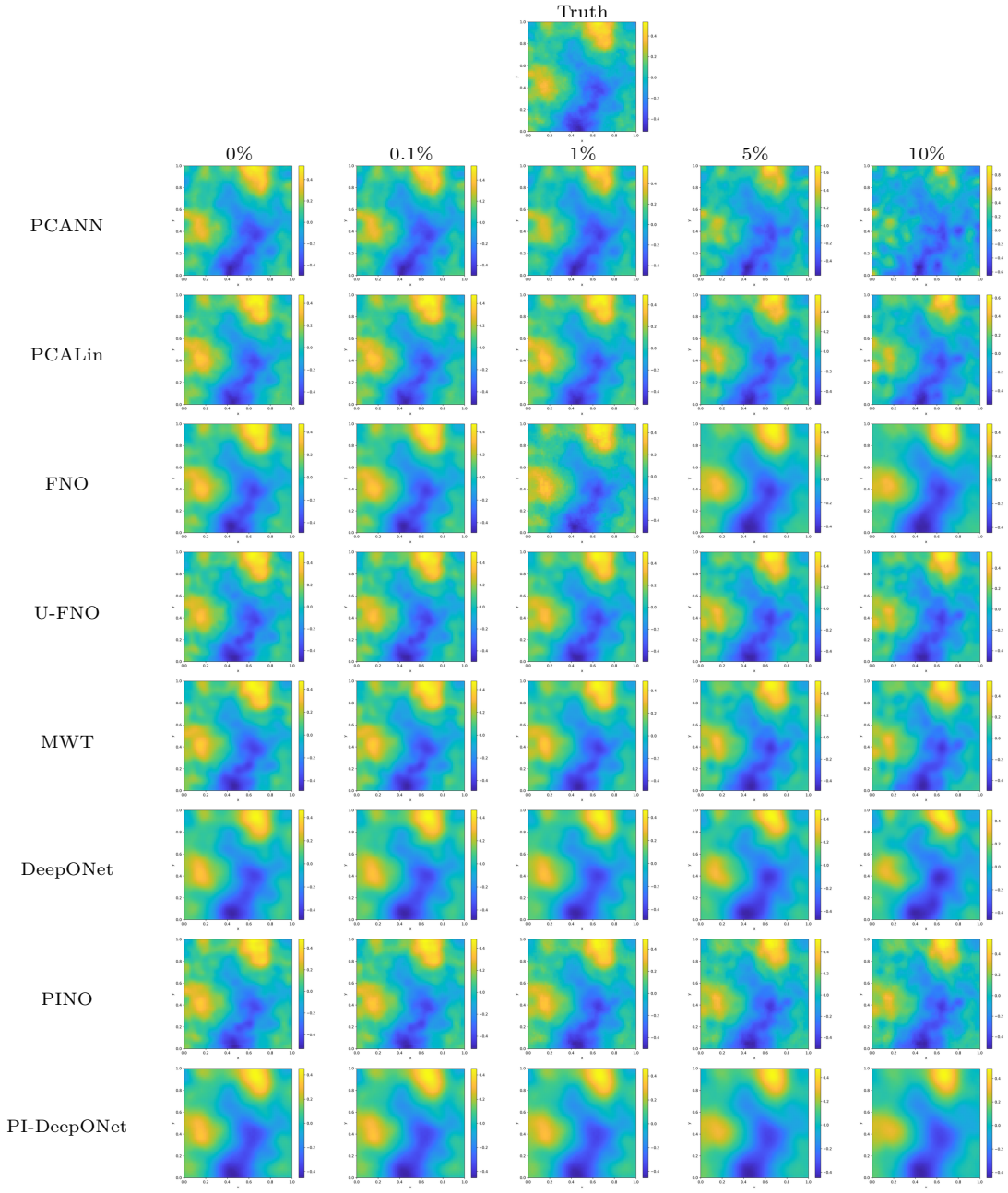


Table 4.9: Effects of noise on Tikhonov-based Inverse Problems on the Poisson problem, using a dataset of, resolution of 65×65.

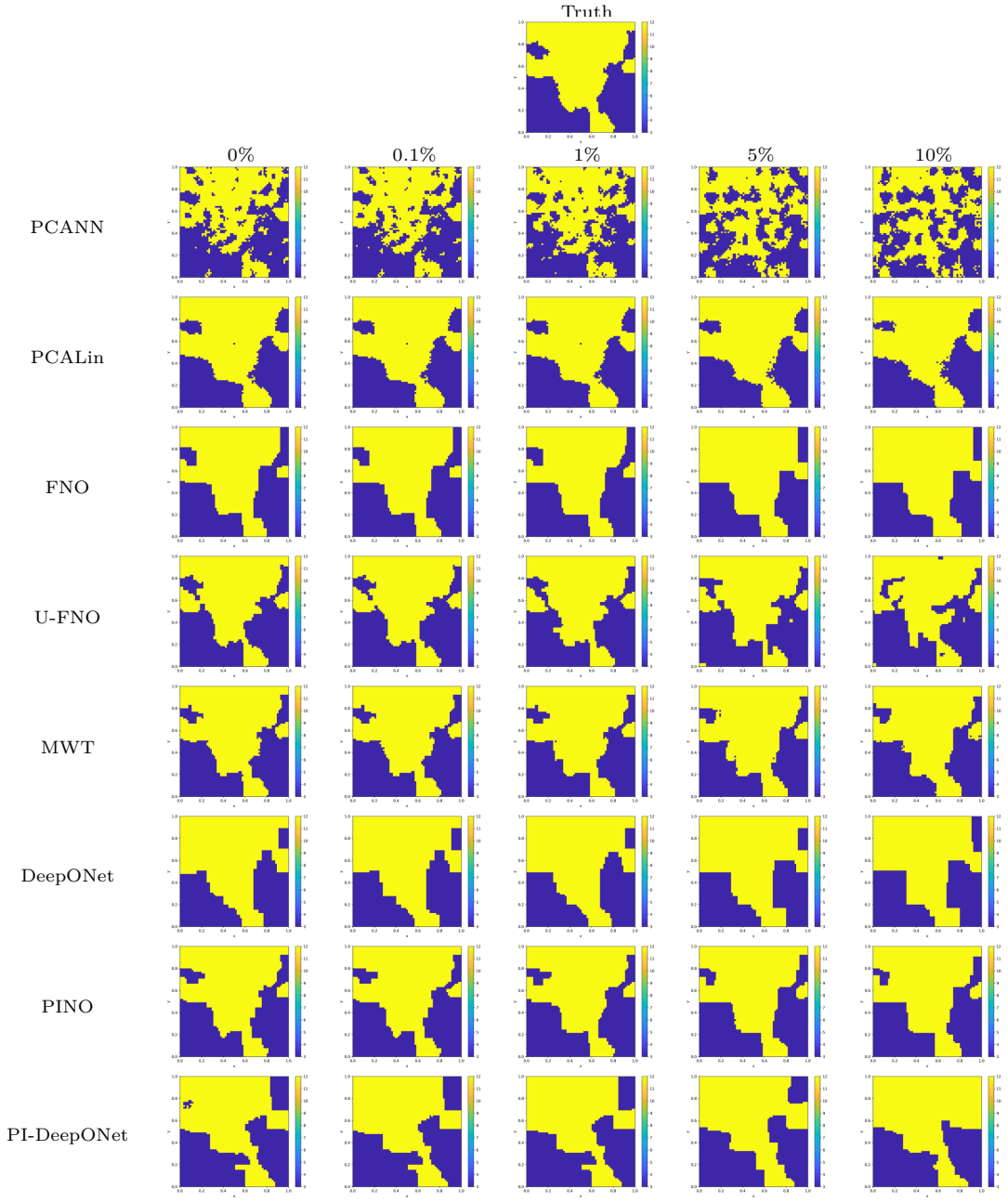


Table 4.10: Effects of noise on Tikhonov-based Inverse Problems on the Darcy Flow pwc problem, using dataset of, resolution of 65×65.

while d determines the regularity of the random field. The dataset used in this work is generated from μ_{G} or it's push-forward under a specified map. Figures 4.7-4.10 shows examples of the parameter-solution pair in the datasets. Details on how each is generated is further presented in the following sections 4.3.1- 4.3.4

4.3.1 Advection

Consider the 1D advection equation with advection speed c defined in the domain $\Omega = [0, 1)$

$$\begin{aligned} \frac{\partial u}{\partial t} + c\frac{\partial u}{\partial s} = 0 \quad s \in \Omega, \\ u(0) = \lambda \end{aligned} \tag{4.6}$$

We equally use the dataset from [144] where the initial condition $u(0) = \lambda$ is sampled from $\mu_{\mathrm{A}} = \mathrm{A}_\sharp \mu_{\mathrm{G}}$, the push forward of $\mu_{\mathrm{G}}(\bar{\mu} = 0, \tau = 3, d = 2)$ under the map

$$\mathrm{A}(x) = \begin{cases} 1 & x > 0 \\ -1 & x \leq 0 \end{cases}$$

Our interest is therefore to estimate the initial solution $u(0) = \lambda$ from the solution at time T, $u(T)$, thus the map Ψ given by

$$\Psi : \mathcal{U}\left([0, 1] \times \Omega; \quad\right) \ni u(T, \cdot) \mapsto \lambda \in \Lambda\left([0, 1] \times \Omega; \quad\right)$$

Specifically, we choose $T = 0.5$ and the advection speed $c = 1$ and the PDE in Equation 4.6 is solved analytically for a 1-dimensional grid of size 200. Figure 4.7 gives a sneak peek at the nature of the samples in the dataset.

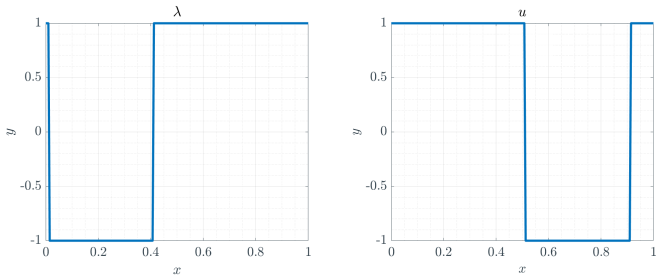


Figure 4.7: Advection dataset sample

4.3.2 Solid Mechanics

Consider an elastic solid in the domain Ω, undergoing inappreciable deformations characterised by the Cauchy stress tensor σ and resulting displace d. We define Γ_v and Γ_λ, two disjoints parts of the boundary $\partial\Omega$ (with outward normal n) of the domain Ω, i.e $\Gamma_v \cup \Gamma_\lambda = \partial\Omega$, and $\Gamma_v \cap \Gamma_\lambda = \quad$. If v is the prescribed displacement imposed on the

boundary of the domain Γ_v and λ is the surface traction on the remaining part of the domain Γ_λ. The PDE governing such a system is

$$\begin{aligned} \nabla \cdot \sigma &= 0 \quad \text{in } \Omega, \\ d &= v \quad \text{on } \Gamma_v \\ \sigma \cdot n &= \lambda \quad \text{on } \Gamma_\lambda \end{aligned} \tag{4.7}$$

Once again, the elastic solid considered is that in [144], which is made of a cylindrical fibre at the centre, made of a linear elastic material, with specified density, Young's modulus and Poisson ratio. Our parameter identification problem here is to obtain the one-dimensional traction λ from the von Mises stress field u, thus the map Ψ given by

$$\Psi : \mathcal{U}(\Omega; \quad) \ni u(s) \mapsto \lambda \in \Lambda(\Gamma_\lambda; \quad_+).$$

This traction is drawn from the GRF $\mu_S = S_\sharp \mu_G$, the push forward of $\mu_G(\bar{\mu} = 100, \tau = 3, d = 1)$ under the map

$$S(x) = 400^2 x$$

We equally highlight that the dataset was generated using a finite element approach with the NNFEM library [322] [147], and interpolated on a 41×41 grid. The load used was on a 1-dimensional grid of size 21.

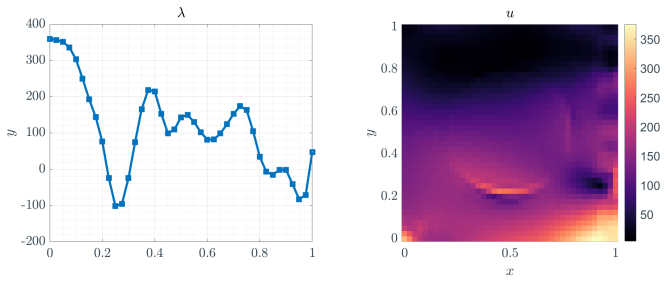


Figure 4.8: Solid Mechanics dataset sample

4.3.3 Navier-Stokes

Consider the incompressible Navier-Stokes equation for a fluid with being the kinematic velocity ν, defined in $\Omega = [0, 2\pi^2]$ and given by

$$\begin{aligned} &\frac{\partial u}{\partial t} + (v \cdot \nabla)u - \nu \Delta u = \lambda \\ &u = -\Delta \psi \quad \int_\Omega \psi = 0 \\ &v = \left(\frac{\partial \psi}{\partial s_2}, -\frac{\partial \psi}{\partial s_1} \right). \end{aligned} \tag{4.8}$$

Equation 4.8 is the vorticity-stream formulation, popularly used for solving the 2D incompressible Navier-Stokes equation. This form is obtained by a change of variable

which replaces the components of the velocity v in the continuity and momentum equations with the vorticity u and stream function ψ, transforming the mixed elliptic-parabolic 2-D incompressible Navier-Stokes equations into a parabolic equation and an elliptic (Poisson) equation. We are interested in inferring the forcing λ from the vorticity $u(T, \cdot)$ at a later time, T, thus the map Ψ given by

$$\Psi : \mathcal{U}\left([0,1] \times \Omega; \quad\right) \ni u(T, \cdot) \mapsto \lambda(s) \in \Lambda\left(\Omega; \quad\right)$$

The dataset used in our experiment is obtained from [144], where the forcing term is sampled from the distribution $\mu_{\mathrm{G}} = \mu_{\mathrm{G}}(\bar{\mu} = 0, \tau = 3, d = 4)$. The initial vorticity $u(0)$, is equally sampled from this distribution. With $\nu = 0.025$, the PDE in 4.8 is solved at final time $u(T)$, where T = 10, using a pseudo-spectral method on a grid of size 64×64, and a Crank-Nicolson scheme for time integration. An example parameter-solution pair is shown in Figure 4.9

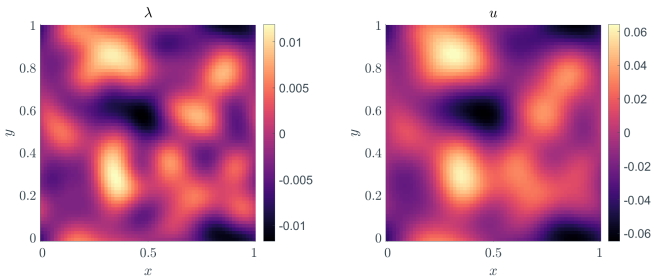


Figure 4.9: Navier Stokes dataset sample

4.3.4 Helmholtz

For a given frequency $\omega \in \quad_{+}$, and wave speed $\lambda \in \Lambda\left(\Omega; \quad_{+}\right)$, the Helmholtz equation defined in $\Omega = [0,1]^2$ and given by

$$\begin{aligned} \left(-\Delta - \frac{\omega^2}{\lambda^2(s)}\right) u = 0 \quad & \text{in } \Omega = (0,1)^2, \\ \frac{\partial u}{\partial n} = 0 \quad & \text{on } \partial\Omega_1, \partial\Omega_2, \partial\Omega_4 \\ \frac{\partial u}{\partial n} = 1 \quad & \text{on } \partial\Omega_3, \end{aligned} \tag{4.9}$$

is the time-independent form of the wave equation, and has as solution the excitation field $u(s) \in \mathcal{U}\left([0,1]^2; \quad_{+}\right)$, where $\partial\Omega_1, \partial\Omega_2, \partial\Omega_4$ are the south, east, and west boundaries respectively, all with zero Neumann boundary conditions and $\partial\Omega_3$ is the north boundary. We are interested in inferring the inhomogeneous wave speed field λ from disturbance field u, thus the mapping

$$\Psi : \mathcal{U}\left(\Omega; \quad_{+}\right) \ni u(s) \mapsto \lambda(s) \in \Lambda\left(\Omega; \quad\right)$$

Once again, we use dataset from [144], which samples the inhomogeneous wave speed from $\mu_{\mathrm{H}} = \mathrm{H}_\sharp \mu_{\mathrm{G}}$, the push forward of $\mu_{\mathrm{G}}(\bar{\mu} = 0, \tau = 3, d = 2)$ under the map

$$\mathrm{H}(x) = 20 + \tanh x.$$

The PDE in 4.9 is then solved by finite element method on a 100×100 grid. Figure 4.9 shows a sample parameter-solution pair.

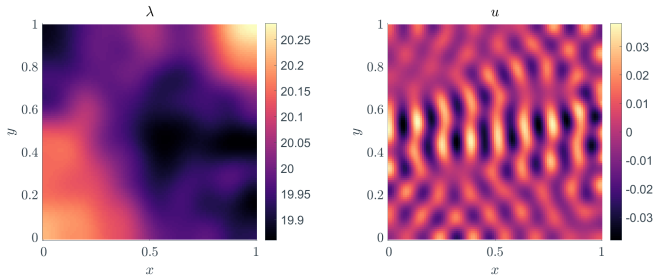


Figure 4.10: Helmholtz dataset sample

4.3.5 Results and Discussion

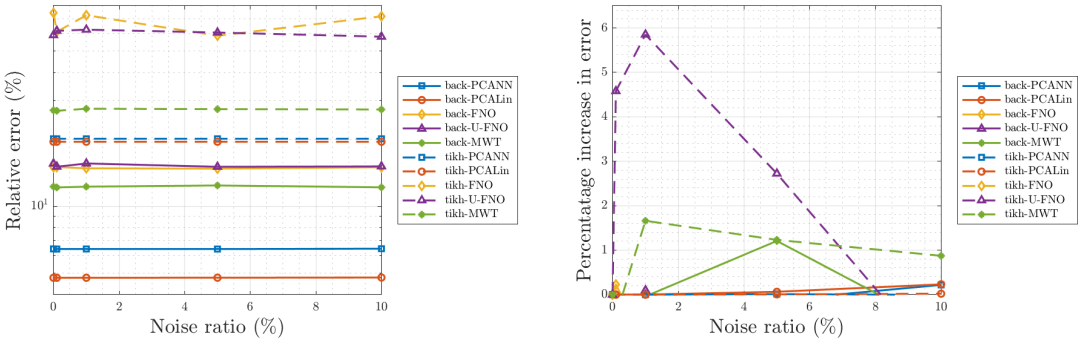
The results obtained from the complex problems follow a similar trend to those of the complex problems. Results for all methods but for the DeepONet, PI-DeepONet, and PINO are shown here. Equally, the function evaluation DL approaches are also not shown here. Firstly, we present the performances of the forward operators in table 4.11 These forward operators are further used for parameter identification in the Tikhonov-based approach, for which results are shown in figures 4.11-4.14. For each of these figures, (a) shows the performance with noise, while (b) shows how this performance in the noisy scenarios compares to that in the noiseless (0%) scenario. Specifically, for the Advection problem, the backward methods show better performance compared to the Tikhonov-based methods as depicted in figure 4.11a. A similar observation is also seen for the solid mechanics problem in figure 4.12a. However, the Tikhonov-based methods show a better scaling with noise, as depicted in figure 4.12b. The Navier-Stokes problem equally shows similar results in 4.13a but for the Tikhonov-based FNO, which scales abnormally with noise as depicted in figure 4.13b. Particularly interesting are the Tikhonov-based approaches for the Helmholtz problem, which (but for the PCANN) beat their backward counterparts (see figure 4.14a. All Tikhonov-based methods, however, do scale better with noise (see figure 4.14b).

4.4 Conclusions

The presented survey aims at developing a guideline on whether and how to use deep learning concepts for PDE-based forward solvers and related inverse problems. To this end, we have implemented and tested the most widely used neural network concepts as well as

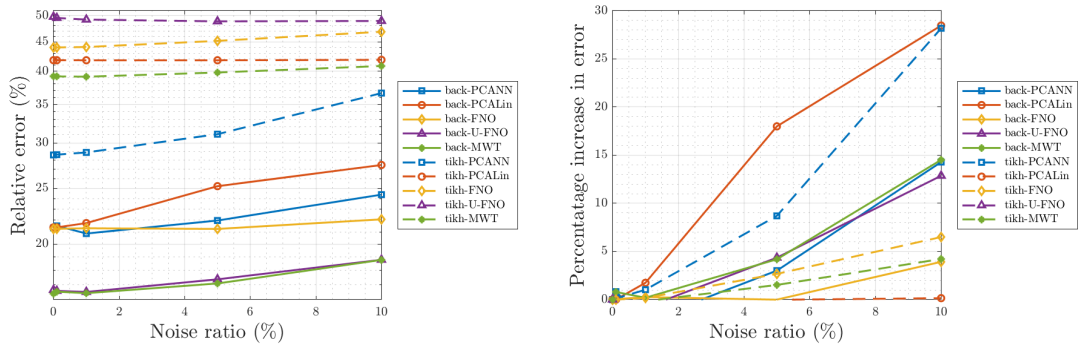
Neural Operator	Navier Stokes	Solid mechanics	Helmholtz	Advection
PCANN	0.06277	0.06039	0.04979	0.06381
PCALin	0.05610	0.23230	0.10309	0.04685
FNO	0.00330	0.07290	0.03478	0.14789
U-FNO	0.00613	0.05751	0.03836	0.14664
MWT	0.00250	0.05640	0.03872	0.12691

Table 4.11: Performance (relative error) of the respective forward problem solvers



(a) Relative errors **(b)** Change in relative errors.

Figure 4.11: Advection



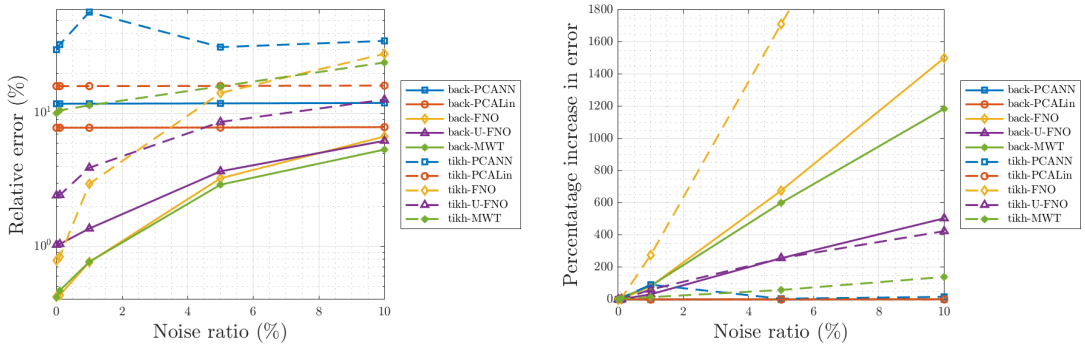
(a) Relative errors **(b)** Change in relative errors.

Figure 4.12: Solid Mechanics

several of their extensions for the two most common PDE problems (Poisson, Darcy flow). In addition, we have applied neural network concepts to two industrial projects.

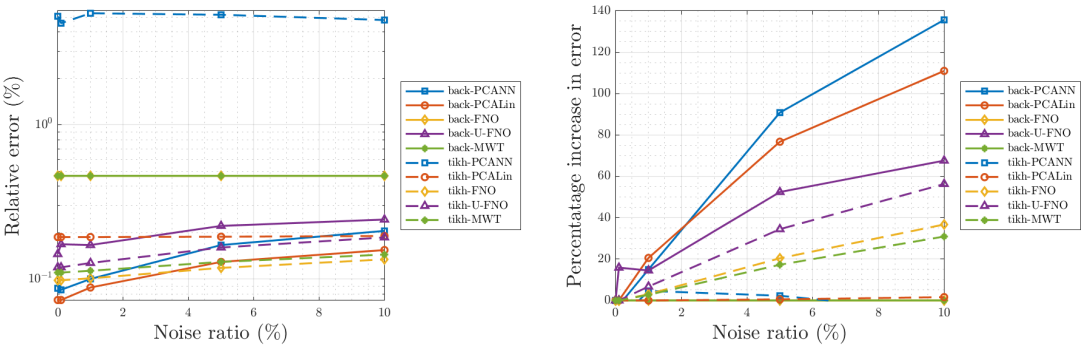
Naturally, the present text tries to give a complete-as-possible snapshot of the situation at the time we started writing up the results of our test. We are well aware that several additional concepts have been proposed since and we apologise to those whose contributions are missing.

There is no unique criterion for determining the best method, this very much depends on the specific task at hand. Hence, we presented a survey of numerical values for each method, which should allow the user to evaluate the particular criterion relevant to its specific application and to determine a suitable method accordingly.



(a) Relative errors **(b)** Change in relative errors.

Figure 4.13: Navier Stokes



(a) Relative errors **(b)** Change in relative errors.

Figure 4.14: Helmholtz

Nevertheless, we want to highlight some general findings. Within the restricted scope of our test environment, we have found the following conclusions. DL concepts for solely solving forward problems lack precision as compared with classical FEM methods. This finding has been confirmed by other authors as well, see e.g., a numerical study for PINNs in [112]. However, in general, they achieve reasonable accuracy and offer great potential for efficient parameter studies, e.g., for applications in rapid prototyping.

Our findings differ depending on whether the underlying PDE is linear or non-linear and whether training data, i.e., sufficiently many parameter-state pairs are known, is available. Several methods such as Ritz-method or PINN only use the PDE and do not use training data. Hence the comparison has to distinguish these cases

For a linear PDE and if only the PDE is known but no data is available, we propose to use PINN for solving the forward problem. If data, i.e., sufficiently many parameter-state pairs are known, then PCA or even better PCALin are an excellent starting point. As mentioned, this is only the first indication, our tests were restricted to the Poisson problem. Surprisingly, see Table 4.6, there is not much difference between different concepts for solving the inverse Poisson problem by backward operator training. All methods considered produce reliable results e.g., for 5% noise in the data. Hence, run times might be the

criterion for the decision. Obviously, concepts for operator training are advantageous and DeepONet and PI-DeepONet are the overall winners in terms of run-time for testing.

The Poisson forward problem is a rather simple linear elliptic Hence, the numerical test should be seen as some preliminary test giving some limited insight into the behaviour of DL concepts for forward linear PDEs. As we will see from the more evolved examples, the findings in this section have only a limited meaning for more general cases.

Concerning solvers for the Darcy problem, MWT is the overall winner for solving the forward problem with noise-free data, at least in our experiments. For solving the inverse Darcy problem we propose to use inverse training in combination with U-FNO and MWT for small noise levels and PCANN for solving the non-linear inverse Darcy flow problem with higher noise levels.

As a general remark, we observe that concepts which combine data-driven learning with mathematical concepts such as reduced order models or incorporating the PDE directly, e.g., PCANN/PCALin, PINO, FNO or MWT, have an advantage in terms of stability and performance. Again, we want to emphasise that it is premature to make this a general rule, this is rather an observation restricted to our numerical setting.

Naturally, there are many meaningful extensions of the chosen test scenario. e.g., inserting a classical FDM/FEM forward solver into a Tikhonov regularisation scheme or including H^1 - error measures for comparison would be most interesting. However, we prefer to stay with the present scope of focusing on comparing DL methods for inverse problems and restrict ourselves to the tests provided.

Concerning industrial applications, additional features gain influence. e.g., the simulation of heat inside a rotating engine requires working with a time-dependent domain of definition. Most codes will require substantial additional work when dealing with such complex domains. TorchPhysics offers an accessible concept for working with such domains, which in connection with a standard PINN concept did yield good performance for a parametric study in this context.

The other industrial problem of identifying parameters in a problem of mechanical engineering reduces to a rather low-dimensional task and can be solved by PCANN. Hence, choosing a method suitable for a given industrial problem requires problem-specific considerations, which have to be evaluated task by task.

We should remark, that the size of the network differs considerably. We have performed rather extensive hyper-parameter searches for each method. The reported network sizes did yield the best results. E.g., the DRM does not benefit from extending the network to larger sizes for the mentioned application.

In Table 3.1, we list the properties of classical methods, Deep Ritz method in [327], PINN in [261], model reduction and Neural Network in [24], DeepONets in [222] and Fourier Neural Operator [211]. We should mention that the properties only correspond to the methods in these mentioned papers, since by some modifications, they can have different properties.

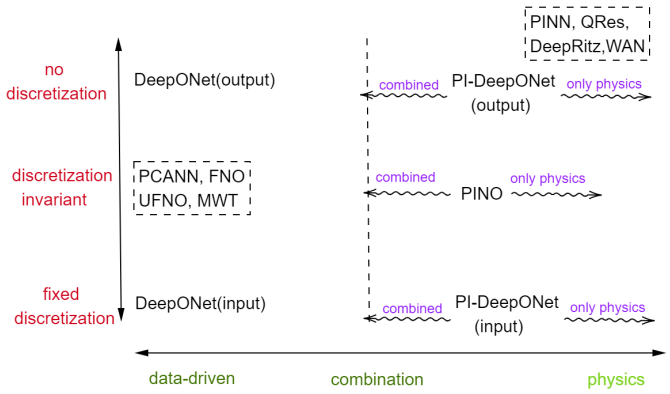


Figure 4.15: Properties of deep learning methods for PDEs.

For example, the PINN may be used to model parameter-solution operators. In Figure4.15, we schematically illustrate the properties of different deep learning methods for PDEs.

The methods discussed above have shown their high potential for solving PDEs as well as their related parametric studies and inverse problems. Whenever efficiency is an issue, these methods perform most convincingly. However, there are still many open questions and problems that are common in the deep learning community. Hyperparameter search can be very time-consuming and would benefit from a sound theoretical foundation. Also, the choice of activation functions and loss functions can be more refined.

It always seems impossible until it's done.

— *Nelson Mandela*

5 Parameter Identification of a Material Model for Granular Media

In this chapter, an application problem is considered, and a DL-based method is proposed. It is based on the following article:

> Derick Nganyu Tanyu, Isabel Michel I, Andreas Rademacher, Peter Maass and Jörg Kuhnert. "Parameter identification by deep learning of a material model for granular media". In: *Manuscript submitted for publication (Jul. 2023). Preprint available at (arXiv:2307.04166).*

5.1 Introduction

In geosciences, engineering, and industry, partial differential equations (PDEs) are widely used to model a great variety of problems. They are a great tool for modeling and solving complex phenomena ranging from the incompressible flow in the Earth's mantle to the electronic structure of materials. For an industrial product, these models follow the full life cycle from classical simulation and optimization during the development phase to process monitoring and control during production. PDE models generally introduce some critical parameters, which have to be calibrated so that the model reflects the system or problem being considered. These parameters could be scalar or space- and time-dependent parameter functions, and their calibration process usually requires multiple runs of the model. In some scenarios, one has access to the solution of the PDE or observation of the system and wishes to infer the parameters underlying the governing PDE, thus an inverse problem. A wide range of inverse problems have been studied, such as tomography [13], inverse

kinematics [300], inverse problems in signal processing [3] and even in quantum mechanics [42]. In the field of geomathematics [89], we equally identify inverse problems such as gravimetric inversion [90, 181], seismic inversion [69, 247] , geodynamic inversion[266], and inverse magnetometry [27] among others. Particularly in [267], by a combination of geophysical imaging and inversion methods of gravimetry, seismology, and geodynamics in a single cost function, the authors study the reconstruction of the Earth's material parameters. In this inverse problem, the system is in essence governed by the Poisson, wave, and Stokes equations, where the gravitational potential, seismic displacement, and surface velocity are respectively observed for the estimation and identification of mass density, wave speed, and the viscosity parameters. A helpful review of selected inverse methods to infer parameters for problems in geosciences can be found in [265].

PDE-based inverse problems are one of the most challenging inverse problems because their complexity is compounded by the fact that the solutions are typically nonlinear. This further emphasizes the need for efficient and fast solvers. While traditional or standard numerical methods such as finite differences and finite elements have been used extensively to solve PDEs, most if not all these standard PDE solvers suffer from the curse of dimensionality [20], i.e. the computational cost grows exponentially as the dimension increases. This has led to the extensive study of data-driven concepts, particularly, neural network approaches for solving PDEs over the last few years. In addition to their potential of overcoming the curse of dimensionality, these data-driven concepts usually have the potential to complete mathematical-physical models as even the finest detail or tricky non-linearity is contained in a sufficient dataset. In addition, since the parameters to be determined most often are not arbitrary, but follow an unknown, application-specific distribution, the training data provides a means to recover and exploit this distribution.

This chapter looks at a PDE-based inverse problem in the field of continuum mechanics, which is applicable to the automobile development process. Specifically, our focus is on a physical model of soil over which vehicles ride. However, the methodology is also transferable to material modeling tasks in geosciences such as the parameterization of material models for snow or landslides. They are necessary e.g. to predict runout zones or to evaluate protective measures. Furthermore, [244] shows how the proposed method and its likes (data-driven methods) can be used to solve the inverse Poisson problem, which has applications in gravimetry [267]. Similar works [211, 262] also show the potential of data-driven methods to solve Stokes and wave equations with respective applications in geodynamics and seismology [185].

The rest of this work is structured as follows: We continue in Section 5.1.1 by looking into reduced order models (ROM) and how proper orthogonal decomposition (POD), as well as deep learning (DL), can be used in ROMs. We equally highlight in Section 5.1.2, how neural networks have been recently applied for PDE solutions, parametric studies, and inverse problems. We then proceed to present the defining equations of our problem in

Section 5.2 and the laboratory test setting, which provides the basis of the MESHFREE simulations [156] used for the data generation. Section 5.3 presents the method used to approach the problem, i.e. PCA-NN. In Section 5.4, we summarize the numerical results, followed by concluding remarks in Section 5.5.

5.1.1 Reduced Order Models and POD/PCA

Full-order models (FOM) like the finite difference method (FDM), finite element method (FEM), finite volume method (FVM), discontinuous Galerkin method (DGM), etc. that discretize the PDEs are usually highly accurate but very expensive. Depending on the application and the goals set, the user has to balance accuracy and computation time as an algorithm of higher accuracy implies higher computation time. In FDM, for example, a finer discretization of the domain (grid) leads to higher accuracy. The result of this is a system of linear equations with many more unknowns/parameters (i.e. the solution vector has a higher dimension); thus, a larger matrix system has to be solved to obtain the PDE solution on this fine grid. This is a major setback for real-time applications, and other settings where the PDE has to be queried multiple times. Reduced Order Models (ROM) offer a solution as they seek to reduce the dimension of the solution vector while maintaining the problem's physical features. The Reduced Basis (RB) method, which has received a lot of attention in the last decade [73, 121, 174, 253, 259, 272] but can be traced back to the 1980s [86, 248, 257], is unarguably one of the most popular ROM. This method consists of an offline and an online stage. During the offline stage, a reduced basis is obtained from a good choice of parameters, and this is used to obtain solutions of the PDE for new parameters. This is very similar to neural operator methods for solving PDEs like Fourier Neural Operator (FNO) [211] and Deep operator network (DeepONet) [222]. The RB method can also be extended for parameter identification tasks [219] as well as inverse problems [97].

Recently, Deep Learning-based reduced order models (DL-ROM) have been popularized to efficiently solve PDEs [91, 101, 203]. Just like the RB method, they consist of an offline (training) phase and an online (testing) phase. The DL-ROM, though time-efficient during testing, might be very costly during training due to the high number of features or dimensions of the input and/or output – similar to RB method. The consequence of this is usually a network with more parameters, and thus more time is needed for optimizing these parameters. A common solution that reduces the number of network parameters while maintaining or even improving the accuracy is the proper orthogonal decomposition (POD). In the field of machine learning, this is commonly known as Principal Component Analysis (PCA), used as a technique for dimensionality reduction [318]. Reduced order models constructed with both deep learning and POD are referred to in [94] as POD-DL-ROM, where accuracy-wise, they are reported to outperform state-of-the-art POD-Galerkin ROMs during the testing stage; and efficiency-wise, they outperform DL-ROMs during the training stage.

5.1.2 Neural Networks and PDEs

Neural Networks have shown interesting results in dealing with high dimensional complex PDEs [130], where they overcome the curse of dimensionality for the Schrödinger equation and Hamilton–Jacobi–Bellman equation [150], Black–Scholes equations [23, 108], and Kolmogorov equations [161] which arise in option pricing [81].

The popularity of neural networks in solving PDEs probably comes from the famous Physics-informed neural networks in [261] that use a neural network to approximate a function, i.e. the solution of the PDE for a single parameter instance. Similar works include quadratic residual networks [39] and Deep Ritz networks [327].

Another class of neural networks – probably closer in its operation to RB methods – approximate an operator by a neural network. They are known as neural operators and can be used to query solutions of different parameter instances when trained. The PCA-based neural operator [24], FNO, DeepONet are part of this class as well as other novel methods and 'variants' like the Multiwavelet-based operator [119], graph neural operator [212], wavelet neural operator [302], and many more. [244] provides a good overview and extends them for parametric studies as well as inverse problems.

5.2 Problem Formulation

To shorten the design cycle of vehicles and reduce the cost of development, the automotive industry employs numerical simulation tools in the vehicle development process for testing and analysis. In this application example, we are interested in the interaction of vehicles with various roadbeds such as sand, snow, mud, etc. Vehicle stability depends largely on this interaction, and the safety of the passengers is thus a concern. To approach this problem, a full-body model of the vehicle dynamics is needed as well as proper modeling of the roadbed. Of interest to us, is the modeling of the roadbed consisting of granular material. This is a continuum mechanics problem that involves not only the well-known conservation equations of mass, momentum, and energy, but also a supplementary phenomenological material model. While the former specify the process conditions and are generally well understood, the latter relates the applied strain to the resulting stress and comes with uncertainties as well as non-linearities. Obviously, the overall goal is for the simulations to match the real-life experiments, thus the selected material model is of great importance.

5.2.1 Barodesy Model

Material models have parameters that are specific to the considered material as well as its reaction to external conditions, and these models range from simple to complex. By using single-parametric models for the granular material (roadbed), for example, the deviation between simulations and experiments increases as the simulation time progresses. As a result, complex material models with many more parameters are used. Such parameters

are usually determined by a great wealth of expert knowledge, and costly experiments. The barodesy model [183, 184] is one of such complex material models which conforms to the basic mechanical properties of the material. It is formulated in tensorial form by Equations (5.1)–(5.2)

$$\frac{d\mathbf{S}}{dt} = \mathbf{WS} - \mathbf{SW} + \mathbf{H}(\mathbf{S}, \mathbf{D}, e) \tag{5.1}$$

$$\frac{de}{dt} = (1 + e) \cdot \mathrm{tr}(\mathbf{D}), \tag{5.2}$$

with

$$\mathbf{D} = \frac{1}{2}\left(\nabla \mathbf{v}^{\mathrm{T}} + \left(\nabla \mathbf{v}^{\mathrm{T}}\right)^{\mathrm{T}}\right)$$
$$\mathbf{W} = \frac{1}{2}\left(\nabla \mathbf{v}^{\mathrm{T}} - \left(\nabla \mathbf{v}^{\mathrm{T}}\right)^{\mathrm{T}}\right)$$

and

$$\mathbf{H}(\mathbf{S}, \mathbf{D}, e) = h_b(\sigma) \cdot \left(f_b \mathbf{R}^0 + g_b \mathbf{S}^0\right) \cdot |\mathbf{D}|,$$

where

$$\begin{aligned}
\sigma &= |\mathbf{S}| = \sqrt{\mathrm{tr}\left(\mathbf{S}^2\right)} \\
\mathbf{S}^0 &= \mathbf{S}/|\mathbf{S}|,\ \mathbf{D}^0 = \mathbf{D}/|\mathbf{D}|,\ \mathbf{R}^0 = \mathbf{R}/|\mathbf{R}| \\
\mathbf{R} &= \mathrm{tr}\left(\mathbf{D}^0\right) \cdot \mathbf{I} + c_1 \cdot \exp\left(c_2 \cdot \mathbf{D}^0\right) \\
h_b &= \sigma^{c_3} \\
f_b &= c_4 \cdot \mathrm{tr}\left(\mathbf{D}^0\right) + c_5 \cdot (e - e_c) + c_6 \\
g_b &= -c_6 \\
e_c &= (1 + e_{c0}) \cdot \exp\left(\frac{\sigma^{1-c_3}}{c_4 \cdot (1 - c_3)}\right) - 1.
\end{aligned}$$

In the above expressions:

- $\mathbf{S} \in \ ^{3\times3}$ is the Cauchy stress tensor (with principal stresses $\sigma_1, \sigma_2, \sigma_3$ in axial and lateral directions),
- $\mathbf{W}$ is the antisymmetric part of the velocity gradient,
- $\mathbf{D}$ is the stretching tensor (the symmetric part of the velocity gradient),
- $e = V_p/V_s$ is the void ratio with critical void ratio e_c, where V_p and V_s are the volume of pores and solids (grains).
- $\mathbf{v} \in \ ^3$ is the velocity field.

The non-linear function $\mathbf{H}$ introduces the material parameters $c_1, c_2, c_3, c_4, c_5, c_6$, and e_{c0} which we seek to identify via deep learning in a supervised learning task, provided the stress is known. For Hostun sand [71], for example, $c_1 = -1.7637, c_2 = -1.0249, c_3 = 0.5517, c_4 = -1174, c_5 = -4175, c_6 = 2218, e_{c0} = 0.8703$.

5.2.2 Oedometric Test

In soil mechanics, laboratory tests are used to measure the physical and mechanical properties of soil. They enable the testing and validation of material models. The tests vary from soil classification, shear strength, consolidation, and permeability tests, etc. [88]. The consolidation or oedometric test is one of the most conducted tests in soil mechanics. The soil (material) sample is loaded as well as unloaded in axial direction and rigid side walls prevent any lateral expansion, see Figure 5.1. With this, the soil's consolidation properties can be measured.

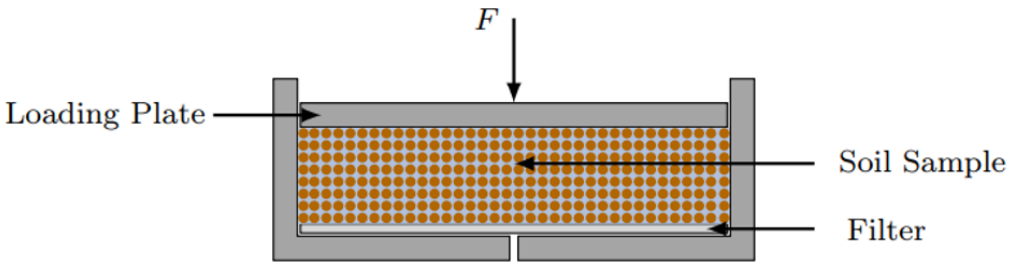


Figure 5.1: Schematic illustration of an oedometric test, see [251].

The laboratory measurements of oedometric tests result in stress paths (relating lateral and axial stress) and stress-strain-curves, e.g. in axial direction illustrated in Figure 5.2. These are compared to corresponding element tests wrt. a material model such as barodesy, in which the material model is integrated for one numerical point. When evaluating the quality of 3D numerical methods, only the comparison with corresponding element tests should be made, since the numerics cannot be better than the material model itself. This was investigated, for example, in [231] for the MESHFREE software (see Section 5.2.3), at that time still referred to as Finite Pointset Method (FPM).

5.2.3 MESHFREE and the Generalized Finite Difference Method

We employ the Generalized Finite Difference Method (GFDM) [191] implemented by Fraunhofer ITWM in the MESHFREE software [156, 192] to numerically solve coupled PDEs governed by the conservation equations and material models such as the barodesy model described in Section 5.2.1. MESHFREE has successfully been applied for the simulation of complex continuum mechanics problems in industry, like vehicles traveling through water[160], flow inside impulse-type turbines [193], solution mining [232], injection molding [307], wet metal cutting [304], and phase change processes [189].

Point Clouds and Generalized Finite Difference Approximation

An overview on point cloud generation for meshfree methods is given in [291]. MESHFREE employs an advancing front procedure [232] that first discretizes the boundary and then

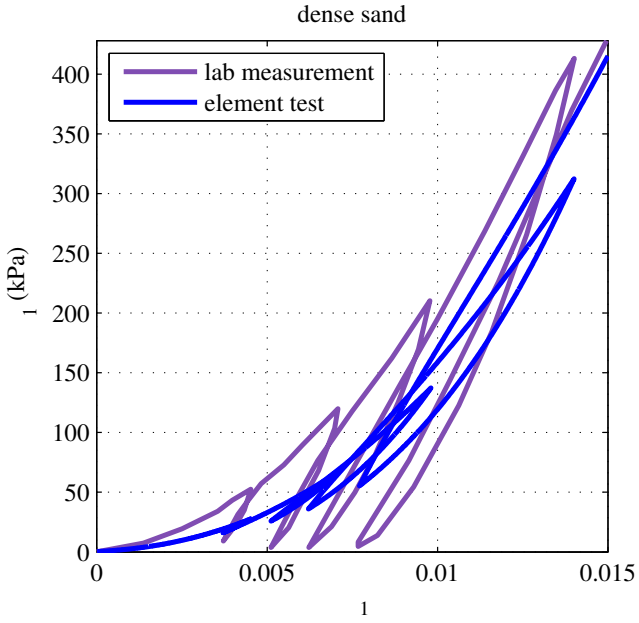


Figure 5.2: Axial stress-strain-curve of an oedometric test, see [231], with axial stress $-\sigma_1$ and axial strain ε_1.

iteratively the interior of the continuum domain depending on a given point interaction radius. Each point carries the physical information (such as velocity, pressure, temperature, stress, etc.) and is moved with the continuum velocity in a Lagrangian formulation [292]. Distortions caused by the movement can be corrected purely locally by adding and deleting points.

Discretizing the governing PDEs in their strong formulation, GFDM generalizes classical finite differences to (scattered/irregular) point clouds. Thereby, all numerical derivatives (function values, x-, y-, z-derivatives or Laplacian) are computed as linear combination of neighboring function values, where the neighbors of a point are determined by the point's interaction radius. The necessary coefficients/stencils are computed by a weighted least squares method. For more details on generalized finite difference approximation we refer to [193, 232, 290].

Data Generation

Using the MESHFREE software, we generate parameters-stress pairs to train our neural network. Here, we use the physical and numerical model presented in [251] including corresponding boundary conditions for the cylindrical oedometric test. As described in [231], the axial stress on the 3D point cloud (Figure 5.3) is averaged over all points of the sample to determine the resulting data for a parameters-stress pair, see Figure 5.4. For simplicity, the representation in this figure is dependent on time and not on axial strain as in Figure 5.2. Note that we use the settings for the dense sample in [231] with

fixed interaction radius $h = 0.01\text{m}$, loading/unloading rate $v_\text{p} = \mp 0.001 \frac{\text{m}}{\text{s}}$, and fixed time step size $\Delta t = 0.0015\text{s}$ for all parameters-stress pairs. The choice of the parameters that

Table 5.1: Value bounds for sampling of the parameters for the data generation

Parameters	Base value (**BV**)	Lower bound (**BV** − 5%)	Upper bound (**BV** + 5%)
c_1	-1.7637×10^0	-1.8519×10^0	-1.6755×10^0
c_2	-1.0249×10^0	-0.1076×10^0	-0.9737×10^0
c_3	0.5517×10^0	0.5241×10^0	0.5793×10^0
c_4	-1.1740×10^3	-1.2327×10^3	-1.1153×10^3
c_5	-4.1750×10^3	-4.3838×10^3	-3.9663×10^3
c_6	2.2180×10^3	2.1071×10^3	2.3289×10^3
ec_0	0.8703×10^0	0.8268×10^0	0.9138×10^0

constitute the data set are selected uniformly within predefined intervals. Guided by expert knowledge (see Section 5.2.1), a base value is selected and the interval is constructed around it by adding and subtracting 5% of this base value to obtain the lower and upper bounds of this interval as shown in Table 5.1.

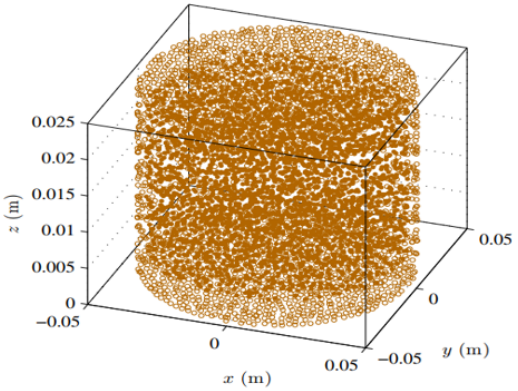


Figure 5.3: Initial 3D MESHFREE point cloud for the cylindrical oedometric test (filled circles: interior points, non-filled circles: boundary points), see [251].

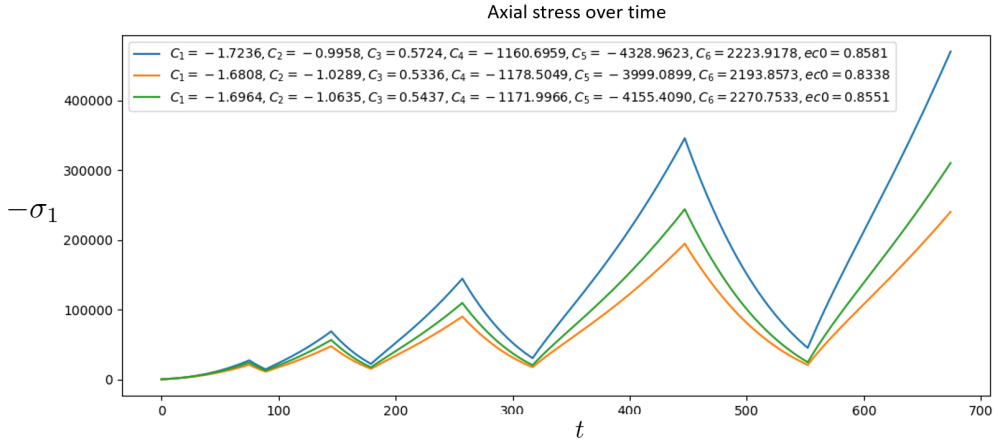


Figure 5.4: Example of generated input data samples for different parameters.

5.3 Proposed Method

The proposed method is inspired by both Reduced Order Models (ROM) and Neural Networks (NN). ROMs have been popular for a long time in dealing with PDEs, and even more when dealing with parameter identification problems, as outlined in Section 5.1.1. NNs have become popular over recent years not only due to their success in computer vision [190], natural language processing [140], but also due to the availability of data and growing computing power [103, 201]. The efficient combination of both methods [24] has already achieved remarkable results not only in simple problems but also in more complex problems such as cardiac electrophysiology [95] (where the use of proper orthogonal decomposition (POD) further improves the results [94]), fluid flow [93], non-linear models [61, 92], etc.

5.3.1 PCA-NN

We implement a variation of the PCA-NN architecture presented in [24], which uses a meshless operator for the evaluation of the solution of a PDE by combining ideas of ROM with deep learning. First, for given training data (λ_i, u_i), obtain a model reduction by the use of principal component analysis (PCA) for both the input (parameter λ) and output (solution u). Only the coefficients of a finite number of PCA components are retained. Thus, PCA reduces the dimensions of both the input and output spaces to finite dimensional latent spaces. Second, use a NN to map the coefficients of the respective representations in these latent spaces.

The evaluation of this operator approximation for a novel parameter λ is highly efficient: compute the scalar products with the specified finite number of PCA components, map these coefficients to the latent coefficients of the output space with the NN, approximate the solution of the PDE by an expansion using these coefficients and the PCA on the output side. A simplified architecture of this method is shown in Figure 3.6.

The formulation of this approach is in a function space setting and hence mesh-free. For implementation purposes, however, we have to specify how to compute the scalar products with the PCA components. These are only given numerically, usually by their values specified at discrete points (in our case time steps).

This PCA-NN operator has been used in [188, 217] in a multiscale plasticity problem to map strain to stress.

5.3.2 Workflow

In our problem, the goal is to learn the parameters $\mu \in \ ^{d_\mu}$, with $d_\mu = 7$, from the variation of the axial stress $-\sigma_1 \in \ ^{d}$ over time t, where $d = 675$ is the fixed number of time steps corresponding to $\Delta t = 0.0015$, see Section 5.2.3. The data set generated from MESHFREE is therefore a vector pair $(\mu,-\sigma_1)$.

Our adopted procedure can be broken down into four major steps as illustrated in Figure 5.5:

- *Data Generation*: Using MESHFREE and the setup described in Section 5.2.3, generate parameters-stress pairs $(\mu^i, -\sigma_1^i)$ with $i = 1, 2, \ldots, N_{\text{train}} + N_{\text{test}}$. These are snapshots of the full order model that is based on the GFDM described in Section 5.2.3.
- *Training (Offline Stage)*: The first N_{train} data pairs are used to train the PCA-NN neural network. During training, the average L^2-loss

$$L_i(\mu, \hat{\mu}) = \frac{1}{d_\mu} \sum_{\ell=1}^{d_\mu} \left\| \frac{\mu_\ell^i - \hat{\mu}_\ell^i}{\mu_\ell^i} \right\|_2 \tag{5.3}$$

  is obtained and its average over the training data

$$L(\mu, \hat{\mu}) = \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} L_i(\mu, \hat{\mu}) \tag{5.4}$$

  is optimized, see Algorithm 3 in Section 5.3.2 for further details. $\hat{\mu}$ is the output of the model which is a composition of PCA applied on the axial stress $-\sigma_1$ followed by the neural network.
- *Testing (Online Stage)*: Once the network is trained, it is used for testing with the next N_{test} unseen data. Testing proceeds as shown in Algorithm 4 in Section 5.3.2. The network's performance is evaluated with the loss function given in Equation 5.4, but averaged over the N_{test} parameters by

$$L(\mu, \hat{\mu}) = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} L_i(\mu, \hat{\mu}). \tag{5.5}$$

- *Verification Stage (optional)*: This stage is used to ascertain the efficiency of the proposed model. Here, the material parameters $\hat{\mu}$ learned from the neural network are used as input to MESHFREE simulations, in order to compare the resulting stress $-\hat{\sigma}_1$ with the stress $-\sigma_1$ obtained from the ground truth parameters μ. The difference is measured using the relative L^2-error given by

$$E(-\sigma_1, -\hat{\sigma}_1) = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} E_i(-\sigma_1, -\hat{\sigma}_1), \tag{5.6}$$

  where

$$E_i(-\sigma_1, -\hat{\sigma}_1) = \left\| \frac{\sigma_1^i - \hat{\sigma}_1^i}{\sigma_1^i} \right\|_2. \tag{5.7}$$

Network Architecture

In our numerical examples, we followed the outline described in [24] and used a fully connected feed-forward neural network (FCN) for the mapping of the stress latent space (output of PCA on stress) to the parameters. The number of nodes per layer starts from $d, 500, 1000, 2000, 1000, 500$, and finally d_μ (which is the number of parameters to be learned, here 7). d is considered a hyperparameter, which has to be tuned. In our case, $d = 50$ lead to the best results. For the PCA, we use standard randomized singular vector decomposition (SVD) implementations [126, 268]. Figure 5.5 illustrates the overall PCA-NN architecture.

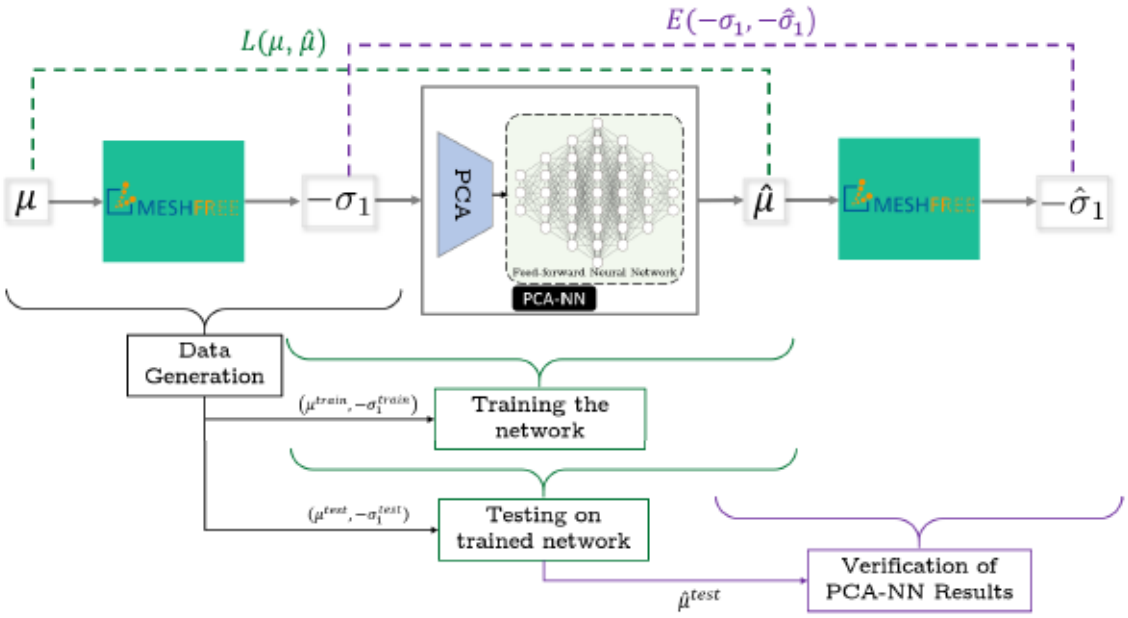


Figure 5.5: Complete workflow of the proposed approach.

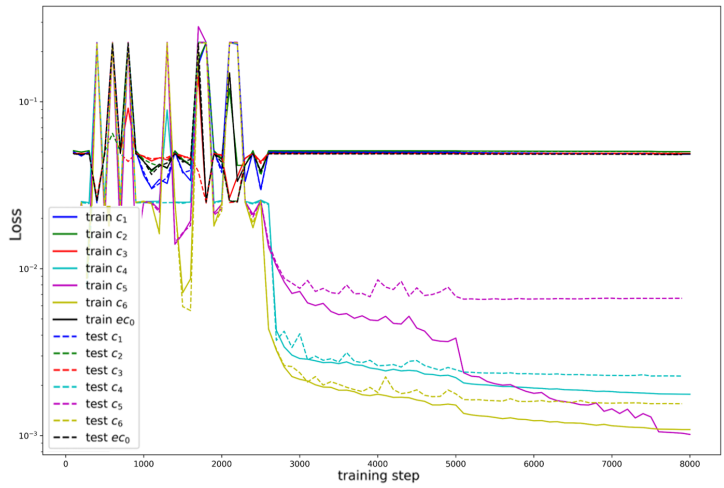
Algorithm

As a purely data-driven method, no physics or PDE is needed in the training of the neural network. However, the data used to train the network is obtained from MESHFREE's GFDM for solving the underlying PDE. By training the network with these numerically-given input-output pairs, we obtain a neural operator that solves the PDE for various instances irrespective of the underlying discretization.

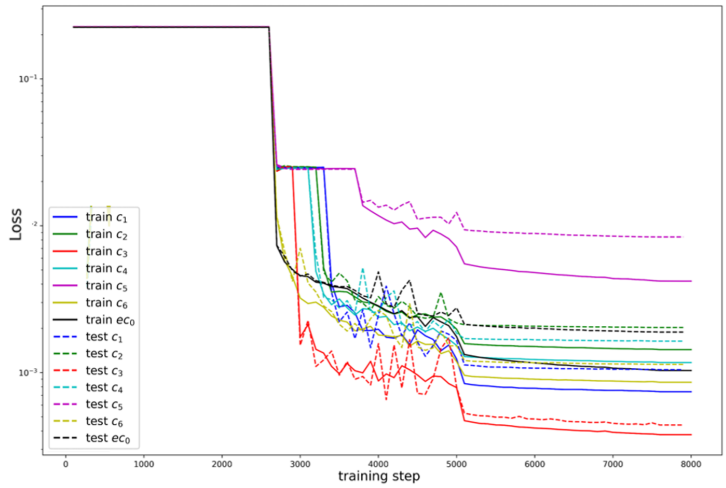
We specify the algorithm for the continuum mechanics problem described in Section 5.2.1 using the barodesy model. The training data is the pair $(\mu^i, -\sigma_1^i)$, with each $\mu^i \in \mathbb{R}^{d_\mu}$ and $-\sigma_1^i \in \mathbb{R}^d$. Training then proceeds as in Algorithm 3, while testing of the trained network proceeds as in Algorithm 4. Adaptation of this algorithm for this specific problem is available in the paper from the author at [299]

5.4 Numerical Results

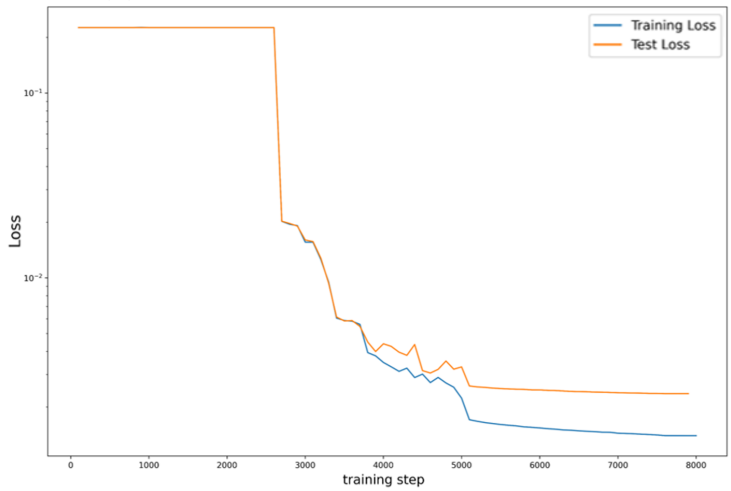
Randomized PCA is used to reduce the dimensions of the principal stress component from 675 to 50. This reduced dimension is the input to the FCN similar to that in Section 5.2. Because the parameter space is low enough, there is no need of a PCA after the FCN. The output of the FCN yields the target parameters directly. Of the 6000 data pairs generated, 75% is used for training. During training, the relative L^2-error of the individual parameters is evaluated and their average is the loss function minimized for optimizing the parameters of the neural network. However, due to the nature of this loss function, the learning of the parameters of higher magnitude is favored during training as can be seen in Figure 5.6a. We observe that the loss for parameters c_4, c_5, and c_6 (that are all of the order of



(a) Individual parameters with unscaled parameters.



(b) Individual parameters with scaled parameters.



(c) Average loss, with scaled parameters.

Figure 5.6: Loss during training on both training and test data

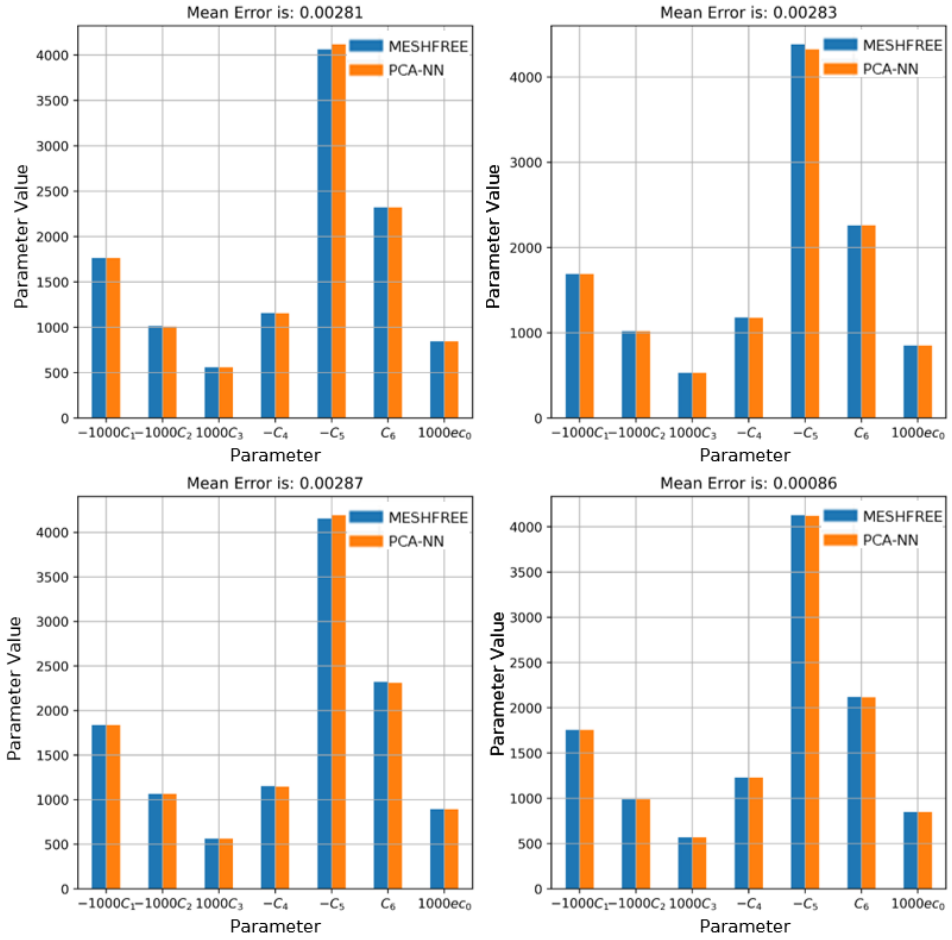


Figure 5.7: Comparison of ground truth (MESHFREE simulation in blue) and learned parameters (PCA-NN in orange) for four randomly chosen examples of the oedometric test.

1000) is minimized, while for the other parameters (that are of the order of 1) the loss is almost not minimized. As a remedy, the parameters of lower magnitude are scaled such that they are of the same order (of 1000) as the parameters of higher magnitude. In this way, learning of all individual parameters is achieved as shown in Figure 5.6b. Figure 5.6c illustrates the overall loss as average of the individual losses.

We obtained an average relative L^2-error of 2.63×10^{-3} on the test data set. Figure 5.7 shows the comparison of the ground truth (input to the MESHFREE simulation in blue) and the learned parameters (PCA-NN in orange) for four randomly selected examples. The learned parameters of these four examples were further used in a verification step in order to compare the resulting MESHFREE output axial stress with that produced by the ground truth parameters. The average relative error obtained was 4.12×10^{-3}. This is illustrated in Figure 5.8 (top), where there is an obvious overlap of the axial stresses from the learned parameters with those from the ground truth parameters. Figure 5.8 (bottom) shows the corresponding relative L^2-errors.

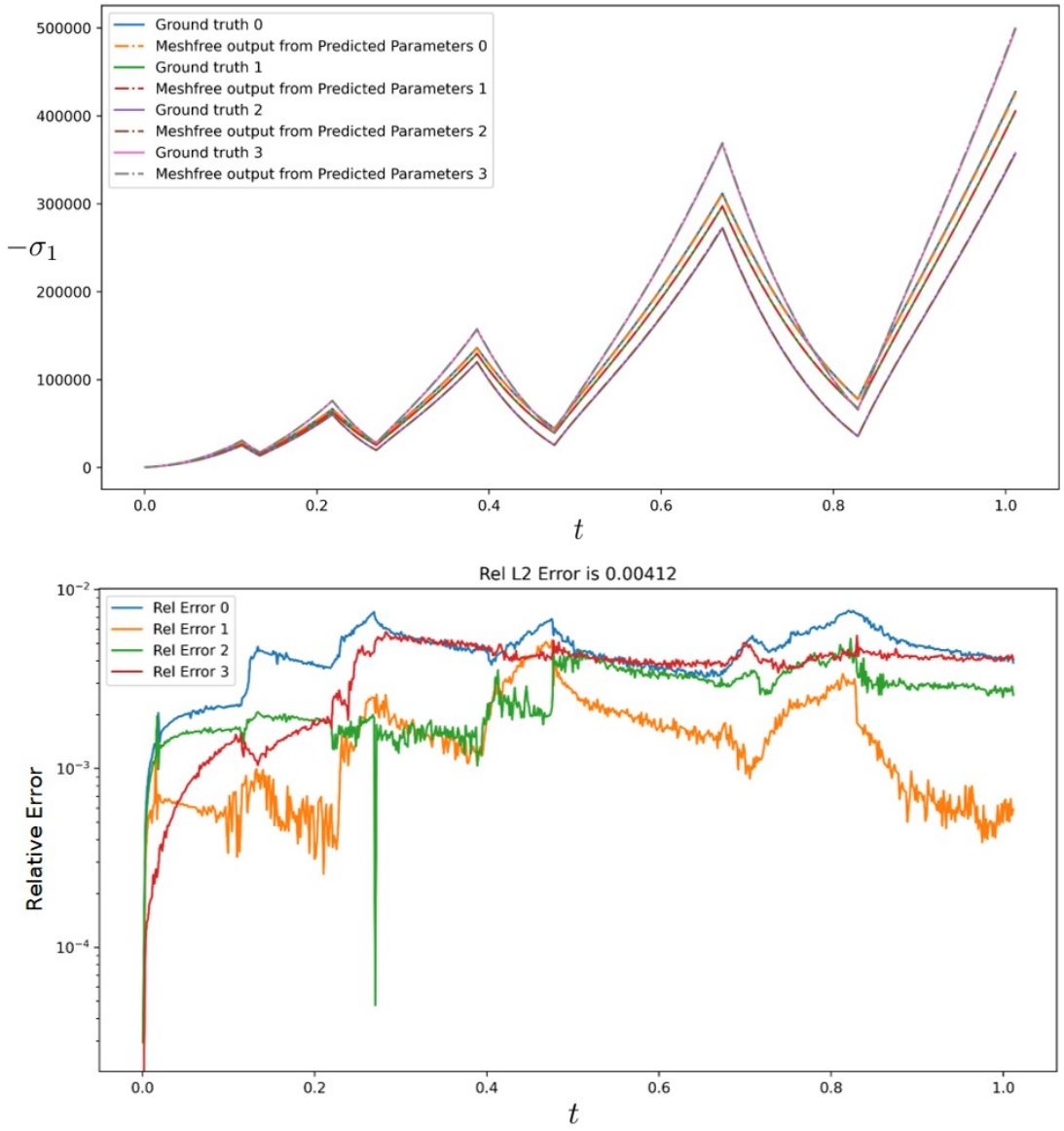


Figure 5.8: Comparison of MESHFREE outputs using ground truth parameters and learned parameters for four randomly chosen examples of the oedometric test (top) as well as corresponding relative L^2-errors.

5.5 Conclusions and Outlook

The presented results highlight the potential of deep learning in continuum mechanics, specifically in material parameters identification for complex material models – a task that up till now depends heavily on expert knowledge if not trial and error. By exploiting deep learning methods, we obtain the model parameters from MESHFREE simulations. It will be equally interesting to see how the results change when experimental data is used instead of or in addition to simulation data.

The proposed method is an important first step since simulation and experimental results

are almost always noisy in real-life problems. An interesting future study will be to look at the effect of different noise levels on the neural network's strength in parameter identification. This is a common practice in the field of inverse problems. For example, [244] studied the effects of noise on both function-approximating networks and neural operators for PDEs. There, the PCA-based method–when fed with noise–did not deviate so much from the noiseless case for increasing noise level. This is also promising for our application problem.

Anyone who attempts to generate random numbers by deterministic means is, of course, living in a state of sin.

— *John von Neumann*

6

Solving the Electrical Impedance Tomography Inverse Problem

A famous PDE-based inverse problem–the Calderón problem–is studied in this chapter. Both analytical methods and DL-based methods are used to solve this problem, and insights are drawn thereof. It is based on the following article:

> Derick Nganyu Tanyu, Jianfeng Ning, Andreas Hauptmann, Bangti Jin and Peter Maass. "Electrical Impedance Tomography: A Fair Comparative Study on Deep Learning and Analytic-based Approaches". In: *Manuscript submitted for publication (Nov. 2023). Preprint to be made available on arXiv.*

6.1 Introduction and motivation

This chapter investigates deep learning concepts for the continuous model of electrical impedance tomography (EIT). EIT is one of the most intensively studied inverse problems, and there already exists a very rich body of literature on various aspects [31, 305]. EIT as an imaging modality is of considerable practical interest in noninvasive imaging and non-destructive testing. For example, the reconstruction can be used for diagnostic purposes in medical applications, e.g. monitoring of lung function, detection of cancer in the skin and breast and location of epileptic foci [143]. Similarly, in geophysics, one uses electrodes on the surface of the earth or in boreholes to locate resistivity anomalies, e.g. minerals or contaminated sites, and it is known as geophysical resistivity tomography in the literature.

Since its first formulation by Calderón [41], the issue of image reconstruction has received enormous attention, and many reconstruction algorithms have been proposed based on regularised reconstructions, e.g., Sobolev smoothness, total variation and sparsity. Due to the severe ill-posed nature of the inverse problem and the high degree of non-linearity

of the forward model, the resolution of the obtained reconstructions has been modest at best. Nonetheless, the last years have witnessed significant improvement in the EIT reconstruction regarding resolution and speed. This impressive progress was primarily driven by recent innovations in deep learning, especially deep neural network architectures, high-quality paired training data, efficient training algorithms (e.g., Adam), and powerful computing facilities, e.g., graphical processing units (GPUs).

This study aims to comprehensively and fairly compare deep learning techniques for solving the EIT inverse problem. This study has several sources of motivation. First, the classical, analytical setting of EIT is severely ill-posed, to such an extent that it allows only rather sketchy reconstructions when employing classical regularisation schemes. Unless one utilises additional *a priori* information, there is no way around the ill-posedness. This has motivated the application of learning concepts in this context. Incorporating additional information in the form of typical data sets and ground truth reconstructions allows constructing an approximation of a data manifold specific to the task at hand. The structures that distinguish these manifolds are typically hard to capture by explicit physical-mathematical models. To some extent, TV- or sparsity-based Tikhonov functionals exploit these features. However, learning the prior distribution from sufficiently large sets of training data potentially offers much greater flexibility than these hand-crafted priors. Second, there already exists a growing and rich body of literature on learned concepts for EIT; see, e.g., the recent survey [176] and Section 6.3 for a detailed description of the state of the art. Nevertheless, most of these works focus on their own approaches, typically showing their superiority compared to somewhat standard and basic analytical methods. In contrast, we aim at a fair and more comprehensive comparison of different learned concepts and include a comparison with two advanced analytical methods (i.e., D-bar and sparsity methods).

It is worth mentioning that inverse problems pose a particular challenge for learned concepts due to their inherent instability. For example, directly adapting well-established network architectures, which have been successfully applied to computer vision or imaging problems, typically fail for inverse problems, e.g., medical image reconstruction tasks. Hence, such learned concepts for inverse parameter identification problems are most interesting in terms of developing an underlying theory and the performance on practical applications. Indeed, the research on learned concepts for inverse problems has exploded over the past years, see e.g. the review [10] and the references cited therein for a recent overview of the state of the art. Arguably, the two most prominent fields of application for inverse problems are PDE-based parameter identification problems and tasks in tomographic image reconstruction. These fields actually overlap, e.g. when it comes to parameter identification problems in PDE-based multi-physics models for imaging. The most common examples in tomography are X-ray tomography (linear) and EIT (non-linear). Hence, one may also regard this study as being prototypical of how deep learning concepts should be evaluated in the context of non-linear PDE inverse problems.

The rest of the chapter is organised as follows. In Section 6.2, we describe the continuum model for EIT, and also two prominent analytic-based approaches for EIT reconstruction, i.e., sparsity and D-bar method. Then, in Section 6.3, we describe four representative deep learning-based approaches for EIT imaging. Finally, in Section 6.4, we present an extensive array of experiments with a suite of performance metrics to shed insights into the relative merits of the methods. We conclude with further discussions in Section 6.5.

6.2 Electrical impedance tomography

Mathematically speaking, the continuous EIT problem aims at determining a spatially-varying electrical conductivity σ within a bounded domain Ω by using measurements of the electrical potential on the boundary $\partial\Omega$. The basic mathematical model for the forward problem is the following elliptic PDE:

$$-\mathrm{div}(\sigma\nabla u) = 0, \quad \text{in } \Omega, \tag{6.1}$$

subject to a Neumann boundary condition $\sigma\frac{\partial u}{\partial n} = j$ on $\partial\Omega$, which satisfies a compatibility condition $\int_{\partial\Omega} j\mathrm{d}S = 0$. An EIT experiment consists of applying an electrical current j on the boundary $\partial\Omega$ and measuring the resulting electrical potential $\phi = u|_{\partial\Omega}$ on $\partial\Omega$. The Neumann to Dirichlet (NtD) operator $\Lambda_{\sigma,N} : j \mapsto \phi$ maps a Neumann boundary condition j to the Dirichlet data $\phi = u|_{\partial\Omega}$ on $\partial\Omega$.

In practice, several input currents are injected, and the induced electrical potentials are measured; see [55, 163] for discussions on the choice of optimal input currents. This data contains information about the underlying NtD map $\Lambda_{\sigma,N}$. The inverse problem is to determine or at least to approximate the true unknown physical electrical conductivity $\sigma^\dagger$ from a partial knowledge of the map. This inverse problem was first formulated by Calderón [41], who also gave a uniqueness result for the linearised problem. The mathematical theory of uniqueness of the inverse problem with the full NtD map $\Lambda_{\sigma,N}$ has received enormous attention, and many profound theoretical results have been obtained. For an in-depth overview of uniqueness results, we refer to the monograph [154] and survey [305].

6.2.1 Theoretical background

This section introduces the mathematical model of the EIT problem and the discrepancy functional used for reconstructing the conductivity σ. Let Ω be an open-bounded domain in ^{d} ($d \geq 2$) with a Lipschitz boundary $\partial\Omega$, and let $\Lambda_{\sigma,N}$ denote the NtD map of problem (6.1). We employ the usual Sobolev space for the Neumann boundary data $\sigma\frac{\partial u}{\partial n} = j \in \tilde{H}^{-\frac{1}{2}}(\partial\Omega)$, respectively Dirichlet boundary condition $u = \phi \in \tilde{H}^{\frac{1}{2}}(\partial\Omega)$ on $\partial\Omega$. Throughout, we make use of the space $\tilde{H}^1(\Omega)$, which is a subspace of the Sobolev space $H^1(\Omega)$ with vanishing mean on $\partial\Omega$, i.e., $\tilde{H}^1(\Omega) = \{v \in H^1(\Omega) : \int_{\partial\Omega} v\mathrm{d}s = 0\}$. The spaces $\tilde{H}^{\frac{1}{2}}(\partial\Omega)$ and $\tilde{H}^{-\frac{1}{2}}(\partial\Omega)$ are defined similarly. These spaces are equipped with the usual

norms. We normalise the solution of the Neumann problem by enforcing $\int_{\partial\Omega} u ds = 0$, so that there exists a unique solution $u \in \tilde{H}^1(\Omega)$. We denote the Dirichlet-to-Neumann (DtN) map by $\Lambda_{\sigma,D}$. Then we have $\Lambda_{\sigma,N} = \Lambda_{\sigma,D}^{-1}$, i.e., DtN and NtD maps are inverse to each other. In usual regularised reconstruction, we employ the NtD map $\Lambda_{\sigma,N}$, whereas in the D-bar method, we employ the DtN map $\Lambda_{\sigma,D}$.

An EIT experiment consists of applying a current j and measuring the resulting potential ϕ on $\partial\Omega$, and it is equivalent to solving a Neumann forward problem with the physical conductivity $\sigma^\dagger$, i.e. $\phi = \Lambda_{\sigma,N} j$, on $\partial\Omega$. In practice, the boundary potential measurements are collected experimentally, and thus ϕ is only an element of the space $L^2(\Gamma)$. see e.g. [44]. Note that the continuum model is mostly academic. A more realistic model is the so-called complete electrode model (CEM) for EIT [152, 286], which models contact impedances and localised electrode geometries. The CEM is finite-dimensional by construction, leading to different mathematical challenges and reconstruction methods.

The solvability, uniqueness and smoothness of the continuum model with respect to L^p norms can be derived using Meyers' gradient estimate [229], as in [270].

Theorem 6.2.1. *Let Ω be a bounded Lipschitz domain in ^{d} ($d \geq 2$). Assume that $\sigma \in L^\infty(\Omega)$ satisfies $\lambda < \sigma < \lambda^{-1}$ for some fixed $\lambda \in (0,1)$. For $f \in (L^q(\Omega))^d$ and $h \in L^q(\Omega)$, let $u \in H^1(\Omega)$ be a weak solution of*

$$-\mathrm{div}(\sigma \nabla u) = -\mathrm{div}(f) + h \quad \text{in } \Omega.$$

Then, there exists a constant $Q \in (2, +\infty)$ depending on λ and d only, $Q \to 2$ as $\lambda \to 0$ and $Q \to \infty$ as $\lambda \to 1$, such that for any $2 < q < Q$, we obtain $u \in W^{1,q}_{\mathrm{loc}}(\Omega)$ and for any $\Omega_1 \subset\subset \Omega$

$$\|u\|_{W^{1,q}(\Omega_1)} \leq C(\|u\|_{H^1(\Omega)} + \|f\|_{L^q(\Omega)} + \|h\|_{L^q(\Omega)}),$$

where the constant C depends on λ, d, q, Ω_1 and Ω.

In Theorem 6.2.1, the boundary condition for the problem can be general. Its effect enters the $W^{1,q}$-estimate through the term $\|u\|_{H^1(\Omega)}$. In addition, no regularity has been assumed on σ. Generally, a precise estimate of the constant $Q(\lambda, d)$ is missing, but in the 2D case, a fairly sharp estimate of $Q(\lambda, d)$ was derived in [11].

6.2.2 Conventional EIT reconstruction algorithms

EIT suffers from a high degree of non-linearity and severe ill-posedness, as typical of many PDE inverse problems with boundary data. However, its potential applications have sparked much interest in designing effective numerical techniques for its efficient solution. Numerous numerical methods have been proposed in the literature; see [31, Section 7] for an overview (up to 2002). These methods can roughly be divided into two groups: regularised reconstruction and direct methods. Below, we give a brief categorisation of conventional reconstruction schemes.

The methods in the first group are of variational type, i.e., based on minimising a certain discrepancy functional. Commonly the discrepancy J is the standard least-squares fitting, i.e., the squared $L^2(\partial\Omega)$ norm of the difference between the electrical potential due to the applied current j and the measured potential ϕ:

$$J(\sigma) = \tfrac{1}{2}\|\Lambda_{\sigma,N} j - \phi^\delta\|^2_{L_2(\partial\Omega)},$$

for one single measurement (j, ϕ^δ). One early approach of this type is given in [56], which applies one step of a Newton method with a constant conductivity as the initial guess. Due to the severe ill-posedness of the problem, regularisation is beneficial for obtaining reconstructions with improved resolution [82, 155, 278]. Commonly used penalties include Sobolev smoothness [165, 227] for a smooth conductivity distribution, total variation [141], Mumford-Shah functional [270], level set method [60] for recovering piecewise constant conductivity, sparsity [100, 164, 166] for recovering small inclusions (relative to the background). The combined functional is given by

$$\Psi(\sigma) = J(\sigma) + \alpha R(\sigma),$$

where $R(\sigma)$ denotes the penalty, and $\alpha > 0$ is the penalty weight. The functional $\Psi(\sigma)$ is then minimised over the admissible set

$$\mathcal{A} = \{\sigma \in L^\infty(\Omega) : \lambda \leq \sigma \leq \lambda^{-1} \text{ a.e. } \Omega\},$$

for some $\lambda \in (0, 1)$. The set $\mathcal{A}$ is usually equipped with an $L^p(\Omega)$ norm $(1 \leq p \leq \infty)$. One may also employ data fitting other than the standard $L^2(\partial\Omega)$-norm. The most noteworthy one is the Kohn-Vogelius approach, which lifts the boundary data to the domain Ω and makes the fitting in Ω [32, 182, 317]; see also [179] for a variant of the Kohn-Vogelius functional. In practice, the regularized formulations have to be properly discretized, commonly done by means of finite element methods [99, 167, 168, 269], due to the spatially variable conductivity and irregular domain geometry. Newton-type methods have also been applied to EIT [199, 200]. Probabilistic formulations of these deterministic approaches are also possible [28, 78, 98, 173], which can provide uncertainty estimates on the reconstruction.

The methods in the second group are of a more direct nature, aiming at extracting relevant information from the given data directly, without going through the expensive iterative process. Bruhl et al [37, 38] developed the factorisation method for EIT, which provides a criterion for determining whether a point lies inside or outside the set of inclusions by carefully analysing the spectral properties of certain operators. Thus, the inclusions can be reconstructed directly by testing every point in the computational domain. The D-bar method of Siltanen, Mueller and Isaacson [240, 283] is based on Nachman's uniqueness proof [242] and utilises the complex geometric solutions and nonphysical scattering transform for direct image reconstruction. Chow, Ito and Zou [59] proposed the direct sampling method when there are only very few Cauchy data pairs. The method employs dipole potential

as the probing function and constructs an indicator function for imaging the inclusions in EIT, and it is easy to implement and computationally cheap. Other notable methods in the group include monotonicity method [131], enclosure method [153], Calderón's method [25, 281], and MUSIC [5, 6, 198] among others. Generally, direct methods are faster than those based on variational regularisation, but the reconstructions are often inferior in terms of resolution and can suffer from severe blurring.

These represent the most common model-based inversion techniques for EIT reconstruction. Despite these important progress and developments, the quality of images produced by EIT remains modest when compared with other imaging modalities. In particular, at present, EIT reconstruction algorithms are still unable to extract sufficiently useful information from data to be an established routine procedure in many medical applications. Moreover, the iterative schemes are generally time-consuming, especially for 3D problems. One possible way of improving the quality of information is to develop an increased focus on identifying useful information and fully exploiting a priori knowledge. This idea has been applied many times, and the recent advent of deep learning significantly expanded its horizon from hand-crafted regularisers to more complex and realistic learned schemes. Indeed, recently, deep learning-based approaches have been developed to address these challenges by drawing on knowledge encoded in the dataset or structural preference of the neural network architecture.

We describe the sparsity approach and D-bar method next, and deep learning approaches in Section 6.3.

6.2.3 Sparsity-based method

The sparsity concept is very useful for modelling conductivity distributions with "simple" descriptions away from the known background σ_0, e.g. when σ consists of an uninteresting background plus some small inclusions. Let $\delta\sigma^\dagger = \sigma^\dagger - \sigma_0$. A "simple" description means that $\delta\sigma$ has a sparse representation with respect to a certain basis/frame/dictionary $\{\psi_k\}$, i.e., there are only a few non-zero expansion coefficients. The ℓ^1 norm $\delta\sigma$ can promote the sparsity of $\delta\sigma$ [67]

$$\Psi(\sigma) = J(\sigma) + \alpha\|\delta\sigma\|_{\ell^1}, \quad \text{with } \|\delta\sigma\|_{\ell^1} = \sum_k |\langle \delta\sigma, \psi_k\rangle|. \tag{6.2}$$

Under certain regularity conditions on $\{\psi_k\}$, the problem of minimising Ψ over the set $\mathcal{A}$ is well-posed [165].

Optimisation problems with the ℓ^1 penalty have attracted intensive interest [30, 35, 67, 320]. The challenge lies in the non-smoothness of the ℓ^1-penalty and high-degree nonlinearity of the discrepancy $J(\sigma)$. The basic algorithm for updating the increment $\delta\sigma_i$ and $\sigma_i = \sigma_0 + \delta\sigma_i$ by minimising Ψ formally reads

$$\delta\sigma_{i+1} = \mathcal{S}_{s\alpha}(\delta\sigma_i - s\Lambda'^*_{\sigma_i,N}(\Lambda_{\sigma_i,N}j - \phi^\delta)),$$

where $s > 0$ is the step size, $\Lambda'_{\sigma_i,N}$ denotes the Gâteaux derivative of the NtD map $\Lambda_{\sigma_i,N}$ in σ, and $\mathcal{S}_\lambda(t) = \operatorname{sign}(t)\max(|t| - \lambda, 0)$ is the soft shrinkage operator. However, a direct application of the algorithm does not yield accurate results. We adopt the procedure in Algorithm 8. The key tasks include computing the gradient J' (Steps 4-5) and selecting the step size (Step 6).

Algorithm 8: Sparsity reconstruction for EIT.

Input: σ_0 and α
Result: an approximate minimiser $\delta\sigma$
1 Set $\delta\sigma_0 = 0$;
2 **for** $i \leftarrow 1, \ldots, I$ **do**
3 Compute $\sigma_i = \sigma_0 + \delta\sigma_i$;
4 Compute the gradient $J'(\sigma_i)$;
5 Compute the H_0^1-gradient $J'_s(\sigma_i)$;
6 Determine the step size s_i;
7 Update inhomogeneity by $\delta\sigma_{i+1} = \delta\sigma_i - s_i J'_s(\sigma_i)$;
8 Threshold $\delta\sigma_{i+1}$ by $\mathcal{S}_{s_i\alpha}(\delta\sigma_{i+1})$;
9 Check stopping criterion.
10 **end**

Gradient evaluation Evaluating the gradient $J'(\sigma) = -\nabla u(\sigma) \cdot \nabla p(\sigma)$ involves solving an adjoint problem

$$-\nabla \cdot (\sigma \nabla p) = 0, \quad \text{in } \Omega, \quad \text{with } \sigma \frac{\partial p}{\partial n} u(\sigma) - \phi^\delta \quad \text{on } \partial\Omega.$$

Note that Indeed, $J'(\sigma)$ is defined via duality mapping $J'(\sigma)[\lambda] = \langle J'(\sigma), \lambda \rangle_{L_2(\Omega)}$, and thus $J'(\sigma) \in (L^\infty(\Omega))'$ may be not smooth enough. Instead, we take the $H_0^1(\Omega)$ metric for σ, by defining $J'_s(\sigma)$ via $J'(\sigma)[\lambda] = \langle J'_s(\sigma), \lambda \rangle_{H_0^1(\Omega)}$. Integration by parts yields $-\Delta J'_s(\sigma) + J'_s(\sigma) = J'(\sigma)$ in Ω and $J'_s(\sigma) = 0$ on $\partial\Omega$. The assumption is that the inclusions are in the interior of Ω. J'_s is also known as Sobolev gradient [243] and is a smoothed version of the $L^2(\Omega)$-gradient. It metrises the set $\mathcal{A}$ by the $H_0^1(\Omega)$-norm, thereby implicitly restricting the admissible conductivity to a smoother subset. Numerically, evaluating the gradient $J'_s(\sigma)$ involves solving a Poisson problem and can be carried out efficiently. Using J'_s, we can locally approximate $\Psi(\sigma) = \Psi(\sigma_0 + \delta\sigma)$ by

$$\Psi(\sigma_0 + \delta\sigma) - \Psi(\sigma_0 + \delta\sigma_i) \sim \langle \delta\sigma - \delta\sigma_i, J'_s(\sigma_i) \rangle_{H^1(\Omega)} + \tfrac{1}{2s_i} \|\delta\sigma - \delta\sigma_i\|^2_{H^1(\Omega)} + \alpha \|\delta\sigma\|_{\ell^1},$$

which is equivalent to

$$\tfrac{1}{2s_i} \|\delta\sigma - (\delta\sigma_i - s_i J'_s(\sigma_i))\|^2_{H^1(\Omega)} + \alpha \|\delta\sigma\|_{\ell^1}. \tag{6.3}$$

Upon identifying $\delta\sigma$ with its expansion coefficients in $\{\psi_k\}$, the solution to problem (6.3) is given by

$$\delta\sigma_{i+1} = \mathcal{S}_{s_i\alpha}(\delta\sigma_i - s_i J'_s(\sigma_i)),$$

This step zeros out small coefficients, thereby promoting the sparsity of $\delta\sigma$.

Step size selection Usually, gradient-type algorithms suffer from slow convergence, e.g., steepest descent methods. One way to enhance its convergence is due to [17]. The idea is to mimic the Hessian with sI over the most recent steps so that $sI(\delta\sigma_i - \delta\sigma_{i-1}) \approx J'_s(\sigma_i) - J'_s(\sigma_{i-1})$ holds in a least-squares sense, i.e.,

$$s_i = \arg\min_s \|s(\delta\sigma_i - \delta\sigma_{i-1}) - (J'_s(\sigma_i) - J'_s(\sigma_{i-1}))\|^2_{H^1(\Omega)}.$$

This gives rise to one popular Barzilai-Borwein rule

$s_i = \langle \delta\sigma_i - \delta\sigma_{i-1}, J'_s(\sigma_i) - J'_s(\sigma_{i-1})\rangle_{H^1(\Omega)} / \langle \delta\sigma_i - \delta\sigma_{i-1}, \delta\sigma_i - \delta\sigma_{i-1}\rangle_{H^1(\Omega)}$ [17, 66]. In practice, following [320], we choose the step length s to enforce a weak monotonicity

$$\Psi(\sigma_0 + \mathcal{S}_{s\alpha}(\delta\sigma_i - sJ'_s(\sigma_i))) \leq \max_{i-M+1\leq k\leq i} \Psi(\sigma_k) - \tau\frac{s}{2}\|\mathcal{S}_{s\alpha}(\delta\sigma_i - sJ'_s(\sigma_i)) - \delta\sigma_i\|^2_{H^1(\Omega)},$$

where τ is a small number, and $M \geq 1$ is an integer. One may use the step size by the above rule as the initial guess at each inner iteration and then decrease it geometrically by a factor q until the weak monotonicity is satisfied. The iteration is stopped when s_i falls below a prespecified tolerance s_{stop} or when the maximum iteration number I is reached.

The above description follows closely the work [166], where the sparsity algorithm was first developed. There are alternative sparse reconstruction techniques, notably based on total variation [33, 99, 270, 333]. For example, [33] presented an experimental (in-vivo) evaluation of the total variation approach using a linearized model, and the resulting optimisation problem solved by the primal-dual interior point method; and the work [333] compared different optimisers. Due to the non-smoothness of the total variation, one may relax the formulation with the Modica-Mortola function in the sense of Gamma convergence [167, 270].

6.2.4 The D-bar method

The D-bar method of Siltanen, Mueller and Isaacson [283] is a direct reconstruction algorithm based on the uniqueness proof due to Nachman [242]; see also Novikov [249]. That is, a reconstruction is directly obtained from the DtN map $\Lambda_{\sigma,D}$, without going through an iterative process. Note that the DtN map $\Lambda_{\sigma,D}$ can be computed as the inverse of the measured NtD map $\Lambda_{\sigma,N}$ when full boundary data is available. Below we briefly overview the classic D-bar algorithm assuming $\sigma \in C^2(\Omega)$, with a positive lower bound (i.e., $\sigma \geq c > 0$ in Ω), and $\sigma \equiv 1$ in a neighbourhood of the boundary $\partial\Omega$. In this part, we consider an embedding of ^{2} in the complex plane, and hence we will identify planar points $x = (x_1, x_2)$ with the corresponding complex number $x_1 + \mathrm{i}x_2$, and the product kx denotes complex multiplication. For more detailed discussions, we refer interested readers to the survey [240].

First, we transform the conductivity equation (6.1) into a Schrödinger-type equation by substituting $\widetilde{u} = \sqrt{\sigma}u$ and setting $q = \Delta\sqrt{\sigma}/\sqrt{\sigma}$ and extending $\sigma \equiv 1$ outside Ω. Then we obtain

$$(-\Delta + q(x))\widetilde{u}(x) = 0, \quad \text{in} \quad ^2. \tag{6.4}$$

Next we introduce a class of special solutions of equation (6.4) due to Faddeev [84], the so-called complex geometrical optics (CGO) solutions $\psi(x,k)$, depending on a complex parameter $k \in \quad \setminus \{0\}$ and $x \in \quad ^2$. These exponentially behaving functions are key to the reconstruction. Specifically, given $q \in L^p(\ ^2)$, $1 < p < 2$, the CGO solutions $\psi(x,k)$ are defined as solutions to

$$(-\Delta + q(x))\psi(\cdot,k) = 0, \quad \text{in} \quad ^2,$$

satisfying the asymptotic condition $e^{-\mathrm{i}kx}\psi(x,k) - 1 \in W^{1,\tilde{p}}(\ ^2)$ with $2 < \tilde{p} < \infty$. These solutions are unique for $k \in \quad \setminus \{0\}$ as shown in [242, Theorem 1.1]. Then D-bar algorithm recovers the conductivity σ from the knowledge of the CGO solutions $\mu(x,k) = e^{-\mathrm{i}kx}\psi(x,k)$ at the limit $k \to 0$ [242, Section 3]

$$\lim_{k\to 0} \mu(x,k) = \sqrt{\sigma}, \quad x \in \Omega.$$

Numerically, one can substitute the limit by $k = 0$ and evaluate $\mu(x,0)$. The reconstruction of σ relies on the use of an intermediate object called non-physical scattering transform $\mathbf{t}$, defined by

$$\mathbf{t}(k) = \int_{\ ^2} e_k(x)\mu(x,k)q(x)\mathrm{d}x,$$

with $e_k(x) := \exp(\mathrm{i}(kx + \bar{k}\bar{x}))$, where over-bar denotes complex conjugate. Since μ is asymptotically close to one, $\mathbf{t}(k)$ is similar to the Fourier transform of $q(x)$. Meanwhile, we can obtain μ by solving the name-giving D-bar equation

$$\bar{\partial}_k \mu(x,k) = \frac{1}{4\pi\bar{k}}\mathbf{t}(k)e_{-k}(x)\overline{\mu(x,k)}, \quad k \neq 0, \tag{6.5}$$

where $\bar{\partial}_k = \frac{1}{2}(\frac{\partial}{\partial k_1} + \mathrm{i}\frac{\partial}{\partial k_2})$ is known as the D-bar operator. To solve the above equation, scattering transform $\mathbf{t}(k)$ is required, which we can not measure directly from the experiment, but $\mathbf{t}(k)$ can be represented using the DtN map. Indeed, using Alessandrini's identity [4], we get the boundary integral

$$\mathbf{t}(k) = \int_{\partial\Omega} e^{\mathrm{i}\bar{k}\bar{x}}(\Lambda_{\sigma,D} - \Lambda_{1,D})\psi(x,k)\mathrm{d}s.$$

Note that $\Lambda_{1,D}$ can be analytically computed, and only $\Lambda_{\sigma,D}$ needs to be obtained from the measurements. Here, we will employ a Born approximation using $\psi \approx e^{\mathrm{i}kx}$, leading to the linearised approximation

$$\mathbf{t}^{\exp}(k) \approx \int_{\partial\Omega} e^{\mathrm{i}\bar{k}\bar{x}}(\Lambda_{\sigma,D} - \Lambda_{1,D})e^{\mathrm{i}kx}\mathrm{d}s. \tag{6.6}$$

This linearised D-bar algorithm can be efficiently implemented. First, one computes the $\mathbf{t}^{\exp}(k)$ from the measured DtN map $\Lambda_{\sigma,D}$, and then one solves the D-bar equation (6.5). Note that the solutions of (6.5) are independent for each $x \in \Omega$ and one can efficiently parallelise over x. This leads to real-time implementations and is especially

relevant for time-critical applications, e.g., monitoring purposes. The fully nonlinear D-bar algorithm would require first computing ψ by solving a boundary integral equation and then computing the scattering transform $\mathbf{t}(k)$.

The above algorithm assumes infinite precision and noise-free data. When the data is noise corrupted with finite measurements, the measured DtN map $\Lambda_{\sigma,D}$ is not accurate, and then the computation of $\mathbf{t}(k)$ becomes exponentially unstable for $|k| > R$. Thus, for practical data, we need to restrict the computations to a certain frequency range so as to stably compute $\mathbf{t}(k)$. Below we choose $R = 5$ for noise-free data and $R = 4.5$, $R = 4$ for 1% and 5% noisy measurements, respectively. This strategy of reducing the cut-off radius for noisy measurements is shown to be a regularisation strategy [180]. The final algorithm can be summarised as outlined below in Algorithm 9.

Algorithm 9: D-bar algorithm using $\mathbf{t}^{\exp}$

Input: $\Lambda_{\sigma,D}$ and R
Result: Regularised reconstruction of σ
1 Compute analytic $\Lambda_{1,D}$;
2 Evaluate $\mathbf{t}^{\exp}(k)$ for $|k| < R$ by (6.6);
3 Solve the D-bar equation (6.5);
4 Obtain $\sigma(x) = \mu(x, 0)^2$ for $x \in \Omega$;

Besides the D-bar method, there are other analytic and direct reconstruction methods available, e.g., enclosure method [153], monotonicity method [131], direct sampling method [59], and Calderón's method [25, 281]. The common advantage of these approaches is their computational efficiency, but unfortunately, also the directly inherited exponential instability to noise. While there are strategies to deal with noise, e.g., reducing the cut-off radius, the reconstruction quality does suffer: the reconstructions tend to be overall smooth. Additionally, there may be theoretical limitations to the reconstructions that can be obtained. For example, for the classic D-bar algorithm, it is C^2 conductivities, and for the enclosure methods, we can only find the convex hull of all inclusions. Thus, it is very interesting to discuss how deep learning can help overcome these limitations.

6.3 Deep learning-based methods

The integration of deep learning techniques has significantly advanced EIT reconstruction. It has successfully addressed several challenges posed by the non-linearity and severe ill-posedness of the inverse problem, leading to improved quality and reconstruction accuracy. Researchers have achieved breakthroughs in noise reduction, edge retention, and spatial resolution, making EIT a more viable imaging modality in medical and industrial applications. This success is mainly attributed to the extraordinary approximation ability of DNNs and the use of a large amount of paired training datasets.

First, much effort has been put into designing DNNs architectures for directly learning the maps from the measured voltages U to conductivity distributions σ, i.e., training a

DNN $\mathcal{G}_\theta$ such that $\sigma \approx \mathcal{G}_\theta(U)$. Li et al. [209] proposed a four-layer DNN framework constituted of a stacked autoencoder and a logistic regression layer for EIT problems. Tan et al. [297] designed the network based on LeNet convolutional layers and refined it using pooling layers and dropout layers. Chen et al. [52] introduce a novel DNN using a fully connected layer to transform the measurement data to the image domain before a U-Net architecture, and [323] a DenseNet with multiscale convolution. Fan and Ying [85] proposed DNNs with compact architectures for the forward and inverse problems in 2D and 3D, exploiting the low-rank property of the EIT problem. Huang et al. [148] first reconstruct an initial guess using RBF networks, which is then fed into a U-Net for further refinement. [279], uses a variational autoencoder to obtain a low-dimensional representation of images, which is then mapped to a low dimension of the measured voltages as well. We refer to [51, 208, 264, 321] for more direct learning methods.

Second, combining traditional analytic-based methods and neural networks is also a popular idea. Abstractly, one employs an analytic operator $\mathcal{R}$ and a neural network $\mathcal{G}_\theta$ such that $\sigma \approx \mathcal{G}_\theta(\mathcal{R}(U))$. One example is the Deep D-bar method [128]. It first generates EIT images by the D-bar method, then employs the U-Net network to refine the initial images further. Along this line, one can design the input of the DNN from Calderón's method [43, 293], domain-current method [314], one-step Gauss-Newton algorithm [228] and conjugate gradient algorithm [331]. Inspired by the mathematical relationship between the Cauchy difference index functions in the direct sampling method, Guo and Jiang [117] proposed the DDSM proposed in [117] employs the Cauchy difference functions as the DNN input. Yet another popular class of deep learning-based methods that combines model-based approaches with learned components is based on the idea of unrolling, which replaces components of a classical iterative reconstructive method with a neural network learned from paired training data (see [236] for an overview). Chen et al. [50] proposed a multiple measurement vector (MMV) model-based learning algorithm (called MMV-Net) for recovering the frequency-dependent conductivity in multi-frequency electrical impedance tomography (mfEIT). It unfolds the update steps of the alternating direction method of multipliers for the MMV problem. The authors validated the approach on the Edinburgh mfEIT Dataset and a series of comprehensive experiments. See also [49] for a mask-guided spatial–temporal graph neural network (M-STGNN) to reconstruct mfEIT images in cell culture imaging. Unrolling approaches based on the Gauss-Newton have also been proposed, where an iterative updating network is learned for the explicitly computed Gauss-Newton updates [138] or a proximal type operator [62]. Likewise, a quasi-Newton method has been proposed by learning an updated singular value decomposition [285].

Reconstruction methods in these two groups are supervised in nature and rely heavily on high-quality training data. Even though there are a few public EIT datasets, they are insufficient to train DNNs (often with many parameters). In practice, the DNN is learned on synthetic data, simulated with phantoms via, e.g., FEM. The main advantage is that

once the neural network is trained, at the inference stage, the process requires only feeding through the trained neural network and thus can be done very efficiently. Generally, these approaches perform well when the test data is close to the distribution of the training data. Still, their performance may degrade significantly when the test data deviates from the setting of the training data [7]. This lack of robustness with respect to the out-of-distribution test data represents one outstanding challenge with all the above approaches.

Third, several unsupervised learning methods have been proposed for EIT reconstruction. Bar et al. [15] employ DNNs to approximate voltage functions $\{u_j\}_{j=1}^J$ and conductivity σ and then train them together to satisfy the strong PDE conditions and the boundary conditions, following the physics-informed neural networks (PINNs) [261]. Furthermore, data-driven energy-based models are imparted onto the approach to improve the convergence rate and robustness for EIT reconstruction [256]. Bao et al. [14] exploited the weak formulation of the EIT problem, using DNNs to parameterise the solutions and test functions and adopting a minimax formulation to alternatively update the DNN parameters (to find an approximate solution of the EIT problem). Liu et al. [218] applied the deep image prior (DIP) [306], a novel DNN-based approach to regularise inverse problems, to EIT, and optimised the conductivity function by back-propagation and the finite element solver. Generally, the methods in this group are robust with respect to the distributional shift of the test data. However, each new test data requires fresh training, and hence, they tend to be computationally more expensive.

In addition, several neural operators, e.g., [211, 223, 302], have been designed to approximate mostly forward operators. The recent survey [244] discusses various extensions of these neural operators for solving inverse problems by reversed input-output and studied Tikhonov regularisation with a trained forward model.

6.3.1 Deep D-bar

In practice, reconstructions obtained with the D-bar method suffer from a smoothing effect due to truncation in the scattering transform, which is necessary for finite and noisy data but leaves out all high-frequency information in the data. Thus, we cannot reconstruct sharp edges, and subsequent processing is beneficial. An early approach to overcome the smoothing is to use a nonlinear diffusion process to sharpen edges [129]. In recent years, deep learning has been highly successful for post-processing insufficient noise or artefact-corrupted reconstruction [169].

In the context of the deep D-bar method, we are given an initial analytic reconstruction operator $\mathcal{R}_{\text{d-bar}}$ that maps the measurements (i.e., the DtN map $\Lambda_{\sigma,D}$ for EIT) to an initial image, which suffers from various artefacts, primarily over-smoothing. Then a U-Net $\mathcal{G}_\theta$ [271] is trained to improve the reconstruction quality of the initial reconstructions, and we refer to the original publication [128] for details on the architecture. Thus, we could write this process as $\sigma \approx \mathcal{G}_\theta(\mathcal{R}_{\text{d-bar}}(\Lambda_{\sigma,D}))$, where the network $\mathcal{G}_\theta$ is trained by minimising the ℓ^2-loss of D-bar

reconstructions to ground-truth images. Specifically, given a collection of N paired training data $\{(\sigma_i^\dagger, \Lambda^\delta_{\sigma_i^\dagger,D})\}_{i=1}^N$ (i.e., ground-truth conductivity $\sigma_i^\dagger$ and the corresponding noisy measurement data $\Lambda^\delta_{\sigma_i^\dagger,D}$), we train a DNN $\mathcal{G}_\theta$ by minimising the following empirical loss

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^N \|\sigma_i^\dagger - \mathcal{G}_\theta(\mathcal{R}_{\text{d-bar}}(\Lambda^\delta_{\sigma_i^\dagger,D}))\|^2_{L^2(\Omega)},$$

This can be viewed as a specialised denoising scheme to remove the artefacts in the initial reconstruction $\mathcal{R}_{\text{d-bar}}(\Lambda^\delta_{\sigma_i^\dagger,D})$ by the D-bar reconstructor $\mathcal{R}_{\text{d-bar}}$. The loss $\mathcal{L}(\theta)$ is then minimised with respect to the DNN parameters θ, typically by the Adam algorithm [177], a very popular variant of stochastic gradient descent. Once a minimiser θ^* of the loss $\mathcal{L}(\theta)$ is found, given a new test measurement $\Lambda^\delta_{\sigma,D}$, we can obtain the reconstruction $\mathcal{G}_{\theta^*}(\mathcal{R}_{\text{d-bar}}(\Lambda^\delta_{\sigma,D}))$. Thus at the testing stage, the method requires only additional feeding of the initial reconstruction $\mathcal{R}_{\text{d-bar}}(\Lambda^\delta_{\sigma,D})$ through the network $\mathcal{G}_{\theta^*}$, which is computationally very efficient. This presents one distinct advantage of a supervisedly learned map.

Several extensions have been proposed. Firstly, the need to model boundary shapes in the training data can be eliminated by using the Beltrami approach [12] instead of the classic D-bar method. This allows for domain-independent training [127]. A similar motivation is given by replacing the classic U-net that operates on rectangular pixel domains with a graph convolutional version; this way learned filters are domain and shape-independent [136, 138]. Similarly, the reconstruction from Calderón's method [25, 281] can be post-processed using U-net, leading to the deep Calderón's method [43]. Distinctly, the deep Calderón's method is capable of directly recovering complex valued conductivity distributions. Finally, even the enclosure method can be improved by predicting the convex hull from values of the involved indicator function [282].

6.3.2 Deep direct sampling method

The deep sampling method (DDSM) [117] is based on the direct sampling method (DSM) due to Chow, Ito and Zou [59]. Using only one single Cauchy data pair on the boundary $\partial\Omega$, The DSM constructs a family of probing functions $\{\eta_{x,d_x}\}_{x\in\Omega,d_x\in\ ^n} \subset H^{2\gamma}(\partial\Omega)$ such that the index function defined by

$$\mathcal{I}(x,d_x) := \frac{\langle \eta_{x,d_x}, u-u_{\sigma_0}\rangle_{\gamma,\partial\Omega}}{\|u-u_{\sigma_0}\|_{L^2(\partial\Omega)}|\eta_{x,d_x}|_Y}, \quad x\in\Omega, \tag{6.7}$$

takes large values for points near the inclusions and relatively small values for points far away from the inclusions, where $|\cdot|_Y$ denotes the $H^{2\gamma}(\partial\Omega)$ seminorm in and the duality product $\langle f,g\rangle_{\gamma,\partial\Omega}$ is defined by

$$\langle f,g\rangle_{\gamma,\partial\Omega} = \int_{\partial\Omega}(-\Delta_{\partial\Omega})^\gamma f g \mathrm{d}s = \langle(-\Delta_{\partial\Omega})^\gamma f, g\rangle_{L^2(\partial\Omega)}, \tag{6.8}$$

where $-\Delta_{\partial\Omega}$ denotes the Laplace-Beltrami operator, and $(-\Delta_{\partial\Omega})^\gamma$ its fractional power via spectral calculus. Let the Cauchy difference function φ be defined by

$$-\Delta\varphi = 0 \quad \text{in} \quad \Omega, \quad \frac{\partial\varphi}{\partial n} = (-\Delta_{\partial\Omega})^\gamma(u_\sigma - u_{\sigma_0}) \quad \text{on} \quad \partial\Omega, \quad \int_{\partial\Omega} \varphi \mathrm{d}s = 0. \tag{6.9}$$

Then the index function $\mathcal{I}(x, d_x)$ can be equivalently rewritten as

$$\mathcal{I}(x, d_x) := \frac{d_x \cdot \nabla\varphi(x)}{\|u_\sigma - u_{\sigma_0}\|_{L^2(\partial\Omega)}|\eta_{x,d_x}|_Y}, \quad x \in \Omega, \tag{6.10}$$

Motivated by the relation between the index function $\mathcal{I}(x, d_x)$ and the Cauchy difference function φ and to fully make use of multiple pairs of measured Cauchy data, Guo and Jiang [117] proposed the DDSM, employing DNNs to learn the relationship between the Cauchy difference functions φ and the true inclusion distribution. That is, DSSM construct and train a DNN $\mathcal{G}_\theta$ such that

$$\sigma \approx \mathcal{G}_\theta(\varphi_1, \varphi_2, ..., \varphi_N), \tag{6.11}$$

where $\{\varphi_i\}_{i=1}^N$ correspond to N pairs of Cauchy data $\{g_\ell, \Lambda_{\sigma,N} g_\ell\}_{\ell=1}^N$. Guo and Jiang [117] employed a CNN-based U-Net network for DDSM, and later [116] designed a U-integral transformer architecture (including comparison with state-of-the-art DNN architectures, e.g., Fourier neural operator, and U-Net). In our numerical experiments, we choose the U-Net as the network architecture for DDSM as we observe that U-Net can achieve better results than the U-integral transformer for resolution 64×64. For higher resolution cases, the U-integral transformer seems to be a better choice due to its more robust ability to capture long-distance information. The following result [117, Theorem 4.1] provides some mathematical foundation of DDSM.

Theorem 6.3.1. *Let $\{g_\ell\}_{\ell=1}^\infty$ be a fixed orthonormal basis of $H^{-1/2}(\partial\Omega)$. Given an arbitrary σ such that $\sigma > \sigma_0$ or $\sigma < \sigma_0$, let $\{g_\ell, \Lambda_{\sigma,N} g_\ell\}_{\ell=1}^\infty$ be the Cauchy data pairs and let $\{\varphi_\ell\}_{\ell=1}^\infty$ be the corresponding Cauchy difference functions with $\gamma = \sigma_0$. Then the inclusion distribution σ can be purely determined from $\{\varphi_\ell\}_{\ell=1}^\infty$.*

The idea of DDSM was extended to diffusive optical tomography in [118]. Ning et al. [246] employ the index functions obtained from the DSM as the input of the DNN for solving inverse obstacle scattering problems.

6.3.3 CNN based on LeNet

Li et al. [209] proposed using CNN to directly learn the map from the measured data and the conductivity distribution. The employed network architecture is based on LeNet and refined by applying dropout layer and moving average. The CNN architecture used in the numerical experiments below is shown in Fig. 6.1. Since the number of injected currents and the discretisation size differ from that in [209], we modify the input size, network depth,

kernel size, etc. The input size is 32×64. The kernel size is 5×5 with zero-padding max pooling rather than average pooling is adopted to gain better performance. The sigmoid activation function used in LeNet causes a serious saturation phenomenon, which can lead to vanishing gradients. So, ReLU is chosen as the activation function below. A dropout layer is added to improve the generalisation ability of this model. One-half of the neurons before the first fully connected layers are randomly discarded from the network during the training process. It can reduce the complex co-adaptation among neurons so that the network can learn more robust features. In addition, a dropout layer has been proven to be very effective in training large datasets.

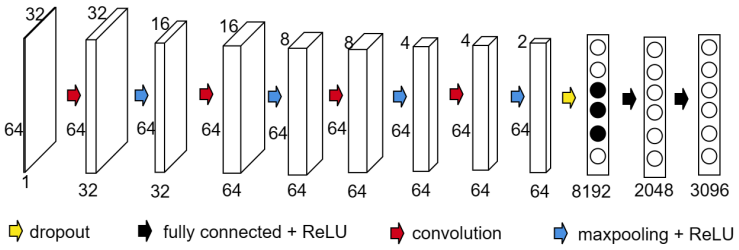


Figure 6.1: The architecture of CNN-based on LeNet.

6.3.4 FC-UNet

Chen et al. [52] proposed a novel deep learning architecture by adding a fully connected layer before the U-Net structure. The input of the network is given by the difference voltage $u_\sigma^\delta - u_{\sigma_0}$. Inspired by a linearized approximation of the EIT problem for a small perturbation of conductivity distribution $\sigma - \sigma_0$:

$$u_\sigma^\delta - u_{\sigma_0} \approx \mathbf{J}(\sigma - \sigma_0), \tag{6.12}$$

where $\mathbf{J}$ donates the sensitivity matrix, the method first generates an initial guess of the conductivity distribution σ from the linear fully connected(FC) layer followed by a ReLU layer and then feeds it to a denoising U-Net model to learn the nonlinear relationship further. Thus we could write this process as $\sigma \approx \mathcal{G}_\theta(u_\sigma^\delta - u_{\sigma_0}) = \mathcal{G}_{\theta_2}(\mathcal{G}_{\theta_1}(u_\sigma^\delta - u_{\sigma_0}))$ with $\mathcal{G}_{\theta_1}$ = FC+ReLU and $\mathcal{G}_{\theta_2}$ = U-Net. The authors also proposed an initialisation strategy to further help obtain the initial guess, i.e., the weights θ_1 of the fully connected layer are initialised with the least-squares solution using training data. The weights θ_2 for the U-Net are initialised randomly as usual. Then, all weights $\theta = \theta_1 \cup \theta_2$ are updated during the training process. According to the numerical results shown in [52], this special weight initialization strategy can reduce the training time and improve the reconstruction quality. With a trained network, different from the deep D-bar and DDSM methods, the

methods FC-UNet and CNN based on LeNet only involve a forward pass of the trained network for each testing example.

Based on our numerical experience, dropping the ReLU layer following the fully connected layer can provide better reconstruction results, at least for the examples in section 6.4. Thus, for the numerical experiments, we employ the FC-UNet network as shown in Fig. 6.2, in which only a linear fully connected layer is employed before the U-Net.

In addition, by employing the FC-UNet to extract structure distribution and a standard CNN to extract conductivity values, a structure-aware dual-branch network was designed in [51] to solve EIT problems.

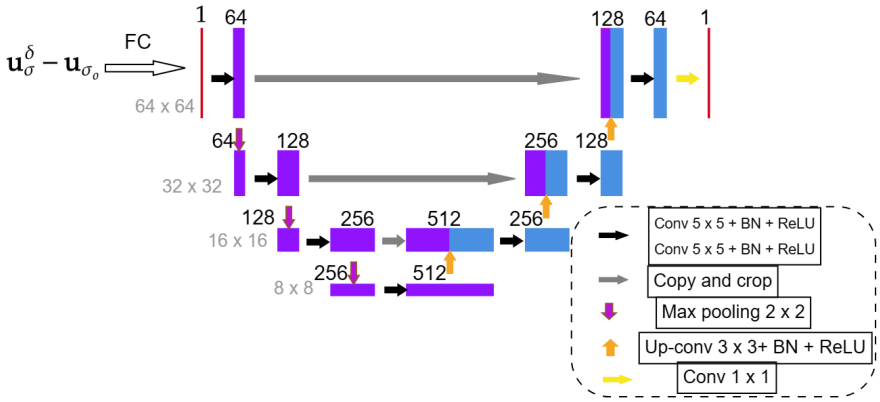


Figure 6.2: The architecture of FC-UNet.

6.4 Numerical experiments and results

The core of this work is the extensive numerical experiments. Now, we describe how to generate the dataset used in the experiments, highlighting its peculiarity and relevance in real-world scenarios, and also the performance metrics used for comparing different methods. Last, we present and discuss the experimental results.

6.4.1 Dataset generation and characteristics

Generating simulated data consists of three main parts, which we describe below. The codes for data generation are available at `https://github.com/derickngany u/EIT_dataset_generation`.

In the 2D setting, we generate N circular phantoms $\{P_i\_i = 1^N$, all restricted to the unit circle centred at the origin, i.e., $\Omega = \{(x, y) : x^2 + y^2 \leq 1\}$ in the Cartesian coordinates or $\{(r, \theta) : r \leq 1, \theta \in [0, 2\pi]\}$ in polar coordinates. The phantoms are generated randomly. Firstly, we decide on the maximum number $M \in$ of inclusions. Each phantom would then contain n inclusions, where $n \in \mathcal{U}\{1, \ldots, M\}$, the uniform distribution over the set $[1, \ldots, M]$. To mimic realistic scenarios in medical imaging, the inclusions are

elliptical and are sampled such that when $n > 1$, the inclusions do not overlap. Since the inclusions are elliptical, each inclusion, $(E_j)_{j=1}^n$ is characterised by a centre $C_j = (h_j, k_j)$, an angle of rotation α_j, a major and minor axis a_j and b_j respectively. The parametric equation of an ellipse E_j is thus given by

$$E_j = \left\{(x,y) : \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} h_j + a_j \cos\theta \cos\alpha_j - b_j \sin\theta \sin\alpha_j \\ k_j + a_j \cos\theta \sin\alpha_j + b_j \sin\theta \cos\alpha_j \end{pmatrix}, \theta \in [0, 2\pi]\right\}. \tag{6.13}$$

To mimic realistic scenarios in medical imaging, the inclusions are sampled to avoid contact with the boundary $\partial\Omega$ of the domain Ω. For an inclusion E_j, we have $x^2 + y^2 < 0.9$ for any $(x, y) \in E_j$. In this way, all phantoms have inclusions contained within Ω. We illustrate this in Algorithm 10.

Each phantom $P_i, i \in \{1, 2, \ldots, N\}$, has $(E_j)_{j=1}^n$ inclusions, with $n \in \mathcal{U}\{1, \ldots, M\}$. For each E_j, we assign a conductivity $\sigma_j^i \in \Sigma_j := \mathcal{U}(0.2, 0.8) \cup \mathcal{U}(1.2, 2.0)$. The background conductivity is set to 1. In this way, given a point $(x, y) \in P_i$ in the domain/phantom, the conductivity $\sigma_i(x, y)$ at that point is therefore given by

$$\sigma_i(x,y) = \begin{cases} \sigma_j^i \in \Sigma_j, & \text{if } (x,y) \in E_j,\ j = 1, \ldots, n \\ 1, & \text{otherwise.} \end{cases} \tag{6.14}$$

Fig. 6.3b shows an example of a phantom generated in this way.

Next, for any simulated σ, we solve the forward problem (6.1) using the Galerkin finite element method (FEM) [99, 199], for the injected currents g_1 and g_2 in (6.15) around the boundary $\partial\Omega$. The points $(x, y) \in P_i$ are thus nodes in the finite element mesh shown in Fig. 6.3a

$$g_1 = \pi^{-1/2} \sin(n\theta) \quad \text{and} \quad g_2 \quad = \pi^{-1/2} \cos(n\theta), \quad n = 1, 2, \ldots, 16 \tag{6.15}$$

We use the MATLAB PDE toolbox in the numerical experiment to solve the forward problem.

In real-life situations, the conductivities of the inclusions are rarely constant. Indeed, usually, there are textures on internal organs in medical applications. Motivated by this, we take a step further in generating phantoms, with inclusions having variable conductivities. This introduces a novel challenge to the EIT problem, and we seek to study its impact on different reconstruction algorithms. The procedure to generate simulated data remains unchanged. However, σ_j^i in equation (6.14) becomes

$$\sigma_j^i = s \circ f \circ R_{\alpha_j, C_j},$$

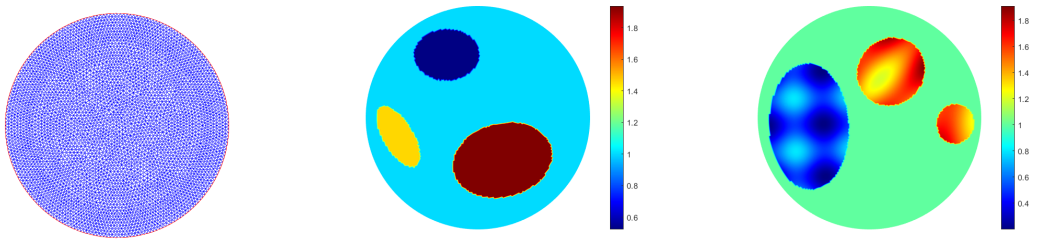
where $f : \ ^2 \ni (x, y) \mapsto \frac{1}{2}(\sin k_x x + \sin k_y y) \in [-1, 1]$, R_{α_j, C_j} is the rotation of centre $C_j = (h_j, k_j)$ and angle α_j, with respect to the centre and angle of the ellipse E_j respectively; and s applies a scaling so that the resulting σ_j^i is either within the range $[0.2, 0.8]$ or $[1.2, 2.0]$. Fig. 6.3c shows an example phantom.

Algorithm 10: Procedure for generating phantoms

Input:
- nodes $(x_\ell, y_\ell), \ell \in \{1, 2, \ldots, L\}$ from FEM mesh
- $N \in$, number of phantoms
- $M \in$, maximum number of inclusions

Result:
- Phantoms P_i, with conductivity $\sigma_i, i \in \{1, 2, \ldots, n\}$

select $n \in \mathcal{U}\{1, M\}$;
for $i \leftarrow 1, \ldots, N$ **do**
  for $j \leftarrow 1, \ldots, n$ **do**
    `/* Sample inclusions and conductivity */`
    Sample E_j, non-overlapping ellipses based on (6.13), within the circle of radius 0.9;
    Sample $\sigma_j^i \in \mathcal{U}(0.2, 0.8) \cup \mathcal{U}(1.2, 2.0)$;
  end
  for $\ell \leftarrow 1, \ldots, L$ **do**
    `/* Evaluate conductivity on mesh nodes */`
    Evaluate $\sigma_i(x_\ell, y_\ell)$ based on (6.14) ;
  end
end



(a) Forward solver FEM mesh. **(b)** Constant $\sigma_j^i\big|_{j=1}^3$ inclusions. **(c)** Textured $\sigma_j^i\big|_{j=1}^3$ inclusions.

Figure 6.3: Illustrating of Phantom characteristics used in simulated data.

We also study the performance of the methods in noisy scenarios, i.e. reconstructing the conductivity from noisy measurements. The resulting solution to the forward problem u, on the boundary $\partial\Omega$, is then perturbed with normally distributed random noise of different levels δ:

$$u^\delta(x) = u(x) + \delta \cdot |u(x)| \cdot \xi(x), \quad x \in \partial\Omega,$$

where $\xi(x)$ follows the standard normal distribution $\mathcal{N}(0, 1)$.

For the deep learning methods, we employ 20,000 training data and 100 validation data without noises added. Then we compare the results for 100 testing data with different noise levels.

We employ several performance metrics commonly used in the literature to compare different reconstruction methods comprehensively. Table 6.1 outlines these metrics with their mathematical expressions and specifications. In Table 6.1, $\boldsymbol{\sigma}$ denotes the ground truth with mean $\mu_{\boldsymbol{\sigma}}$ and variance $s^2_{\boldsymbol{\sigma}}$, while $\hat{\boldsymbol{\sigma}}$ the predicted conductivity with mean $\mu_{\hat{\boldsymbol{\sigma}}}$ and variance $s^2_{\hat{\boldsymbol{\sigma}}}$. $\hat{\sigma}_i$ is the i-th element of $\hat{\boldsymbol{\sigma}}$ while σ_i is the i-th element of $\boldsymbol{\sigma}$. N is the total number of pixels, so that $\boldsymbol{\sigma} = (\sigma_i)_{i=1}^N$ and $\hat{\boldsymbol{\sigma}} = (\hat{\sigma}_i)_{i=1}^N$.

Error Metric	Mathematical Expression	Highlights
Relative Image Error (RIE)	$\frac{\|\hat{\boldsymbol{\sigma}} - \boldsymbol{\sigma}\|}{\|\boldsymbol{\sigma}\|} = \frac{\sum_{i=1}^{N} \|\hat{\sigma}_i - \sigma\|}{\sum_{i=1}^{N} \|\sigma_i\|}$	Evaluates the relative error between the true value and prediction [52].
Image Correlation Coefficient (ICC)	$\frac{\sum_{i=1}^{N} (\hat{\sigma}_i - \mu_{\hat{\boldsymbol{\sigma}}})(\sigma_i - \mu_{\boldsymbol{\sigma}})}{\sqrt{\sum_{i=1}^{N} (\hat{\sigma}_i - \mu_{\hat{\boldsymbol{\sigma}}})^2} \sqrt{\sum_{i=1}^{N} (\sigma_i - \mu_{\boldsymbol{\sigma}})^2}}$	Measures the similarity between the true value and prediction[52, 323].
Dice Coefficient (DC)	$\frac{2\|X \cap Y\|}{\|X\| + \|Y\|}$	Tests the accuracy of the results. It provides a ratio of pixels correctly predicted to the total number of pixels—the closer to 1, the better [116]. For our experiments, we round the pixel values to 2 decimal places before evaluation.
Relative L^2 Error (RLE)	$\frac{\|\hat{\boldsymbol{\sigma}} - \boldsymbol{\sigma}\|_2}{\|\boldsymbol{\sigma}\|_2} = \frac{\left(\sum_{i=1}^{N} \|\hat{\sigma}_i - \sigma\|^2\right)^{1/2}}{\left(\sum_{i=1}^{N} \|\sigma_i\|^2\right)^{1/2}}$	Measures the relative difference between the truth and the prediction. The closer to 0, the better. [116, 323]
Root Mean Squared Error (RMSE)	$\sqrt{\frac{1}{N} \sum_{i=1}^{N} (\sigma_i - \hat{\sigma}_i)^2}$	Evaluates the average magnitude of the differences between the truth and the prediction. [323]
Mean Absolute Error (MAE)	$\frac{1}{N} \sum_{i=1}^{N} \|\sigma_i - \hat{\sigma}_i\|$	Evaluates the average magnitude of the differences between the truth and the prediction[323]

Table 6.1: Description of various performance metrics.

6.4.2 Results and discussions

Tables 6.2 and 6.3 present quantitative values for the performance metrics of various EIT reconstruction methods, in the presence of different noise levels, $\delta = 0\%$, $\delta = 1\%$ and, $\delta = 5\%$. The considered performance metrics are described in Table 6.1. Understanding the results requires considering the behaviour of these metrics: For RIE, RMSE, MAE, and RLE, lower values indicate better performance and the objective is to minimise them; for DC and ICC, values closer to 1 indicate better performance, and the goal is to maximise them. Below, we examine the results in each table more closely.

Piece-wise constant conductivities

In the noiseless scenario as depicted in Table 6.2a, FC-UNet shows the best performance across all metrics, with notably low RIE, RMSE, MAE, and RLE. It also achieves a high DC and ICC, indicating robustness and accuracy in image reconstruction. The DDSM also performs well, particularly regarding RIE, RMSE, MAE, and RLE. The Deep D-bar method exhibits competitive results, although slightly inferior to FC-UNet. Both Sparsity and D-bar methods show weaker performance compared to the deep learning-

	RIE	ICC	DC	RMSE	MAE	RLE
Sparsity	0.03844	0.02159	0.79134	0.09989	0.10360	0.03904
D-bar	0.09784	0.01486	0.08581	0.15515	0.16123	0.09928
Deep D-bar	0.03677	0.02627	0.45121	0.09957	0.10269	0.03721
DDSM	0.03450	0.02690	0.48590	0.08793	0.09075	0.03494
FC-UNet	0.01863	0.02954	0.76004	0.06405	0.06615	0.01890
CNN LeNet	0.04951	0.02509	0.18129	0.08579	0.08856	0.05011

(a) $\delta = 0\%$

	RIE	ICC	DC	RMSE	MAE	RLE
Sparsity	0.03835	0.02162	0.79263	0.09982	0.10353	0.03894
D-bar	0.08756	0.01429	0.08125	0.14254	0.14798	0.08889
Deep D-bar	0.02738	0.02762	0.75264	0.08477	0.08751	0.02774
DDSM	0.03581	0.02705	0.46511	0.09047	0.09342	0.03630
FC-UNet	0.02159	0.02929	0.72974	0.07170	0.07409	0.02194
CNN LeNet	0.05905	0.02499	0.15198	0.09884	0.10215	0.05988

(b) $\delta = 1\%$

	RIE	ICC	DC	RMSE	MAE	RLE
Sparsity	0.03952	0.02159	0.78729	0.10272	0.10658	0.04015
D-bar	0.08585	0.01349	0.06841	0.13870	0.14377	0.08713
Deep D-bar	0.05563	0.02272	0.51711	0.13523	0.13954	0.05648
DDSM	0.04833	0.02412	0.38292	0.11310	0.11704	0.04917
FC-UNet	0.04332	0.02672	0.28293	0.11519	0.11923	0.04415
CNN LeNet	0.13901	0.02312	0.04568	0.21138	0.21876	0.14155

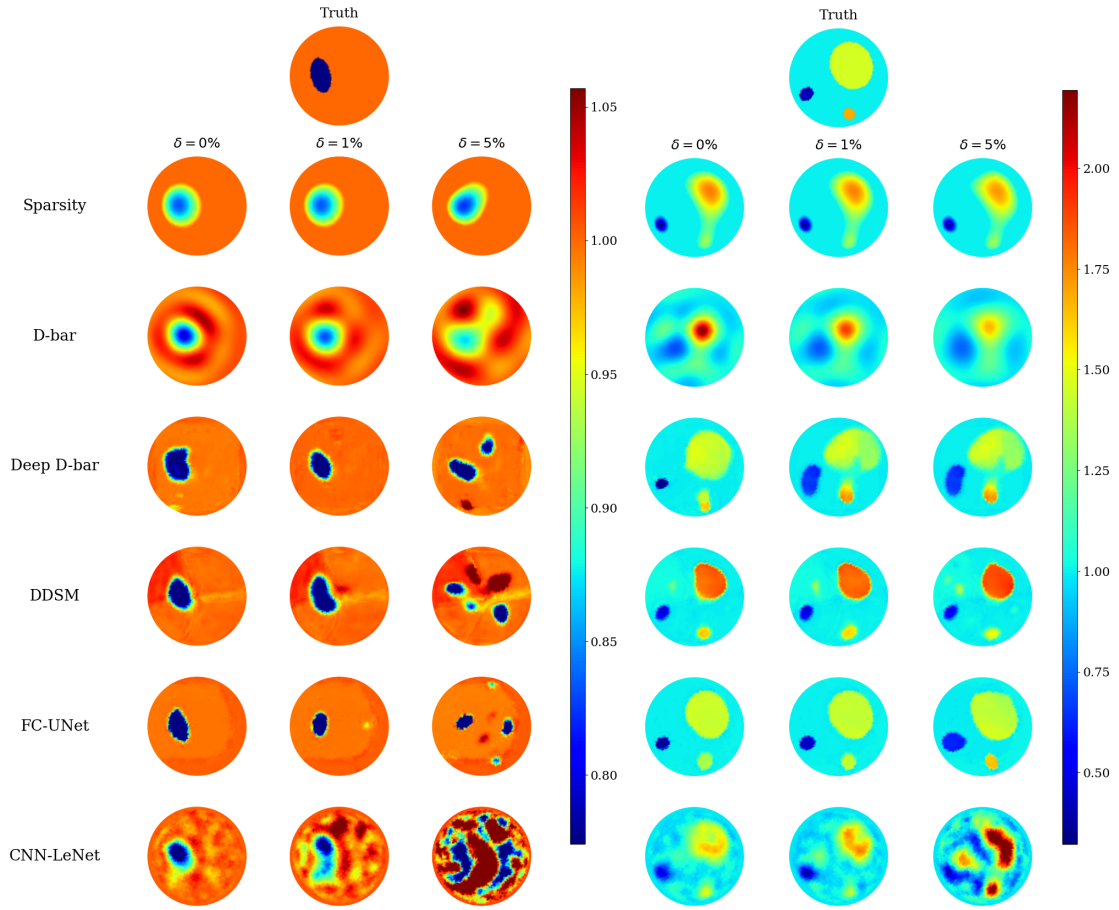
(c) $\delta = 5\%$

Table 6.2: The performance of various methods trained and tested with piece-wise constant data at different noise levels. The neural networks used were trained with noiseless measurements for the deep learning-based methods.

based methods. The CNN-LeNet method generally has the worst performance metrics, indicating less accurate image reconstruction.

Under increased noise of $\delta = 1\%$, the relative performance of the methods remains consistent, with FC-UNet still demonstrating strong performance. Also, the Deep D-bar performs exceptionally well in this case, particularly in terms of RIE, RMSE, MAE, and RLE. The DDSM also exhibits robust performance under this noise level, while the CNN LeNet method continues to have the highest values for most metrics, indicating challenges in handling noise. In contrast, the analytic-based methods of Sparsity and D-bar show particular robustness to the added noise, evidenced by the unnoticeable change in the performance metrics.

At a higher noise level $\delta = 5\%$, the inverse problem becomes more challenging due to the severe ill-posed nature; and in the learned context, since the neural networks are trained on noiseless data, which differ markedly from the noisy data, the setting may be viewed as an out-of-distribution robustness test. Here, the sparsity method comes on top across most metrics, having almost maintained constant performance. However, the FC-UNet continues to maintain the best performance in terms of ICC, emphasising its robustness in noisy conditions. Deep D-bar and DDSM display competitive results, indicating resilience to increased noise. The D-bar methods exhibit slightly weaker performance, especially in



(a) Sample 1. **(b)** Sample 2.

Figure 6.4: Effects of noise on two piecewise constant samples by various reconstruction methods.

terms of RIE, RMSE, and MAE. In contrast, the CNN LeNet method continues to have the highest values for most metrics, suggesting difficulty in coping with substantial noise.

Overall, these results illustrate the varying performance of different EIT methods under different noise levels. The deep learning-based methods, particularly FC-UNet, exhibit good performance across low noise levels. In contrast, the sparsity method shows proof of consistent robustness across higher noise levels, indicating their effectiveness in reconstructing EIT images, even in the presence of noise. Visual results across all the noise levels are shown for two test samples in Figure 6.4.

Textured inclusions scenario

In the noiseless scenario depicted in Table 6.3a, The best-performing method based on RIE, ICC, RMSE, MAE, and RLE is FC-UNet, with the best values across these metrics. The sparsity method and DDSM also perform well, being the first runners-up in these metrics, particularly for DC; the sparsity method achieves the highest values, indicating good performance, with DDSM as the first runner-up. The worst-performing method

	RIE	ICC	DC	RMSE	MAE	RLE
Sparsity	0.03869	0.01968	0.79695	0.10473	0.10864	0.03949
D-bar	0.08856	0.01473	0.09956	0.14597	0.15257	0.09063
Deep D-bar	0.03677	0.02627	0.45121	0.09957	0.10269	0.03721
DDSM	0.03559	0.02360	0.42812	0.08930	0.09282	0.03641
FC-UNet	0.02781	0.02639	0.45464	0.07379	0.07679	0.02850
CNN LeNet	0.04930	0.02308	0.18749	0.09010	0.09357	0.05034

(a) $\delta = 0\%$

	RIE	ICC	DC	RMSE	MAE	RLE
Sparsity	0.03871	0.01961	0.79497	0.10455	0.10846	0.03951
D-bar	0.07982	0.01381	0.09294	0.13634	0.14231	0.08168
Deep D-bar	0.02738	0.02762	0.75264	0.08477	0.08751	0.02774
DDSM	0.03663	0.02325	0.43803	0.09261	0.09626	0.03750
FC-UNet	0.02980	0.02592	0.42355	0.07957	0.08280	0.03055
CNN LeNet	0.06301	0.02253	0.12531	0.10709	0.11112	0.06426

(b) $\delta = 1\%$

	RIE	ICC	DC	RMSE	MAE	RLE
Sparsity	0.03975	0.01961	0.79157	0.10766	0.11177	0.04061
D-bar	0.08015	0.01265	0.07503	0.13590	0.14161	0.08193
Deep D-bar	0.05563	0.02272	0.51711	0.13523	0.13954	0.05648
DDSM	0.04775	0.02091	0.38445	0.11451	0.11883	0.04882
FC-UNet	0.05195	0.02269	0.28514	0.12528	0.12999	0.05312
CNN LeNet	0.17301	0.01907	0.03654	0.25557	0.26481	0.17637

(c) $\delta = 5\%$

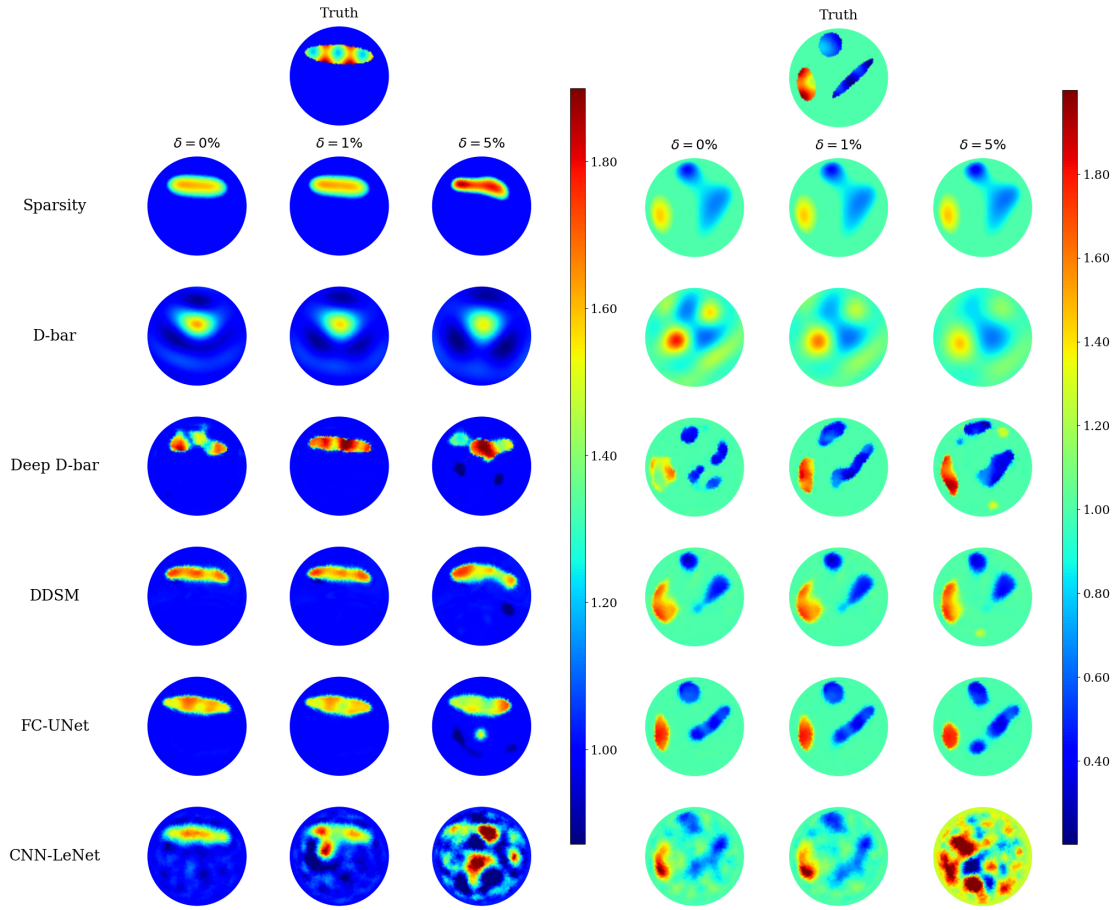
Table 6.3: The performance of various methods trained and tested with textured data at different noise levels. The neural networks used were trained with noiseless measurements for the deep learning-based methods.

across all metrics in this scenario is "D-bar." With a bit of noise of 1% added, the Deep D-bar surprisingly stands out as the best-performing for most of the considered metrics. The FC-UNet closely follows it. The sparsity-based method continues to lead in DC. Like the noiseless scenario, D-bar remains one of the less effective methods across all metrics. This is depicted in Table 6.3b.

For higher noise levels in Table 6.3c, the sparsity-based methods once again excel in all metrics but for the ICC, making it the best-performing method. The DDSM and FC-UNet closely follow in most of these metrics, while the Deep D-bar continues to perform best in ICC. The CNN LeNet consistently performs the poorest across all metrics and noise levels, especially in this high-noise scenario.

In summary, the best-performing method varies depending on the specific performance metric and noise level. Sparsity consistently demonstrates robust performance in both noiseless and noisy scenarios, while the D-bar is generally less effective. However, in terms of computational expense, the sparsity method is more expensive. The Deep D-bar, FC-UNet, and DDSM often serve as strong contenders, shifting their rankings across noise scenarios and metrics. Meanwhile, CNN LeNet consistently performs the poorest, particularly in high-noise scenarios ($\sigma = 5\%$). Figure 6.5 depicts this for two test examples.

Furthermore, for both piecewise constant and textured phantoms, the sparsity-based method consistently performed well for noisy scenarios. This consistently good performance



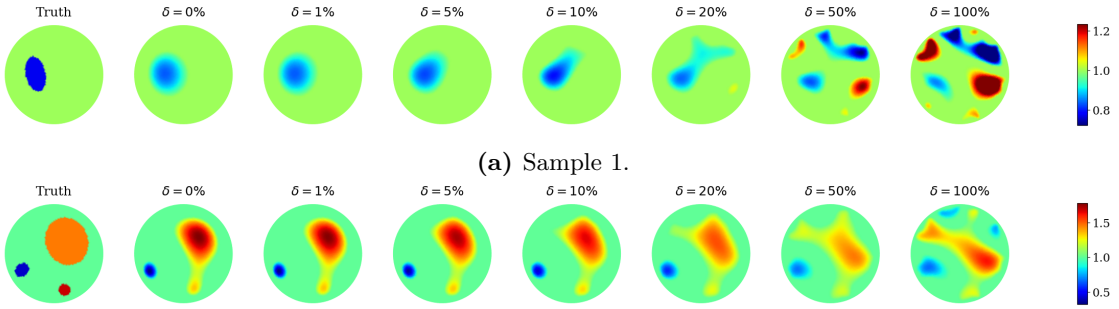
(a) Sample 1. **(b)** Sample 2.

Figure 6.5: Effects of noise on two textured samples by various reconstruction methods.

of the sparsity concept in detecting and locating inclusions even for higher noise levels is most remarkable. The error metrics are almost constant over noise levels up to 5%. Hence, as a side result, we did check the limits of the sparsity concept for very high noise levels, which not surprisingly showed a sharp decrease in the reconstruction accuracy for very high noise levels. We show this in Figure 6.6, once again for the two piecewise constant samples initially displayed in Figure 6.4. The respective performances, all metrics considered, for these two samples are equally shown in Figure 6.7 (ICC is not plotted for the sake of visibility since its values are smallest). Figures 6.8 and 6.9 show the corresponding plots for the textured samples initially displayed in Figure 6.5.

6.5 Conclusion and future directions

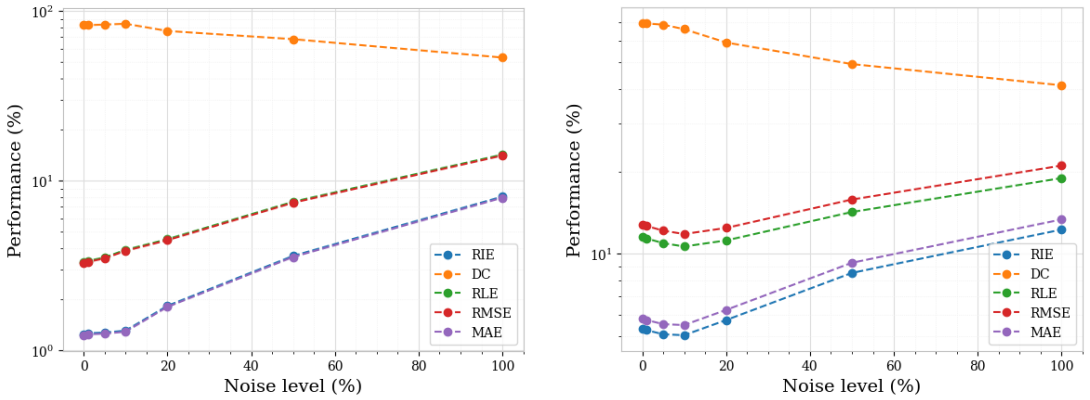
In summary, this review has comprehensively examined numerical methods for addressing the EIT inverse problem. EIT, a versatile imaging technique with applications in various fields, presents a highly challenging task of reconstructing internal conductivity distributions



(a) Sample 1.

(b) Sample 2.

Figure 6.6: Effects of additional noise on two piecewise samples by the sparsity method.



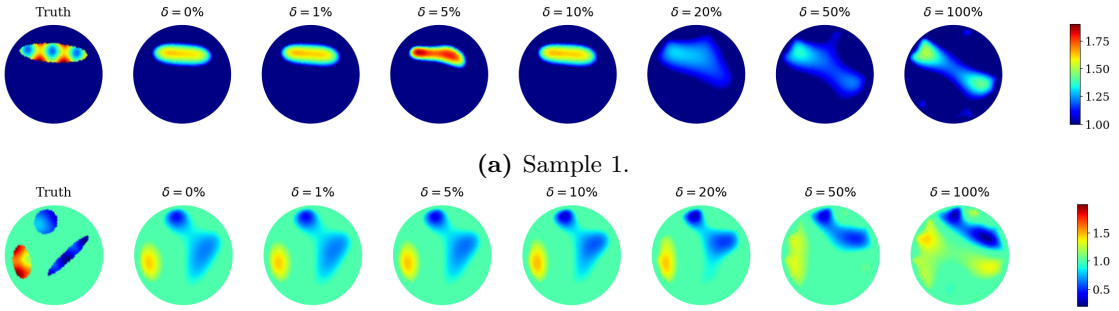
(a) Sample 1.

(b) Sample 2.

Figure 6.7: Performance variation with noise for two piecewise samples by the sparsity method.

from boundary measurements. We explored the interplay between modern deep learning-based approaches and traditional analytic methods for solving the EIT inverse problem. Four advanced deep learning algorithms were rigorously assessed, including the deep D-bar method, deep direct sampling method, fully connected U-net, and convolutional neural networks. Additionally, two analytic-based methods, incorporating mathematical formulations and regularisation techniques, were examined regarding their efficacy and limitations. Our evaluation involved a comprehensive array of numerical experiments encompassing diverse scenarios that mimic real-world complexities. Multiple performance metrics were employed to shed insights into the methods' capabilities to capture essential features and delineate complex conductivity patterns.

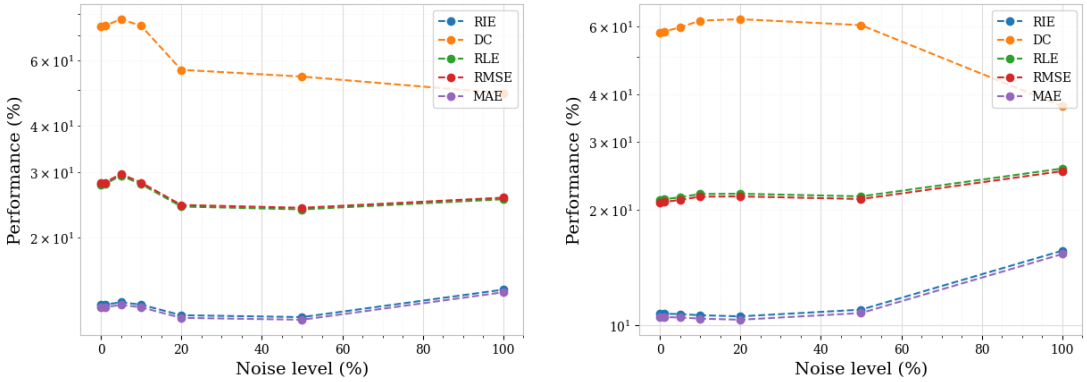
The first evaluation was based on piecewise constant conductivities. The clear winners of this series of tests are the analytic sparsity-based reconstruction and the learned FC-UNet. Both perform best, with slight variations depending on the noise level. This is not surprising for learned methods, which adapt well to this particular set of test data. However, the excellent performance of sparsity methods, which can identify and locate piecewise constant inclusions correctly, is most remarkable.



(a) Sample 1.

(b) Sample 2.

Figure 6.8: Effects of additional noise on two textured samples by the sparsity method.



(a) Sample 1. **(b)** Sample 2.

Figure 6.9: Performance variation with noise for two textured samples by the sparsity method.

A noteworthy aspect of this study was the introduction of variable conductivity scenarios, mimicking textured inclusions and departing from uniform conductivity assumptions. This enabled us to assess how each method responds to varying conductivity, shedding light on their robustness and adaptability. Here, the D-bar with learned post-processing achieves competitive results. The winning algorithm alternates between sparsity, Deep D-bar and FC-UNet. The good performance of the sparsity concepts is somewhat surprising for these textured test samples. However, none of the proposed methods was able to reconstruct the textures reliably for higher noise levels. That is, the quality of the reconstruction was mainly measured in terms of how well the inclusions were located - which gives a particular advantage to sparsity concepts.

These results naturally raise questions about the numerical results presented in several existing EIT studies, where learned methods were only compared with sub-optimal analytic methods. Our findings clearly indicate that at least within the restricted scope of the present study, optimised analytical methods can reach a comparable or even superior accuracy. Of course, one should note that after training, learned methods are much more efficient and provide a preferred option for real-time imaging.

In conclusion, this review contributes to a deeper understanding of the available solutions

for the EIT inverse problem, highlighting the role of deep learning and analytic-based methods in advancing the field.

You can't connect the dots looking forward; you can only connect them looking backwards. So you have to trust that the dots will somehow connect in your future.

— Steve Jobs

7 Summary and Conclusions

The convergence of deep learning and mathematics has undeniably reshaped both disciplines, ushering in profound advancements. This synergy has led to a two-fold exploration: a deep dive into the mathematical foundations of deep learning for heightened robustness, and a strategic application of deep learning to tackle complex mathematical problems, forging a path for scientific machine learning. This interdisciplinary collaboration has catalysed a paradigm shift in scientific computing, with a specific focus on partial differential equations (PDEs). New neural network architectures, purpose-built to address distinct classes of PDEs, leverage the intrinsic properties of these equations. These pioneering developments have left an indelible mark on mathematical modelling, particularly in the context of parametric PDEs, which play a pivotal role in representing natural and physical processes in science and engineering.

This thesis has been focused on the examination of these specialized neural network techniques, extending their utility for parametric studies and the resolution of related inverse problems. Firstly, somewhat basic but interesting PDEs like the Poisson and Darcy flow equations were studied for both the forward and inverse problems. Neural networks were seen to be quite applicable in the inverse setting, where multiple solutions of the PDE are usually needed, as they offer a better cost than classical methods for solving PDEs, which on the other hand offer a better accuracy. Building on this foundational work, our research ventured into more intricate PDEs encountered in the scientific and engineering domains, including but not limited to the Navier-Stokes equation, Helmholtz equation, advection equation, and Solid mechanics equation. These methodologies underwent rigorous scrutiny, pitting neural operator-based techniques against classical finite element solvers and Tikhonov functional-based approaches. Extensive numerical experiments, conducted across varying noise levels, have illuminated the trade-offs and applicability of these diverse methods for the multitude of PDE-based challenges they address.

The relevance of this method was equally demonstrated in industrial applications, with a prominent example being the realm of continuum mechanics in the automotive industry. This application situates itself in vehicle development, where stress and resulting strain (deformation) are of importance

A distinct facet of this thesis explored Electrical Impedance Tomography (EIT), an influential imaging technique with diverse applications. The primary focus was the resolution of its intricate PDE-based inverse problem. Our inquiry encompassed a comprehensive examination of deep learning-based and analytic-based strategies, shedding light on their respective strengths and limitations. We introduced novel variable conductivity scenarios, mirroring the complexities of real-world applications, and facilitating a nuanced assessment of the methods' robustness and adaptability.

In summation, this thesis stands as a testament to the profound transformations that have unfolded at the intersection of deep learning and mathematics, specifically in the fields of PDEs and Inverse problems. The deep insights and innovative methodologies uncovered here will undoubtedly continue to fuel the evolution of scientific computing, addressing intricate challenges across diverse domains.

References

[1] C. Aarset, M. Holler, and T. T. N. Nguyen. "Learning-informed parameter identification in nonlinear time-dependent PDEs". In: *arXiv preprint arXiv:2202.10915* (2022).

[2] J. Adler and O. Öktem. "Learned primal-dual reconstruction". In: *IEEE transactions on medical imaging* 37.6 (2018), pp. 1322–1332.

[3] P. del Aguila Pla. "Inverse problems in signal processing: Functional optimization, parameter estimation and machine learning". PhD thesis. KTH Royal Institute of Technology, 2019.

[4] G. Alessandrini. "Stable determination of conductivity by boundary measurements". In: *Applicable Analysis* 27.1-3 (1988), pp. 153–172.

[5] H. Ammari, R. Griesmaier, and M. Hanke. "Identification of small inhomogeneities: asymptotic factorization". In: *Math. Comp.* 76.259 (2007), pp. 1425–1448. URL: https://doi.org/10.1090/S0025-5718-07-01946-1.

[6] H. Ammari, E. Iakovleva, and D. Lesselier. "A MUSIC algorithm for locating small inclusions buried in a half-space from the scattering amplitude at a fixed frequency". In: *Multiscale Model. Simul.* 3.3 (2005), pp. 597–628. URL: https://doi.org/10.1137/040610854.

[7] V. Antun, F. Renna, C. Poon, and A. C. Hansen. "On instabilities of deep learning in image reconstruction and the potential costs of AI". In: *Proc. Nat. Acad. Sci.* 117.48 (2020), pp. 30088–30095.

[8] L. Ardizzone, J. Kruse, S. Wirkert, D. Rahner, E. W. Pellegrini, R. S. Klessen, L. Maier-Hein, C. Rother, and U. Köthe. "Analyzing inverse problems with invertible neural networks". In: *arXiv preprint arXiv:1808.04730* (2018).

[9] S. Arridge and A. Hauptmann. "Networks for nonlinear diffusion problems in imaging". In: *Journal of Mathematical Imaging and Vision* 62.3 (2020), pp. 471–487.

[10] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb. "Solving inverse problems using data-driven models". In: *Acta Numerica* 28 (2019), pp. 1–174.

[11] K. Astala, D. Faraco, and L. Székelyhidi Jr. "Convex integration and the L^p theory of elliptic equations". In: *Ann. Sc. Norm. Super. Pisa Cl. Sci. (5)* 7.1 (2008), pp. 1–50.

[12] K. Astala and L. Päivärinta. "Calderón's inverse conductivity problem in the plane". In: *Ann. of Math. (2)* 163.1 (2006), pp. 265–299. URL: https://doi.org/10.4007/annals.2006.163.265.

[13] D. O. Baguer, J. Leuschner, and M. Schmidt. "Computed tomography reconstruction using deep image prior and learned reconstruction methods". In: *Inverse Problems* 36.9 (2020), p. 094004.

[14] G. Bao, X. Ye, Y. Zang, and H. Zhou. "Numerical solution of inverse problems by weak adversarial networks". In: *Inverse Problems* 36.11 (2020), p. 115003.

[15] L. Bar and N. Sochen. "Strong solutions for PDE-based tomography by unsupervised learning". In: *SIAM J. Imag. Sci.* 14.1 (2021), pp. 128–155.

[16] A. R. Barron. "Approximation and estimation bounds for artificial neural networks". In: *Machine learning* 14 (1994), pp. 115–133.

[17] J. Barzilai and J. M. Borwein. “Two-point step size gradient methods”. In: *IMA J. Numer. Anal.* 8.1 (1988), pp. 141–148. URL: https://doi.org/10.1093/imanum/8.1.141.

[18] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. “Automatic differentiation in machine learning: a survey”. In: *Journal of Marchine Learning Research* 18 (2018), pp. 1–43.

[19] C. Beck, S. Becker, P. Cheridito, A. Jentzen, and A. Neufeld. “Deep splitting method for parabolic PDEs”. In: *SIAM Journal on Scientific Computing* 43.5 (2021), A3135–A3154.

[20] R. Bellman. *Dynamic Programming*. 1st ed. Princeton, NJ, USA: Princeton University Press, 1957.

[21] P. Beneventano, P. Cheridito, A. Jentzen, and P. von Wurstemberger. “High-dimensional approximation spaces of artificial neural networks and applications to partial differential equations”. In: *arXiv preprint arXiv:2012.04326* (2020).

[22] P. Benner, W. Schilders, S. Grivet-Talocia, A. Quarteroni, G. Rozza, and L. Miguel Silveira. *Model Order Reduction: Volume 2: Snapshot-Based Methods and Algorithms*. De Gruyter, 2020.

[23] J. Berner, P. Grohs, and A. Jentzen. “Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of Black–Scholes partial differential equations”. In: *SIAM Journal on Mathematics of Data Science* 2.3 (2020), pp. 631–657.

[24] K. Bhattacharya, B. Hosseini, N. B. Kovachki, and A. M. Stuart. “Model reduction and neural networks for parametric PDEs”. In: *arXiv preprint arXiv:2005.03180* (2020).

[25] J. Bikowski and J. L. Mueller. “2D EIT reconstructions using Calderón’s method”. In: *Inverse Probl. Imaging* 2.1 (2008), pp. 43–61. URL: https://doi.org/10.3934/ipi.2008.2.43.

[26] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[27] C. Blick, W. Freeden, M. Z. Nashed, H. Nutz, and M. Schreiner. *Inverse Magnetometry: Mollifier Magnetization Distribution from Geomagnetic Field Data*. Lecture Notes in Geosystems Mathematics and Computing. Cham: Birkhäuser, 2021.

[28] J. Bohr. “A Bernstein–von-Mises theorem for the Calderón problem with piecewise constant conductivities”. In: *Inverse Problems* 39.1 (2023), Paper No. 015002, 18.

[29] H. Bolcskei, P. Grohs, G. Kutyniok, and P. Petersen. “Optimal approximation with sparsely connected deep neural networks”. In: *SIAM Journal on Mathematics of Data Science* 1.1 (2019), pp. 8–45.

[30] T. Bonesky, K. Bredies, D. A. Lorenz, and P. Maass. “A generalized conditional gradient method for nonlinear operator equations with sparsity constraints”. In: *Inverse Problems* 23.5 (2007), pp. 2041–2058. URL: https://doi.org/10.1088/0266-5611/23/5/014.

[31] L. Borcea. “Electrical impedance tomography”. In: *Inverse Problems* 18.6 (2002), R99–R136. URL: https://doi.org/10.1088/0266-5611/18/6/201.

[32] L. Borcea, G. A. Gray, and Y. Zhang. “Variationally constrained numerical solution of electrical impedance tomography”. In: *Inverse Problems* 19.5 (2003), pp. 1159–1184.

[33] A. Borsic, B. M. Graham, A. Adler, and W. R. B. Lionheart. “In vivo impedance imaging With total variation regularization”. In: *IEEE Trans. Med. Imag.* 29.1 (2010), pp. 44–54.

[34] L. Böselt, M. Thürlemann, and S. Riniker. “Machine learning in QM/MM molecular dynamics simulations of condensed-phase systems”. In: *Journal of Chemical Theory and Computation* 17.5 (2021), pp. 2641–2658.

[35] K. Bredies, D. A. Lorenz, and P. Maass. “A generalized conditional gradient method and its connection to an iterative shrinkage method”. In: *Comput. Optim. Appl.* 42.2 (2009), pp. 173–193. URL: https://doi.org/10.1007/s10589-007-9083-3.

[36] S. C. Brenner and L. R. Scott. *The Mathematical Theory of Finite Element Methods*. Vol. 15. Texts in Applied Mathematics. Springer, 2008.

[37] M. Brühl and M. Hanke. "Numerical implementation of two noniterative methods for locating inclusions by impedance tomography". In: *Inverse Problems* 16.4 (2000), pp. 1029–1042. URL: https://doi.org/10.1088/0266-5611/16/4/310.

[38] M. Brühl, M. Hanke, and M. S. Vogelius. "A direct impedance tomography algorithm for locating small inhomogeneities". In: *Numer. Math.* 93.4 (2003), pp. 635–654. URL: https://doi.org/10.1007/s002110200409.

[39] J. Bu and A. Karpatne. "Quadratic residual networks: A new class of neural networks for solving forward and inverse problems in physics involving PDEs". In: *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM. 2021, pp. 675–683.

[40] M. Burger and A. Neubauer. "Analysis of Tikhonov regularization for function approximation by neural networks". In: *Neural Networks* 16.1 (2003), pp. 79–90.

[41] A.-P. Calderón. "On an inverse boundary value problem". In: *Seminar on Numerical Analysis and its Applications to Continuum Physics (Rio de Janeiro, 1980)*. Rio de Janeiro: Soc. Brasil. Mat., 1980, pp. 65–73.

[42] N. Cao, J. Xie, A. Zhang, S.-Y. Hou, L. Zhang, and B. Zeng. "Neural networks for quantum inverse problems". In: *New Journal of Physics* (2022).

[43] S. Cen, B. Jin, K. Shin, and Z. Zhou. "Electrical impedance tomography with deep Calderon Method". In: *J. Comput. Phys.* 493 (2023), p. 112427.

[44] S. Chaabane, C. Elhechmi, and M. Jaoua. "Error estimates in smoothing noisy data using cubic B-splines". In: *C. R. Math. Acad. Sci. Paris* 346.1-2 (2008), pp. 107–112.

[45] G. Chen, X. Liu, Y. Li, Q. Meng, and L. Chen. "Laplace neural operator for complex geometries". In: *arXiv preprint arXiv:2302.08166* (2023).

[46] J.-S. Chen, M. Hillman, and S.-W. Chi. "Meshfree Methods: Progress Made after 20 Years". In: *Journal of Engineering Mechanics* 143.4 (2017), p. 04017001.

[47] T. Chen and H. Chen. "Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems". In: *IEEE Transactions on Neural Networks* 6.4 (1995), pp. 911–917.

[48] Y. Chen, L. Lu, G. E. Karniadakis, and L. Dal Negro. "Physics-informed neural networks for inverse problems in nano-optics and metamaterials". In: *Optics express* 28.8 (2020), pp. 11618–11633.

[49] Z. Chen, Z. Liu, L. Ai, S. Zhang, and Y. Yang. "Mask-guided spatial–temporal graph neural network for multifrequency electrical impedance tomography". In: *IEEE Trans. Instrum. Meas.* 71 (2022), p. 4505610.

[50] Z. Chen, J. Xiang, P.-O. Bagnaninchi, and Y. Yang. "MMV-Net: A multiple measurement vector network for multifrequency electrical impedance tomography". In: *IEEE Trans. Neural Networks Learn. System* (2022), in press.

[51] Z. Chen and Y. Yang. "Structure-aware dual-branch network for electrical impedance tomography in cell culture imaging". In: *IEEE Trans. Instrum. Meas.* 70 (2021), pp. 1–9.

[52] Z. Chen, Y. Yang, J. Jia, and P. Bagnaninchi. "Deep Learning Based Cell Imaging with Electrical Impedance Tomography". In: *2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. 2020, pp. 1–6.

[53] Z. Chen, J. Lu, and Y. Lu. "On the representation of solutions to elliptic PDEs in Barron spaces". In: *Advances in neural information processing systems* 34 (2021), pp. 6454–6465.

[54] Z. Chen, J. Lu, Y. Lu, and S. Zhou. "A Regularity Theory for Static Schrödinger Equations on ^{d} in Spectral Barron Spaces". In: *SIAM Journal on Mathematical Analysis* 55.1 (2023), pp. 557–570.

[55] M. Cheney and D. Isaacson. “Distinguishability in impedance imaging”. In: *IEEE Trans. Biomed. Imag.* 39.8 (1992), pp. 852–860.

[56] M. Cheney, D. Isaacson, J. C. Newell, S. Simske, and J. Goble. “NOSER: An algorithm for solving the inverse conductivity problem”. In: *Int. J. Imag. Syst. Tech.* 2 (1990), pp. 66–75.

[57] F. Chinesta, A. Huerta, G. Rozza, and K. Willcox. “Model order reduction”. In: *Encyclopedia of computational mechanics* (2016).

[58] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[59] Y. T. Chow, K. Ito, and J. Zou. “A direct sampling method for electrical impedance tomography”. In: *Inverse Problems* 30.9 (2014), p. 095003.

[60] E. T. Chung, T. F. Chan, and X.-C. Tai. “Electrical impedance tomography using level set representation and total variational regularization”. In: *J. Comput. Phys.* 205.1 (2005), pp. 357–372. URL: https://doi.org/10.1016/j.jcp.2004.11.022.

[61] L. Cicci, S. Fresca, and A. Manzoni. “Deep-HyROMnet: A deep learning-based operator approximation for hyper-reduction of nonlinear parametrized PDEs”. In: *arXiv preprint arXiv:2202.02658* (2022).

[62] F. Colibazzi, D. Lazzaro, S. Morigi, and A. Samoré. “Deep-plug-and-play proximal Gauss-Newton method with applications to nonlinear, ill-posed inverse problems”. In: *Inverse Problems and Imaging* (2023).

[63] D. Colombo, E. Turkoglu, W. Li, and D. Rovetta. “Coupled physics-deep learning inversion”. In: *Computers & Geosciences* 157 (2021), p. 104917.

[64] S. Cuomo, V. S. 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”. In: *Journal of Scientific Computing* 92.3 (2022), p. 88.

[65] G. Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[66] Y.-H. Dai, W. W. Hager, K. Schittkowski, and H. Zhang. “The cyclic Barzilai-Borwein method for unconstrained optimization”. In: *IMA J. Numer. Anal.* 26.3 (2006), pp. 604–627. URL: https://doi.org/10.1093/imanum/drl006.

[67] I. Daubechies, M. Defrise, and C. De Mol. “An iterative thresholding algorithm for linear inverse problems with a sparsity constraint”. In: *Comm. Pure Appl. Math.* 57.11 (2004), pp. 1413–1457. URL: https://doi.org/10.1002/cpa.20042.

[68] M. De Hoop, D. Z. Huang, E. Qian, and A. M. Stuart. “The Cost-Accuracy Trade-Off In Operator Learning With Neural Networks”. In: *arXiv preprint arXiv:2203.13181* (2022).

[69] M. V. De Hoop, H. Smith, G. Uhlmann, and R. D. Van Der Hilst. “Seismic imaging with the generalized Radon transform: a curvelet transform perspective”. In: *Inverse Problems* 25.2 (2009), p. 025005.

[70] B. Deng, Y. Shin, L. Lu, Z. Zhang, and G. E. Karniadakis. “Convergence rate of DeepONets for learning operators arising from advection-diffusion equations”. In: *arXiv preprint arXiv:2102.10621* (2021).

[71] J. Desrues, B. Zweschper, and P. Vermeer. *Database for tests on Hostun RF sand.* Publication Series of the Institute of Geotechnik, University Stuttgart. 2000.

[72] R. DeVore, B. Hanin, and G. Petrova. “Neural network approximation”. In: *Acta Numerica* 30 (2021), pp. 327–444.

[73] R. A. DeVore. “Chapter 3: The theoretical foundation of reduced basis methods”. In: *Model reduction and approximation: theory and algorithms.* Ed. by P. Benner, M. Ohlberger, A. Cohen, and K. Willcox. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2017, pp. 137–168.

[74] S. Dittmer, T. Kluth, P. Maass, and D. Otero Baguer. "Regularization by Architecture: A Deep Prior Approach for Inverse Problems". In: *Journal of Mathematical Imaging and Vision* 62.3 (2019), pp. 456–470.

[75] P. Dondl, J. Müller, and M. Zeinhofer. "Uniform Convergence Guarantees for the Deep Ritz Method for Nonlinear Problems". In: *arXiv preprint arXiv:2111.05637* (2021).

[76] C. Drygala, B. Winhart, F. di Mare, and H. Gottschalk. "Generative modeling of turbulence". In: *Physics of Fluids* 34.3 (2022), p. 035114.

[77] C. Duan, Y. Jiao, Y. Lai, X. Lu, and Z. Yang. "Convergence rate analysis for Deep Ritz". In: *arXiv preprint arXiv:2103.13330* (2021).

[78] M. M. Dunlop and A. M. Stuart. "The Bayesian formulation of EIT: analysis and algorithms". In: *Inverse Probl. Imaging* 10.4 (2016), pp. 1007–1036. URL: https://doi.org/10.3934/ipi.2016030.

[79] W. E, J. Han, and A. Jentzen. "Algorithms for solving high dimensional PDEs: From nonlinear Monte Carlo to machine learning". In: *Nonlinearity* 35.1 (2021), p. 278.

[80] W. E, C. Ma, and L. Wu. "The Barron space and the flow-induced function spaces for neural network models". In: *Constructive Approximation* 55.1 (2022), pp. 369–406.

[81] D. Elbrächter, P. Grohs, A. Jentzen, and C. Schwab. "DNN expression rate analysis of high-dimensional PDEs: Application to option pricing". In: *Constructive Approximation* 55.1 (2022), pp. 3–71.

[82] H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of inverse problems*. Vol. 375. Mathematics and its Applications. Kluwer Academic Publishers Group, Dordrecht, 1996, pp. viii+321.

[83] L. C. Evans. *Partial differential equations*. Vol. 19. American Mathematical Society, 2022.

[84] L. D. Faddeev. "Increasing solutions of the Schrödinger equation". In: *Sov.-Phys. Dokl.* 10 (1966), pp. 1033–5.

[85] Y. Fan and L. Ying. "Solving electrical impedance tomography with deep learning". In: *J. Comput. Phys.* 404 (2020), p. 109119.

[86] J. P. Fink and W. C. Rheinboldt. "On the error behavior of the reduced basis technique for nonlinear finite element approximations". In: *Zeitschrift für Angewandte Mathematik und Mechanik* 63.1 (1983), pp. 21–28.

[87] N. R. Franco, S. Fresca, A. Manzoni, and P. Zunino. "Approximation bounds for convolutional neural networks in operator learning". In: *Neural Networks* 161 (2023), pp. 129–141.

[88] D. Fratta, J. Aguettant, and L. Roussel-Smith. *Introduction to Soil Mechanics Laboratory Testing (1st ed.)* Boca Raton, FL: CRC Press, 2007.

[89] W. Freeden. "Geomathematics: Its Role, Its Aim, and Its Potential". In: *Handbook of Geomathematics*. Ed. by W. Freeden, M. Z. Nashed, and T. Sonar. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–42.

[90] W. Freeden and M. Z. Nashed. "Inverse Gravimetry: Density Signatures from Gravitational Potential Data". In: *Handbuch der Geodäsie: 6 Bände*. Ed. by W. Freeden and R. Rummel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019, pp. 1–83.

[91] S. Fresca, L. Dede, and A. Manzoni. "A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs". In: *Journal of Scientific Computing* 87.2 (2021), pp. 1–36.

[92] S. Fresca, G. Gobat, P. Fedeli, A. Frangi, and A. Manzoni. "Deep learning-based reduced order models for the real-time simulation of the nonlinear dynamics of microstructures". In: *International Journal for Numerical Methods in Engineering* 123.20 (2022), pp. 4749–4777.

[93] S. Fresca and A. Manzoni. "Real-time simulation of parameter-dependent fluid flows through deep learning-based reduced order models". In: *Fluids* 6.7 (2021), p. 259.

[94] S. Fresca and A. Manzoni. "POD-DL-ROM: enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition". In: *Computer Methods in Applied Mechanics and Engineering* 388 (2022), p. 114181.

[95] S. Fresca, A. Manzoni, L. Dedè, and A. Quarteroni. "Deep learning-based reduced order models in cardiac electrophysiology". In: *PloS one* 15.10 (2020), e0239416.

[96] S. Garg and S. Chakraborty. "Variational Bayes Deep Operator Network: A data-driven Bayesian solver for parametric differential equations". In: *arXiv preprint arXiv:2206.05655* (2022).

[97] D. Garmatter, B. Haasdonk, and B. Harrach. "A reduced basis Landweber method for nonlinear inverse problems". In: *Inverse Problems* 32.3 (2016), p. 035001.

[98] M. Gehre and B. Jin. "Expectation propagation for nonlinear inverse problems–with an application to electrical impedance tomography". In: *J. Comput. Phys.* 259 (2014), pp. 513–535.

[99] M. Gehre, B. Jin, and X. Lu. "An analysis of finite element approximation in electrical impedance tomography". In: *Inverse Problems* 30.4 (2014), p. 045013.

[100] M. Gehre, T. Kluth, A. Lipponen, B. Jin, A. Seppänen, J. P. Kaipio, and P. Maass. "Sparsity reconstruction in electrical impedance tomography: an experimental evaluation". In: *J. Comput. Appl. Math.* 236.8 (2012), pp. 2126–2136.

[101] G. Gobat, A. Opreni, S. Fresca, A. Manzoni, and A. Frangi. "Reduced order modeling of nonlinear microstructures through Proper Orthogonal Decomposition". In: *Mechanical Systems and Signal Processing* 171 (2022), p. 108864.

[102] L. Gonon, P. Grohs, A. Jentzen, D. Kofler, and D. Šiška. "Uniform error estimates for artificial neural network approximations for heat equations". In: *IMA Journal of Numerical Analysis* 42.3 (2022), pp. 1991–2054.

[103] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. `http://www.deeplearningbook.org`. Cambridge, MA: MIT Press, 2016.

[104] P. Gopalani, S. Karmakar, and A. Mukherjee. "Capacity Bounds for the DeepONet Method of Solving Differential Equations". In: *arXiv preprint arXiv:2205.11359* (2022).

[105] S. Goswami, A. Bora, Y. Yu, and G. E. Karniadakis. "Physics-Informed Deep Neural Operator Networks". In: *arXiv preprint arXiv:2207.05748* (2022).

[106] S. Goswami, M. Yin, Y. Yu, and G. Karniadakis. "A physics-informed variational DeepONet for predicting the crack path in brittle materials". In: *arXiv preprint arXiv:2108.06905* (2021).

[107] R. Gribonval, G. Kutyniok, M. Nielsen, and F. Voigtlaender. "Approximation spaces of deep neural networks". In: *Constructive approximation* 55.1 (2022), pp. 259–367.

[108] P. Grohs, F. Hornung, A. Jentzen, and P. von Wurstemberger. "A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations". In: *arXiv preprint arXiv:1809.02362* (2018).

[109] P. Grohs, S. Ibragimov, A. Jentzen, and S. Koppensteiner. "Lower bounds for artificial neural network approximations: A proof that shallow neural networks fail to overcome the curse of dimensionality". In: *Journal of Complexity* (2023), p. 101746.

[110] P. Grohs, A. Jentzen, and D. Salimova. "Deep neural network approximations for solutions of PDEs based on Monte Carlo algorithms". In: *Partial Differential Equations and Applications* 3.4 (2022), p. 45.

[111] P. Grohs and F. Voigtlaender. "Proof of the theory-to-practice gap in deep learning via sampling complexity bounds for neural network approximation spaces". In: *arXiv preprint arXiv:2104.02746* (2021).

[112] T. G. Grossmann, U. J. Komorowska, J. Latz, and C.-B. Schönlieb. "Can physics-informed neural networks beat the finite element method?" In: *arXiv preprint arXiv:2302.04107* (2023).

[113] B. Guenin, J. Könemann, and L. Tuncel. *A gentle introduction to optimization*. Cambridge University Press, 2014.

[114] I. Gühring, G. Kutyniok, and P. Petersen. "Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms". In: *Analysis and Applications* 18.05 (2020), pp. 803–859.

[115] I. Gühring, M. Raslan, and G. Kutyniok. "Expressivity of deep neural networks". In: *arXiv preprint arXiv:2007.04759* (2020).

[116] R. Guo, S. Cao, and L. Chen. "Transformer meets boundary value inverse problems". In: *The Twelfth International Conference on Learning Representations*. 2023.

[117] R. Guo and J. Jiang. "Construct deep neural networks based on direct sampling methods for solving electrical impedance tomography". In: *SIAM Journal on Scientific Computing* 43.3 (2021), B678–B711.

[118] R. Guo, J. Jiang, and Y. Li. "Learn an Index Operator by CNN for Solving Diffusive Optical Tomography: A Deep Direct Sampling Method". In: *J. Sci. Comput.* 95.1 (2023), p. 31.

[119] G. Gupta, X. Xiao, and P. Bogdan. "Multiwavelet-based operator learning for differential equations". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 24048–24062.

[120] J. Gwinner and E. P. Stephan. *Advanced boundary element methods*. Springer, 2018.

[121] B. Haasdonk. "Reduced basis methods for parametrized PDEs–a tutorial introduction for stationary and instationary problems". In: *Model reduction and approximation: theory and algorithms* 15 (2017), p. 65.

[122] E. Haber and L. Ruthotto. "Stable architectures for deep neural networks". In: *Inverse problems* 34.1 (2017), p. 014004.

[123] J. Hadamard. *Lectures on Cauchy's problem in linear partial differential equations*. Vol. 15. Yale university press, 1923.

[124] P. S. Hadorn. "Shift-DeepONet: Extending Deep Operator Networks for Discontinuous Output Functions". In: *Master thesis, ETH Computational Science and Engineering*. ETH Zurich, Seminar for Applied Mathematics. 2022.

[125] N. Halko, P.-G. Martinsson, Y. Shkolnisky, and M. Tygert. "An algorithm for the principal component analysis of large data sets". In: *SIAM Journal on Scientific computing* 33.5 (2011), pp. 2580–2594.

[126] N. Halko, P.-G. Martinsson, and J. A. Tropp. "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions". In: *SIAM review* 53.2 (2011), pp. 217–288.

[127] S. J. Hamilton, A. Hänninen, A. Hauptmann, and V. Kolehmainen. "Beltrami-net: domain-independent deep D-bar learning for absolute imaging with electrical impedance tomography (a-EIT)". In: *Physiol. Meas.* 40.7 (2019), p. 074002.

[128] S. J. Hamilton and A. Hauptmann. "Deep D-bar: Real-time electrical impedance tomography imaging with deep neural networks". In: *IEEE transactions on medical imaging* 37.10 (2018), pp. 2367–2377.

[129] S. J. Hamilton, A. Hauptmann, and S. Siltanen. "A data-driven edge-preserving D-bar method for electrical impedance tomography". In: *Inverse Probl. Imaging* 8.4 (2014), pp. 1053–1072. URL: `https://doi.org/10.3934/ipi.2014.8.1053`.

[130] J. Han, A. Jentzen, and W. E. "Solving high-dimensional partial differential equations using deep learning". In: *Proceedings of the National Academy of Sciences* 115.34 (2018), pp. 8505–8510.

[131] B. Harrach and M. Ullrich. "Monotonicity-based shape reconstruction in electrical impedance tomography". In: *SIAM J. Math. Anal.* 45.6 (2013), pp. 3382–3403. URL: https://doi.org/10.1137/120886984.

[132] A. Hauptmann and B. T. Cox. "Deep learning in photoacoustic tomography: current approaches and future directions". In: *Journal of Biomedical Optics* 25.11 (2020), p. 112903.

[133] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1998.

[134] D. Hendrycks and K. Gimpel. "Bridging nonlinearities and stochastic regularizers with gaussian error linear units". In: *CoRR, abs/1606.08415* 3 (2016).

[135] D. Hendrycks and K. Gimpel. "Gaussian error linear units (gelus)". In: *arXiv preprint arXiv:1606.08415* (2016).

[136] W. Herzberg, A. Hauptmann, and S. J. Hamilton. "Domain independent post-processing with graph U-nets: Applications to Electrical Impedance Tomographic Imaging". In: *arXiv preprint arXiv:2305.05020* (2023).

[137] W. Herzberg, D. B. Rowe, A. Hauptmann, and S. J. Hamilton. "Graph convolutional networks for model-based learning in nonlinear inverse problems". In: *IEEE transactions on computational imaging* 7 (2021), pp. 1341–1353.

[138] W. Herzberg, D. B. Rowe, A. Hauptmann, and S. J. Hamilton. "Graph convolutional networks for model-based learning in nonlinear inverse problems". In: *IEEE Trans. Comput. Imag.* 7 (2021), pp. 1341–1353.

[139] J. S. Hesthaven and S. Ubbiali. "Non-intrusive reduced order modeling of nonlinear problems using neural networks". In: *Journal of Computational Physics* 363 (2018), pp. 55–78.

[140] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". In: *IEEE Signal processing magazine* 29.6 (2012), pp. 82–97.

[141] M. Hinze, B. Kaltenbacher, and T. N. T. Quyen. "Identifying conductivity in electrical impedance tomography with total variation regularization". In: *Numer. Math.* 138.3 (2018), pp. 723–765. URL: https://doi.org/10.1007/s00211-017-0920-8.

[142] S. Hochreiter and J. Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[143] D. S. Holder, ed. *Electrical Impedance Tomography: Methods, History and Applications*. Series in Medical Physics and Biomedical Engineering. Bristol: Institute of Physics Publishing, 2004.

[144] M. de Hoop and Huang. *The Cost-Accuracy Trade-Off In Operator Learning With Neural Networks*. Apr. 2022.

[145] K. Hornik, M. Stinchcombe, and H. White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[146] F. Hornung, A. Jentzen, and D. Salimova. "Space-time deep neural network approximations for high-dimensional partial differential equations". In: *arXiv preprint arXiv:2006.02199* (2020).

[147] D. Z. Huang, K. Xu, C. Farhat, and E. Darve. "Learning constitutive relations from indirect observations using deep neural networks". In: *Journal of Computational Physics* 416 (2020), p. 109491.

[148] S.-W. Huang, H.-M. Cheng, and S.-F. Lin. “Improved imaging resolution of electrical impedance tomography using artificial neural networks for image reconstruction”. In: *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE. 2019, pp. 1551–1554.

[149] C. Huré, H. Pham, and X. Warin. “Deep backward schemes for high-dimensional nonlinear PDEs”. In: *Mathematics of Computation* 89.324 (2020), pp. 1547–1579.

[150] M. Hutzenthaler, A. Jentzen, T. Kruse, et al. “On multilevel Picard numerical approximations for high-dimensional nonlinear parabolic partial differential equations and high-dimensional nonlinear backward stochastic differential equations”. In: *Journal of Scientific Computing* 79.3 (2019), pp. 1534–1571.

[151] M. Hutzenthaler, A. Jentzen, T. Kruse, and T. A. Nguyen. “A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations”. In: *SN partial differential equations and applications* 1 (2020), pp. 1–34.

[152] N. Hyvönen. “Complete electrode model of electrical impedance tomography: approximation properties and characterization of inclusions”. In: *SIAM J. Appl. Math.* 64.3 (2004), pp. 902–931. URL: https://doi.org/10.1137/S0036139903423303.

[153] M. Ikehata. “Size estimation of inclusion”. In: *J. Inverse Ill-Posed Probl.* 6.2 (1998), pp. 127–140. URL: https://doi.org/10.1515/jiip.1998.6.2.127.

[154] V. Isakov. *Inverse Problems for Partial Differential Equations (2nd ed.)* Vol. 127. Applied Mathematical Sciences. New York: Springer, 2006, pp. xiv+344.

[155] K. Ito and B. Jin. *Inverse problems: Tikhonov theory and algorithms*. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2015, pp. x+318.

[156] F. ITWM and F. SCAI. *MESHFREE Homepage*. https://www.meshfree.eu. Accessed: 2023-01-06. 2023.

[157] A. Jacot, F. Gabriel, and C. Hongler. “Neural Tangent Kernel: Convergence and Generalization in Neural Networks”. In: *NIPS’18*. Montréal, Canada: Curran Associates Inc., 2018, pp. 8580–8589.

[158] A. D. Jagtap and G. E. Karniadakis. “Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations”. In: *Communications in Computational Physics* 28.5 (2020), pp. 2002–2041.

[159] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis. “Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems”. In: *Computer Methods in Applied Mechanics and Engineering* 365 (2020), p. 113028.

[160] A. Jefferies, J. Kuhnert, L. Aschenbrenner, and U. Giffhorn. “Finite pointset method for the simulation of a vehicle travelling through a body of water”. In: *Meshfree Methods for Partial Differential Equations VII. Lecture Notes in Computational Science and Engineering*. Ed. by M. Griebel and M. A. Schweitzer. Vol. 100. Cham: Springer International Publishing, 2015, pp. 205–221.

[161] A. Jentzen, D. Salimova, and T. Welti. “A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of Kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients”. In: *arXiv preprint arXiv:1809.07321* (2018).

[162] Y. Jiao, Y. Lai, Y. Lo, Y. Wang, and Y. Yang. “Error analysis of Deep Ritz methods for elliptic equations”. In: *arXiv preprint arXiv:2107.14478* (2021).

[163] B. Jin, T. Khan, P. Maass, and M. Pidcock. “Function spaces and optimal currents in impedance tomography”. In: *J. Inverse Ill-Posed Probl.* 19.1 (2011), pp. 25–48. URL: https://doi.org/10.1515/JIIP.2011.022.

[164] B. Jin, P. Maaß, and O. Scherzer. “Sparsity regularization in inverse problems”. In: *Inverse Problems* 33.6 (2017), p. 060301.

[165] B. Jin and P. Maass. “An analysis of electrical impedance tomography with applications to Tikhonov regularization”. In: *ESAIM: Control, Optim. Cal. Var.* 18.4 (2012), pp. 1027–1048.

[166] B. Jin and P. Maass. “Sparsity regularization for parameter identification problems”. In: *Inverse Problems* 28.12 (Nov. 2012), p. 123001. URL: https://dx.doi.org/10.1088/0266-5611/28/12/123001.

[167] B. Jin and Y. Xu. “Adaptive reconstruction for electrical impedance tomography with a piecewise constant conductivity”. In: *Inverse Problems* 36.1 (2019), p. 014003.

[168] B. Jin, Y. Xu, and J. Zou. “A convergent adaptive finite element method for electrical impedance tomography”. In: *IMA J. Numer. Anal.* 37.3 (2017), pp. 1520–1550.

[169] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser. “Deep Convolutional Neural Network for Inverse Problems in Imaging”. In: *IEEE Trans. Image Proc.* 26.9 (2017), pp. 4509–4522.

[170] P. Jin, S. Meng, and L. Lu. “MIONet: Learning multiple-input operators via tensor product”. In: *arXiv preprint arXiv:2202.06137* (2022).

[171] I. T. Jolliffe and J. Cadima. “Principal component analysis: a review and recent developments”. In: *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences* 374.2065 (2016), p. 20150202.

[172] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. “Highly accurate protein structure prediction with AlphaFold”. In: *Nature* 596.7873 (2021), pp. 583–589.

[173] J. P. Kaipio, V. Kolehmainen, E. Somersalo, and M. Vauhkonen. “Statistical inversion and Monte Carlo sampling methods in electrical impedance tomography”. In: *Inverse Problems* 16.5 (2000), pp. 1487–1522. URL: https://doi.org/10.1088/0266-5611/16/5/321.

[174] M. Kärcher. “Certified reduced basis methods for parametrized PDE-constrained optimization problems”. PhD thesis. Universitätsbibliothek der RWTH Aachen, 2016.

[175] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. “Physics-informed machine learning”. In: *Nature Reviews Physics* 3.6 (2021), pp. 422–440.

[176] T. A. Khan and S. H. Ling. “Review on electrical impedance tomography: Artificial intelligence methods and its applications”. In: *Algorithms* 12.5 (2019), p. 88.

[177] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference for Learning Representations*. San Diego, 2015.

[178] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. “Self-normalizing neural networks”. In: *Advances in neural information processing systems* 30 (2017).

[179] I. Knowles. “A variational algorithm for electrical impedance tomography”. In: *Inverse Problems* 14.6 (1998), pp. 1513–1525. URL: https://doi.org/10.1088/0266-5611/14/6/010.

[180] K. Knudsen, M. Lassas, J. L. Mueller, and S. Siltanen. “Regularized D-bar method for the inverse conductivity problem”. In: *Inverse Probl. Imaging* 3.4 (2009), pp. 599–624. URL: https://doi.org/10.3934/ipi.2009.3.599.

[181] A. Kobrunov. “The method of functional representations in the solution of inverse problems of gravimetry”. In: *Izvestiya, Physics of the Solid Earth* 51 (2015), pp. 459–468.

[182] R. V. Kohn and A. McKenney. “Numerical implementation of a variational method for electrical impedance tomography”. In: *Inverse Problems* 6.3 (1990), pp. 389–414. URL: http://stacks.iop.org/0266-5611/6/389.

[183] D. Kolymbas. “Barodesy: a new constitutive frame for soils”. In: *Géotechnique Letters* 2.2 (2012), pp. 17–23.

[184] D. Kolymbas. "Barodesy: a new hypoplastic approach". In: *International Journal for Numerical and Analytical Methods in Geomechanics* 36.9 (2012), pp. 1220–1240.

[185] T. Konuk and J. Shragge. "Physics-guided deep learning using Fourier neural operators for solving the acoustic VTI wave equation". In: *82nd EAGE Annual Conference & Exhibition*. Vol. 2021. 1. European Association of Geoscientists & Engineers. 2021, pp. 1–5.

[186] N. Kovachki, S. Lanthaler, and S. Mishra. "On universal approximation and error bounds for Fourier Neural Operators". In: *Journal of Machine Learning Research* 22 (2021), Art–No.

[187] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar. "Neural operator: Learning maps between function spaces". In: *arXiv preprint arXiv:2108.08481* (2021).

[188] N. Kovachki, B. Liu, X. Sun, H. Zhou, K. Bhattacharya, M. Ortiz, and A. Stuart. "Multiscale modeling of materials: Computing, data science, uncertainty and goal-oriented optimization". In: *Mechanics of Materials* 165 (2022), p. 104156.

[189] H. Kraus, J. Kuhnert, A. Meister, and P. Suchde. "A meshfree point collocation method for elliptic interface problems". In: *Applied Mathematical Modelling* 113 (2023). `https://doi.org/10.1016/j.apm.2022.08.002`, pp. 241–261.

[190] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[191] J. Kuhnert. "Meshfree numerical schemes for time dependent problems in fluid and continuum mechanics". In: *Advances in PDE Modeling and Computation*. Ed. by S. Sundar. New Delhi: Ane Books, 2014, pp. 119–136.

[192] J. Kuhnert. "MESHFREE simulations in car design: closing the gaps of classical simulation tools". In: *German Success Stories in Industrial Mathematics* 35 (2021), p. 130.

[193] J. Kuhnert, I. Michel, and R. Mack. "Fluid structure interaction (fsi) in the meshfree finite pointset method (fpm): Theory and applications". In: *Meshfree Methods for Partial Differential Equations IX. IWMMPDE 2017. Lecture Notes in Computational Science and Engineering*. Ed. by M. Griebel and M. A. Schweitzer. Vol. 129. Springer, Cham. 2017, pp. 73–92.

[194] G. Kutyniok, P. Petersen, M. Raslan, and R. Schneider. "A theoretical analysis of deep neural networks and parametric PDEs". In: *Constructive Approximation* 55.1 (2022), pp. 73–125.

[195] S. Lanthaler. "Operator learning with PCA-Net: upper and lower complexity bounds". In: *arXiv preprint arXiv:2303.16317* (2023).

[196] S. Lanthaler, Z. Li, and A. M. Stuart. "The Nonlocal Neural Operator: Universal Approximation". In: *arXiv preprint arXiv:2304.13221* (2023).

[197] S. Lanthaler, S. Mishra, and G. E. Karniadakis. "Error estimates for DeepONets: A deep learning framework in infinite dimensions". In: *Transactions of Mathematics and Its Applications* 6.1 (2022), tnac001.

[198] A. Lechleiter. "The MUSIC algorithm for impedance tomography of small inclusions from discrete data". In: *Inverse Problems* 31.9 (2015), pp. 095004, 19. URL: `https://doi.org/10.1088/0266-5611/31/9/095004`.

[199] A. Lechleiter and A. Rieder. "Newton regularizations for impedance tomography: a numerical study". In: *Inverse Problems* 22.6 (2006), pp. 1967–1987. URL: `https://doi.org/10.1088/0266-5611/22/6/004`.

[200] A. Lechleiter and A. Rieder. "Newton regularizations for impedance tomography: convergence by local injectivity". In: *Inverse Problems* 24.6 (2008), pp. 065009, 18. URL: `https://doi.org/10.1088/0266-5611/24/6/065009`.

[201] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[202] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE* (1998).

[203] K. Lee and K. T. Carlberg. "Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders". In: *Journal of Computational Physics* 404 (2020), p. 108973.

[204] J. Leuschner, M. Schmidt, D. O. Baguer, and P. Maass. "LoDoPaB-CT, a benchmark dataset for low-dose computed tomography reconstruction". In: *Scientific Data* 8.1 (2021), p. 109.

[205] J. Leuschner, M. Schmidt, P. S. Ganguly, V. Andriiashen, S. B. Coban, A. Denker, D. Bauer, A. Hadjifaradji, K. J. Batenburg, P. Maass, et al. "Quantitative comparison of deep learning-based image reconstruction methods for low-dose and sparse-angle CT applications". In: *Journal of Imaging* 7.3 (2021), p. 44.

[206] R. J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002.

[207] H. Li, J. Schwab, S. Antholzer, and M. Haltmeier. "NETT: Solving inverse problems with deep neural networks". In: *Inverse Problems* 36.6 (2020), p. 065005.

[208] X. Li, R. Lu, Q. Wang, J. Wang, X. Duan, Y. Sun, X. Li, and Y. Zhou. "One-dimensional convolutional neural network (1D-CNN) image reconstruction for electrical impedance tomography". In: *Rev. Sci. Instrument.* 91.12 (2020).

[209] X. Li, Y. Lu, J. Wang, X. Dang, Q. Wang, X. Duan, and Y. Sun. "An image reconstruction framework based on deep neural network for electrical impedance tomography". In: *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2017, pp. 3585–3589.

[210] Z. Li, D. Z. Huang, B. Liu, and A. Anandkumar. "Fourier neural operator with learned deformations for PDEs on general geometries". In: *arXiv preprint arXiv:2207.05209* (2022).

[211] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. "Fourier neural operator for parametric partial differential equations". In: *arXiv preprint arXiv:2010.08895* (2020).

[212] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. "Neural operator: Graph kernel network for partial differential equations". In: *arXiv preprint arXiv:2003.03485* (2020).

[213] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, A. Stuart, K. Bhattacharya, and A. Anandkumar. "Multipole graph neural operator for parametric partial differential equations". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6755–6766.

[214] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar. "Physics-informed neural operator for learning partial differential equations". In: *arXiv preprint arXiv:2111.03794* (2021).

[215] Y. Liao and P. Ming. "Deep Nitsche method: Deep Ritz method with essential boundary conditions". In: *arXiv preprint arXiv:1912.01309* (2019).

[216] G. Lin, C. Moya, and Z. Zhang. "Accelerated replica exchange stochastic gradient Langevin diffusion enhanced Bayesian DeepONet for solving noisy parametric PDEs". In: *arXiv preprint arXiv:2111.02484* (2021).

[217] B. Liu, N. Kovachki, Z. Li, K. Azizzadenesheli, A. Anandkumar, A. M. Stuart, and K. Bhattacharya. "A learning-based multiscale method and its application to inelastic impact problems". In: *Journal of the Mechanics and Physics of Solids* 158 (2022), p. 104668.

[218] D. Liu, J. Wang, Q. Shan, D. Smyl, J. Deng, and J. Du. "DeepEIT: deep image prior enabled electrical impedance tomography". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 45.8 (2023), pp. 9627–9638.

[219] G. Liu, K. Zaw, and Y. Wang. "Rapid inverse parameter estimation using reduced-basis approximation with asymptotic error estimation". In: *Computer methods in applied mechanics and engineering* 197.45-48 (2008), pp. 3898–3910.

[220] L. Liu and W. Cai. "Multiscale DeepONet for Nonlinear Operators in Oscillatory Function Spaces for Building Seismic Wave Responses". In: *arXiv preprint arXiv:2111.04860* (2021).

[221] J. Lu, Z. Shen, H. Yang, and S. Zhang. "Deep network approximation for smooth functions". In: *SIAM Journal on Mathematical Analysis* 53.5 (2021), pp. 5465–5506.

[222] L. Lu, P. Jin, and G. E. Karniadakis. "DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators". In: *arXiv preprint arXiv:1910.03193* (2019).

[223] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators". In: *Nature Machine Intelligence* 3.3 (2021), pp. 218–229.

[224] L. Lu, X. Meng, S. Cai, Z. Mao, S. Goswami, Z. Zhang, and G. E. Karniadakis. "A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data". In: *Computer Methods in Applied Mechanics and Engineering* 393 (2022), p. 114778.

[225] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis. "DeepXDE: A deep learning library for solving differential equations". In: *SIAM Review* 63.1 (2021), pp. 208–228.

[226] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson. "Physics-informed neural networks with hard constraints for inverse design". In: *SIAM Journal on Scientific Computing* 43.6 (2021), B1105–B1132.

[227] M. Lukaschewitsch, P. Maass, and M. Pidcock. "Tikhonov regularization for electrical impedance tomography on unbounded domains". In: *Inverse Problems* 19.3 (2003), pp. 585–610. URL: https://doi.org/10.1088/0266-5611/19/3/308.

[228] S. Martin and C. T. Choi. "A post-processing method for three-dimensional electrical impedance tomography". In: *Sci. Rep.* 7.1 (2017), p. 7212.

[229] N. G. Meyers. "An L^pe-estimate for the gradient of solutions of second order elliptic divergence equations". In: *Ann. Scuola Norm. Sup. Pisa Cl. Sci. (3)* 17 (1963), pp. 189–206.

[230] H. N. Mhaskar. "Neural networks for optimal approximation of smooth and analytic functions". In: *Neural computation* 8.1 (1996), pp. 164–177.

[231] I. Michel, S. Bathaeian, J. Kuhnert, D. Kolymbas, C.-H. Chen, I. Polymerou, C. Vrettos, and A. Becker. "Meshfree generalized finite difference methods in soil mechanics—part ii: numerical results". In: *GEM-International Journal on Geomathematics* 8.2 (2017), pp. 191–217.

[232] I. Michel, T. Seifarth, J. Kuhnert, and P. Suchde. "A meshfree generalized finite difference method for solution mining processes". In: *Computational Particle Mechanics* 8.3 (2021), pp. 561–574.

[233] P. M. Milani, J. Ling, and J. K. Eaton. "Generalization of machine-learned turbulent heat flux models applied to film cooling flows". In: *Journal of Turbomachinery* 142.1 (2020).

[234] S. Mishra and R. Molinaro. "Estimates on the generalization error of Physics Informed Neural Networks (PINNs) for approximating a class of inverse problems for PDEs". In: *arXiv preprint arXiv:2007.01138* (2020).

[235] S. Mishra and R. Molinaro. "Estimates on the generalization error of Physics Informed Neural Networks (PINNs) for approximating PDEs". In: *arXiv preprint arXiv:2006.16144* (2020).

[236] V. Monga, Y. Li, and Y. C. Eldar. "Algorithm unrolling: interpretable, efficient deep learning for signal and image processing". In: *IEEE Signal Proc. Magaz.* 38.2 (2021), pp. 18–44.

[237] C. Moya and G. Lin. "Fed-DeepONet: Stochastic Gradient-Based Federated Training of Deep Operator Networks". In: *Algorithms* 15.9 (2022), p. 325.

[238] C. Moya, S. Zhang, M. Yue, and G. Lin. "DeepONet-Grid-UQ: A Trustworthy Deep Operator Framework for Predicting the Power Grid's Post-Fault Trajectories". In: *arXiv preprint arXiv:2202.07176* (2022).

[239] M. Mozumder, A. Hauptmann, I. Nissilä, S. R. Arridge, and T. Tarvainen. "A model-based iterative learning approach for diffuse optical tomography". In: *IEEE Transactions on Medical Imaging* 41.5 (2021), pp. 1289–1299.

[240] J. L. Mueller and S. Siltanen. "The D-bar method for electrical impedance tomography—demystified". In: *Inverse Problems* 36.9 (2020), pp. 093001, 28. URL: https://doi.org/10.1088/1361-6420/aba2f5.

[241] J. Müller and M. Zeinhofer. "Deep Ritz revisited". In: *arXiv preprint arXiv:1912.03937* (2019).

[242] A. I. Nachman. "Global uniqueness for a two-dimensional inverse boundary value problem". In: *Ann. of Math. (2)* 143.1 (1996), pp. 71–96. URL: https://doi.org/10.2307/2118653.

[243] J. W. Neuberger. *Sobolev gradients and differential equations*. Vol. 1670. Lecture Notes in Mathematics. Springer-Verlag, Berlin, 1997, pp. viii+150. URL: https://doi.org/10.1007/BFb0092831.

[244] D. Nganyu Tanyu, J. Ning, T. Freudenberg, N. Heilenkoetter, A. Rademacher, U. Iben, and P. Maass. "Deep learning methods for partial differential equations and related parameter identification problems". In: *Inverse Problems* 39.10 (Aug. 2023), p. 103001. URL: https://dx.doi.org/10.1088/1361-6420/ace9d4.

[245] M. A. Nielsen. *Neural Networks and Deep Learning: A Textbook*. Online resource. 2015. URL: http://neuralnetworksanddeeplearning.com/.

[246] J. Ning, F. Han, and J. Zou. "A Direct Sampling-Based Deep Learning Approach for Inverse Medium Scattering Problems". In: *arXiv preprint arXiv:2305.00250* (2023).

[247] G. Nolet. "Transmission Tomography in Seismology". In: *Handbook of Geomathematics*. Ed. by W. Freeden, M. Z. Nashed, and T. Sonar. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 1887–1904.

[248] A. K. Noor and J. M. Peters. "Reduced basis technique for nonlinear analysis of structures". In: *Aiaa journal* 18.4 (1980), pp. 455–462.

[249] R. G. Novikov. "A multidimensional inverse spectral problem for the equation $-\Delta\psi + (v(x) - Eu(x))\psi = 0$". In: *Funktsional. Anal. i Prilozhen.* 22.4 (1988), pp. 11–22, 96. URL: https://doi.org/10.1007/BF01077418.

[250] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. "Activation functions: Comparison of trends in practice and research for deep learning". In: *arXiv preprint arXiv:1810.04805* (2018).

[251] I. Ostermann, J. Kuhnert, D. Kolymbas, C.-H. Chen, I. Polymerou, V. Šmilauer, C. Vrettos, and D. Chen. "Meshfree generalized finite difference methods in soil mechanics—part I: theory". In: *GEM-International Journal on Geomathematics* 4.2 (2013), pp. 167–184.

[252] G. Pang, L. Lu, and G. E. Karniadakis. "fPINNs: Fractional physics-informed neural networks". In: *SIAM Journal on Scientific Computing* 41.4 (2019), A2603–A2626.

[253] A. T. Patera, G. Rozza, et al. *Reduced basis approximation and a posteriori error estimation for parametrized partial differential equations*. 2007.

[254] P. Petersen and F. Voigtlaender. "Optimal approximation of piecewise smooth functions using deep ReLU neural networks". In: *Neural Networks* 108 (2018), pp. 296–330.

[255] A. Pinkus. "Approximation theory of the MLP model in neural networks". In: *Acta numerica* 8 (1999), pp. 143–195.

[256] A. Pokkunuru, P. Rooshenas, T. Strauss, A. Abhishek, and T. Khan. "Improved Training of Physics-Informed Neural Networks Using Energy-Based Priors: a Study on Electrical Impedance Tomography". In: *The Eleventh International Conference on Learning Representations*. 2022.

[257] T. Porsching and M. L. Lee. "The reduced basis method for initial value problems". In: *SIAM Journal on Numerical Analysis* 24.6 (1987), pp. 1277–1287.

[258] M. Prasthofer, T. De Ryck, and S. Mishra. "Variable-input deep operator networks". In: *arXiv preprint arXiv:2205.11404* (2022).

[259] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations: An Introduction*. UNITEXT. Cham: Springer, 2015.

[260] M. Raissi. "Deep hidden physics models: Deep learning of nonlinear partial differential equations". In: *The Journal of Machine Learning Research* 19.1 (2018), pp. 932–955.

[261] 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". In: *Journal of Computational Physics* 378 (2019), pp. 686–707.

[262] B. Raonić, R. Molinaro, T. Rohner, S. Mishra, and E. de Bezenac. "Convolutional Neural Operators". In: *arXiv preprint arXiv:2302.01178* (2023).

[263] C. Reisinger and Y. Zhang. "Rectified deep neural networks overcome the curse of dimensionality for nonsmooth value functions in zero-sum games of nonlinear stiff systems". In: *Analysis and Applications* 18.06 (2020), pp. 951–999.

[264] S. Ren, R. Guan, G. Liang, and F. Dong. "RCRC: A deep neural network for dynamic image reconstruction of electrical impedance tomography". In: *IEEE Trans. Instrum. Meas.* 70 (2021), pp. 1–11.

[265] G. S. Reuber. "Statistical and deterministic inverse methods in the geosciences: introduction, review, and application to the nonlinear diffusion equation". In: *GEM-International Journal on Geomathematics* 12.1 (2021), p. 19.

[266] G. S. Reuber, B. J. Kaus, A. A. Popov, and T. S. Baumann. "Unraveling the physics of the Yellowstone magmatic system using geodynamic simulations". In: *Frontiers in Earth Science* 6 (2018), p. 117.

[267] G. S. Reuber and F. J. Simons. "Multi-physics adjoint modeling of Earth structure: combining gravimetric, seismic, and geodynamic inversions". In: *GEM-International Journal on Geomathematics* 11.1 (2020), p. 30.

[268] V. Rokhlin, A. Szlam, and M. Tygert. "A randomized algorithm for principal component analysis". In: *SIAM Journal on Matrix Analysis and Applications* 31.3 (2010), pp. 1100–1124.

[269] L. Rondi. "Discrete approximation and regularisation for the inverse conductivity problem". In: *Rend. Istit. Mat. Univ. Trieste* 48 (2016), pp. 315–352. URL: https://doi.org/10.13137/2464-8728/13162.

[270] L. Rondi and F. Santosa. "Enhanced electrical impedance tomography via the Mumford-Shah functional". In: *ESAIM Control Optim. Calc. Var.* 6 (2001), pp. 517–538. URL: https://doi.org/10.1051/cocv:2001121.

[271] O. Ronneberger, P. Fischer, and T. Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer. 2015, pp. 234–241.

[272] G. Rozza, D. B. P. Huynh, and A. T. Patera. "Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations". In: *Archives of Computational Methods in Engineering* 15.3 (2008), pp. 229–275.

[273] D. E. Rumelhart, G. E. Hinton, R. J. Williams, et al. *Learning internal representations by error propagation*. 1985.

[274] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536.

[275] L. Ruthotto and E. Haber. "Deep neural networks motivated by partial differential equations". In: *Journal of Mathematical Imaging and Vision* 62.3 (2020), pp. 352–364.

[276] W. Schilders. "Introduction to model order reduction". In: *Model order reduction: theory, research aspects and applications* (2008), pp. 3–32.

[277] W. Schilders. "MSODE: Modelling, Simulation and Optimization in a Data-rich Environment". In: *Mathematics: Key Enabling Technology for Scientific Machine Learning*. 2021, pp. 24–30.

[278] T. Schuster, B. Kaltenbacher, B. Hofmann, and K. S. Kazimierski. *Regularization methods in Banach spaces*. Vol. 10. Radon Series on Computational and Applied Mathematics. Walter de Gruyter GmbH & Co. KG, Berlin, 2012, pp. xii+283. URL: https://doi.org/10.1515/9783110255720.

[279] J. K. Seo, K. C. Kim, A. Jargal, K. Lee, and B. Harrach. "A learning-based method for solving ill-posed nonlinear inverse problems: a simulation study of lung EIT". In: *SIAM journal on Imaging Sciences* 12.3 (2019), pp. 1275–1295.

[280] U. Shaham, A. Cloninger, and R. R. Coifman. "Provable approximation properties for deep neural networks". In: *Applied and Computational Harmonic Analysis* 44.3 (2018), pp. 537–557.

[281] K. Shin and J. L. Mueller. "A second order Calderón's method with a correction term and *a priori* information". In: *Inverse Problems* 36.12 (2020), pp. 124005, 22. URL: https://doi.org/10.1088/1361-6420/abb014.

[282] S. Siltanen and T. Ide. "Electrical impedance tomography, enclosure method and machine learning". In: *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2020, pp. 1–6.

[283] S. Siltanen, J. Mueller, and D. Isaacson. "An implementation of the reconstruction algorithm of A. Nachman for the 2D inverse conductivity problem". In: *Inverse Problems* 16.3 (2000), pp. 681–699. URL: https://doi.org/10.1088/0266-5611/16/3/310.

[284] J. Sirignano and K. Spiliopoulos. "DGM: A deep learning algorithm for solving partial differential equations". In: *Journal of computational physics* 375 (2018), pp. 1339–1364.

[285] D. Smyl, T. N. Tallman, D. Liu, and A. Hauptmann. "An efficient quasi-Newton method for nonlinear inverse problems via learned singular values". In: *IEEE Signal Processing Letters* 28 (2021), pp. 748–752.

[286] E. Somersalo, M. Cheney, and D. Isaacson. "Existence and uniqueness for electrode models for electric current computed tomography". In: *SIAM J. Appl. Math.* 52.4 (1992), pp. 1023–1040. URL: https://doi.org/10.1137/0152060.

[287] W. A. Strauss. *Partial differential equations: An introduction*. John Wiley & Sons, 2007.

[288] J. C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations*. second. Society for Industrial and Applied Mathematics, 2004.

[289] A. M. Stuart. “Inverse problems: a Bayesian perspective”. In: *Acta numerica* 19 (2010), pp. 451–559.

[290] P. Suchde. “Conservation and Accuracy in Meshfree Generalized Finite Difference Methods”. PhD thesis. Germany: University of Kaiserslautern, 2018.

[291] P. Suchde, T. Jacquemin, and O. Davydov. “Point Cloud Generation for Meshfree Methods: An Overview”. In: *Archives of Computational Methods in Engineering* (2022). https://doi.org/10.1007/s11831-022-09820-w.

[292] P. Suchde and J. Kuhnert. “Point cloud movement for fully Lagrangian meshfree methods”. In: *Journal of Computational and Applied Mathematics* 340 (2018). https://doi.org/10.1016/j.cam.2018.02.020, pp. 89–100.

[293] B. Sun, H. Zhong, Y. Zhao, L. Ma, and H. Wang. “Calderón’s method-guided deep neural network for electrical impedance tomography”. In: *IEEE Trans. Instrum. Meas.* (2023), in press.

[294] Y. Sun, C. Moya, G. Lin, and M. Yue. “DeepGraphONet: A Deep Graph Operator Network to Learn and Zero-shot Transfer the Dynamic Response of Networked Systems”. In: *arXiv preprint arXiv:2209.10622* (2022).

[295] I. Sutskever, O. Vinyals, and Q. V. Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems* 27 (2014).

[296] C. Szegedy and et al. “Going Deeper with Convolutions”. In: *arXiv preprint arXiv:1409.4842* (2014).

[297] C. Tan, S. Lv, F. Dong, and M. Takei. “Image reconstruction based on convolutional neural network for electrical resistance tomography”. In: *IEEE Sensors Journal* 19.1 (2018), pp. 196–204.

[298] L. Tan and L. Chen. “Enhanced DeepONet for Modeling Partial Differential Operators Considering Multiple Input Functions”. In: *arXiv preprint arXiv:2202.08942* (2022).

[299] D. N. Tanyu, I. Michel, A. Rademacher, J. Kuhnert, and P. Maass. “Parameter identification by deep learning of a material model for granular media”. In: *arXiv preprint arXiv:2307.04166* (2023).

[300] S. Tejomurtula and S. Kak. “Inverse kinematics in robotics using neural networks”. In: *Information sciences* 116.2-4 (1999), pp. 147–164.

[301] N. Thuerey, P. Holl, M. Mueller, P. Schnell, F. Trost, and K. Um. “Physics-based Deep Learning”. In: *arXiv preprint arXiv:2109.05237* (2021).

[302] T. Tripura and S. Chakraborty. “Wavelet neural operator: a neural operator for parametric partial differential equations”. In: *arXiv preprint arXiv:2205.02191* (2022).

[303] F. Tröltzsch. *Optimal Control of Partial Differential Equations*. Vol. 112. Graduate Studies in Mathematics. Providence: American Mathematical Society, 2010.

[304] E. Uhlmann, E. Barth, T. Seifarth, M. Höchel, J. Kuhnert, and A. Eisenträger. “Simulation of metal cutting with cutting fluid using the Finite-Pointset-Method”. In: *Procedia CIRP* 101 (2021). https://doi.org/10.1016/j.procir.2021.02.013, pp. 98–101.

[305] G. Uhlmann. “Electrical impedance tomography and Calderón’s problem”. In: *Inverse Problems* 25.12 (2009), pp. 123011, 39. URL: https://doi.org/10.1088/0266-5611/25/12/123011.

[306] D. Ulyanov, A. Vedaldi, and V. Lempitsky. “Deep image prior”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9446–9454.

[307] L. Veltmaat, F. Mehrens, H.-J. Endres, J. Kuhnert, and P. Suchde. “Mesh-free simulations of injection molding processes”. In: *Physics of Fluids* 34 (2022). https://doi.org/10.1063/5.0085049, p. 033102.

[308] F. Voigtlaender. "The universal approximation theorem for complex-valued neural networks". In: *Applied and Computational Harmonic Analysis* 64 (2023), pp. 33–61.

[309] R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu. "Towards physics-informed deep learning for turbulent flow prediction". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 1457–1466.

[310] S. Wang, H. Wang, and P. Perdikaris. "Learning the solution operator of parametric partial differential equations with physics-informed DeepOnets". In: *arXiv preprint arXiv:2103.10974* (2021).

[311] S. Wang, H. Wang, and P. Perdikaris. "On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks". In: *Computer Methods in Applied Mechanics and Engineering* 384 (2021), p. 113938.

[312] S. Wang, H. Wang, and P. Perdikaris. "Improved Architectures and Training Algorithms for Deep Operator Networks". In: *Journal of Scientific Computing* 92.2 (2022), p. 35.

[313] S. Wang, X. Yu, and P. Perdikaris. "When and why PINNs fail to train: A Neural Tangent Kernel perspective". In: *arXiv preprint arXiv:2007.14527* (2020).

[314] Z. Wei, D. Liu, and X. Chen. "Dominant-current deep learning scheme for electrical impedance tomography". In: *IEEE Transactions on Biomedical Engineering* 66.9 (2019), pp. 2546–2555.

[315] G. Wen, C. Hay, and S. M. Benson. "CCSNet: A deep learning modeling suite for CO_2 storage". In: *Advances in Water Resources* 155 (2021), p. 104009.

[316] 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". In: *Advances in Water Resources* 163 (2022), p. 104180.

[317] A. Wexler, B. Fry, and N. M. "Impedance-computed tomography algorithm and system". In: *Appl. Opt.* 25 (1985), pp. 3985–92.

[318] I. H. Witten and E. Frank. "Data mining: practical machine learning tools and techniques with Java implementations". In: *Acm Sigmod Record* 31.1 (2002), pp. 76–77.

[319] S. Wold, K. Esbensen, and P. Geladi. "Principal component analysis". In: *Chemometrics and intelligent laboratory systems* 2.1-3 (1987), pp. 37–52.

[320] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo. "Sparse reconstruction by separable approximation". In: *IEEE Trans. Signal Process.* 57.7 (2009), pp. 2479–2493. URL: https://doi.org/10.1109/TSP.2009.2016892.

[321] Y. Wu, B. Chen, K. Liu, C. Zhu, H. Pan, J. Jia, H. Wu, and J. Yao. "Shape reconstruction with multiphase conductivity for electrical impedance tomography using improved convolutional neural network method". In: *IEEE Sensors J.* 21.7 (2021), pp. 9277–9287.

[322] K. Xu, D. Z. Huang, and E. Darve. "Learning constitutive relations using symmetric positive definite neural networks". In: *Journal of Computational Physics* 428 (2021), p. 110072.

[323] D. Yang, S. Li, Y. Zhao, B. Xu, and W. Tian. "An EIT image reconstruction method based on DenseNet with multi-scale convolution". In: *Math. Biosci. Eng.* 20.4 (2023), pp. 7633–7660.

[324] D. Yarotsky. "Error bounds for approximations with deep ReLU networks". In: *Neural Networks* 94 (2017), pp. 103–114.

[325] S. Yeonjong, J. Darbon, and G. E. Karniadakis. "On the Convergence of Physics Informed Neural Networks for Linear Second-Order Elliptic and Parabolic Type PDEs". In: *Communications in Computational Physics* 28.5 (2020), pp. 2042–2074.

[326] H. You, Y. Yu, M. D'Elia, T. Gao, and S. Silling. "Nonlocal kernel network (NKN): A stable and resolution-independent deep neural network". In: *arXiv preprint arXiv:2201.02217* (2022).

[327] B. Yu and W. E. "The Deep Ritz method: A deep learning-based numerical algorithm for solving variational problems". In: *arXiv preprint arXiv:1710.00211* (2017).

[328] J. Yu, L. Lu, X. Meng, and G. E. Karniadakis. "Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems". In: *Computer Methods in Applied Mechanics and Engineering* 393 (2022), p. 114823.

[329] Y. Zang, G. Bao, X. Ye, and H. Zhou. "Weak adversarial networks for high-dimensional partial differential equations". In: *Journal of Computational Physics* 411 (2020), p. 109409.

[330] D. Zhang, L. Lu, L. Guo, and G. E. Karniadakis. "Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems". In: *Journal of Computational Physics* 397 (2019), p. 108850.

[331] X. Zhang, Z. Wang, R. Fu, D. Wang, X. Chen, X. Guo, and H. Wang. "V-shaped dense denoising convolutional neural network for electrical impedance tomography". In: *IEEE Transactions on Instrumentation and Measurement* 71 (2022), pp. 1–14.

[332] D.-X. Zhou. "Universality of deep convolutional neural networks". In: *Applied and computational harmonic analysis* 48.2 (2020), pp. 787–794.

[333] Z. Zhou, G. S. dos Santos, T. Dowrick, J. Avery, Z. Sun, H. Xu, and D. S. Holder. "Comparison of total variation algorithms for electrical impedance tomography". In: *Physiol. Meas.* 36.6 (2015), pp. 1193–1209.

[334] Y. Zhu and N. Zabaras. "Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification". In: *Journal of Computational Physics* 366 (2018), pp. 415–447.