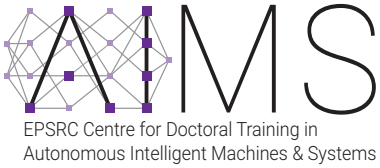
Physics-informed machine learning: from concepts to real-world applications



Ben Moseley

St Catherine's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2022

Acknowledgements

Firstly, I would like to thank Andrew Markham; asking Andrew to be my supervisor was the single best decision I made during my time at Oxford. He allowed me to roam freely in my research, yet gave me essential guidance exactly when I needed it, he challenged my thinking, yet appreciated where I was coming from, he taught me to trust my own instincts, and has made me feel like a valued member of the research community. We bonded immediately over our love of interdisciplinary research, and he gave me the confidence not just to follow the trend, but to focus on the problems which matter. I hope to receive Andrew's advice far into the future.

Secondly, I would like to thank my co-supervisor, Tarje Nissen-Meyer, for his guidance and encouragement. He has championed my research at every step along the way, and has provided critical context and understanding, particularly from the geophysical side.

I am grateful to have worked alongside some wonderful friends and collaborators. In particular, I want to thank Valentin Bickel for his truly impressive understanding of all things to do with the Moon, and Ada Alevizaki for her friendship as we completed our DPhil journeys together.

Several institutions have supported me through my study. I would like to thank the EPSRC Autonomous Intelligent Machines and Systems Centre for Doctoral Training (AIMS CDT) for funding me; I have been given invaluable opportunities through this program and I wholeheartedly recommend it to anyone interested. Secondly, parts of this theses were carried out in collaboration with NASA's Frontier Development Lab (FDL), and I would like to thank FDL for connecting me to their talented and diverse network of researchers, and providing me the chance to apply my skills to some of the biggest challenges out there.

Finally, I want to thank Jess, who has supported me through my ups and downs, and, at the right times, helped pull me away and enjoy the rest of life, and my family, to whom I dedicate this thesis to, for always being by my side and always wanting the best for me.

Abstract

Machine learning (ML) has caused a fundamental shift in how we practice science, with many now placing learning from data at the focal point of their research. As the complexity of the scientific problems we want to study increases, and the amount of data generated by today's scientific experiments grows, ML is helping to automate, accelerate and enhance traditional workflows.

Emerging at the forefront of this revolution is a field called scientific machine learning (SciML). The central goal of SciML is to more tightly combine existing scientific understanding with ML, generating powerful ML algorithms which are informed by our prior knowledge.

A plethora of approaches exist for incorporating scientific principles into ML and expectations are rising for SciML to address some of the biggest challenges in science. However, the field is burgeoning and many questions are still arising. A major one is whether SciML approaches can scale to more complex, real-world problems. Much SciML research is at a proof-of-concept stage, where techniques are validated on simplified, toy problems. Yet, understanding how well they scale to more complex problems is essential for them to become widely applicable.

This question is of central focus in this thesis. Firstly, multiple different physics-informed ML approaches are designed for three complex, real-world, domain-specific case studies taken from the fields of lunar science and geophysics, and their performance and scalability is assessed. Secondly, the scalability of physics-informed neural networks, a popular and general SciML approach, for solving differential equations with large domains and high frequency solutions is evaluated and improved. Common observations across these studies are discussed, and significant advantages and underlying limitations are identified, highlighting the importance of designing scalable SciML techniques.

Contents

List of Figures

List of Tables

List of Abbreviations

1D, 2D . . . . One- or two-dimensional.

ADC . . . . . Analogue-to-digital converter.

AI . . . . . Artificial intelligence.

AOI . . . . . Area of interest.

ASU . . . . . Arizona State University.

BNN . . . . . Bayesian neural network.

CCD . . . . . Charge-coupled device.

CFD . . . . . Computational fluid dynamics.

CMOS . . . . . Complementary metal–oxide–semiconductor.

CNN . . . . . Convolutional neural network.

CPU . . . . . Central processing unit.

CT . . . . . Computerised tomography.

DARTS . . . . Differentiable architecture search.

DLRE . . . . . Diviner Lunar Radiometer Experiment.

DN . . . . . Digital numbers.

EDR . . . . . Experiment data record.

ELBO . . . . . Evidence lower bound.

ELM . . . . . Extreme learning machine.

FBPINN . . . Finite basis physics-informed neural network.

FD . . . . . Finite difference.

FEM . . . . . Finite element method.

FFT . . . . . Fast Fourier transform.

FLOP . . . . . Floating point operation.

FNO . . . . . Fourier neural operator.

FOV . . . . . Field of view.

FWI . . . . . Full waveform inversion.

GAN . . . . . Generative adversarial neural network.

GP . . . . . Gaussian process.

GPU . . . . . Graphical processing unit.

GSFC . . . . . Goddard Space Flight Center.

HMC . . . . . Hamiltonian Monte Carlo.

HNN . . . . . Hamiltonian neural network.

HORUS . . . . Hyper-effective noise removal U-net software.

ISIS . . . . . Integrated Software for Imagers and Spectrometers.

KL . . . . . Kullback–Leibler.

LRO . . . . . Lunar Reconnaissance Orbiter.

LROC . . . . . Lunar Reconnaissance Orbiter Camera.

LSTM . . . . . Long short-term memory.

ML . . . . . Machine learning.

MRI . . . . . Magnetic resonance imaging.

NAC . . . . . Narrow Angle Camera.

NAS . . . . . Neural architecture search.

NETT . . . . . Network Tikhonov.

NMO . . . . . Normal move-out.

NN . . . . . Neural network.

ODE . . . . . Ordinary differential equation.

PDE . . . . . Partial differential equation.

PDS . . . . . Planetary Data System.

PIML . . . . . Physics-informed machine learning.

PINN . . . . . Physics-informed neural network.

PINO . . . . . Physics-informed neural operator.

PML . . . . . Perfectly matched layer.

PSNR . . . . . Peak signal-to-noise ratio.

PSR . . . . . Permanently shadowed region.

QCD . . . . . Quantum chromodynamics.

RDR . . . . . Reduced data record.

SEM . . . . . Spectral element method.

SNR . . . . . Signal-to-noise ratio.

SSIM . . . . . Structural similarity index measure.

TB . . . . . Terabyte.

TSR . . . . . Temporarily shadowed region.

VAE . . . . . Variational autoencoder.

VI . . . . . Variational inference.

VIPER . . . . Volatiles Investigating Polar Exploration Rover.

VPINN . . . . Variational physics-informed neural network.

WAC . . . . . Wide Angle Camera.

XPINN . . . . Extended physics-informed neural network.

1 Introduction

1.1 Overview

Machine learning (ML) has caused a revolution in the sciences. Traditionally, scientific research has revolved around theory and experiment: one proposes a hand-crafted and well-defined theory, then continuously refines it using experimental data and analyses it to make new predictions. But today, many are placing learning from data at the focal point of their research. Here, models of the world are learnt from data by ML algorithms and an existing theory is not required.

There are multiple reasons why this shift has occurred. Firstly, the field of ML has undergone exponential growth over the last decade, with the main driver behind this surge generally being attributed to breakthroughs in deep learning [Goodfellow et al., 2016]. Important discoveries such as the use of deeper network designs and better training algorithms, as well as the availability of more powerful computing architectures, have led to rapid improvements in the performance of deep learning techniques over a wide range of problems [Dally et al., 2021]. Modern ML algorithms are now able to learn about and solve incredibly complex tasks, from self-driving cars [Schwarting et al., 2018] to beating world-class Go players [Silver et al., 2018].

Alongside these advances, today's scientific experiments are generating increasingly large amounts of data and studying increasingly complex phenomena [Baker

et al., 2019, Hey et al., 2020]. It is quickly becoming impossible for humans, and our traditional workflows, to analyse and theorise about all of this data, and before long it is likely that scientific experiments will be limited by how well they can extract insights from the data they have rather than what data they can collect [Baker et al., 2019]. Given the powerful tools ML can offer, many researchers are turning towards ML to help automate, accelerate and enhance traditional workflows.

This combination of new ML algorithms and the availability of data has led to some significant scientific advances over the last decade. For example, ML has been used to predict protein structures more accurately than ever before [Jumper et al., 2021], synthesize speech from neural activity [Anumanchipalli et al., 2019] and improve the simulation of quantum many-body systems [Carleo and Troyer, 2017]. Indeed, modern ML algorithms have now been applied to nearly every facet of science and a defining research question of the era has become: "Take problem X and apply ML to it", with intriguing and often exciting results ensuing.

However, despite these advances, various shortcomings of ML and in particular deep learning algorithms have been crystallising in the field of ML. For example, whilst they are able to learn about highly complex phenomena, deep neural networks are generally regarded as "black-boxes" and there is a lack of understanding about how they represent and reason about the world. This uninterpretability is a key issue, particularly for safety-critical applications where a justification of the network's predictions is required [Gilpin et al., 2019, Castelvecchi, 2016]. Furthermore, there is little theoretical guidance on how to design deep learning algorithms which perform appropriately for a given task. The selection of deep neural network architectures is largely carried out empirically, although the fields of meta-learning and neural architecture search are starting to provide more automated approaches [Elsken et al., 2019, Hospedales et al., 2021]. Finally, although they are highly expressive, deep neural networks are limited by their training data and usually do not perform well outside of their training distributions. Learning generalisable models of the world which perform well on novel tasks is a crucial trait of more general artificial intelligence (AI) systems and a key outstanding challenge of the field of ML [Bengio et al., 2021].

Researchers are starting to encounter these limitations when using ML in scientific problems [Ourmazd, 2020, Forde and Paganini, 2019]. Given the poor generalisation ability of deep neural networks, a key concern is whether they truly "learn" scientific principles. A good scientific theory is expected to make novel and accurate predictions beyond experimental data, yet deep neural networks struggle to make accurate predictions outside of their training data. Even if a network could make reliable predictions, given their uninterpretability it may be challenging to extract any meaningful scientific insights from them. Another major concern is that many current ML workflows entirely replace traditional scientific models with learned models. Whilst this can be useful, these purely data-driven approaches "throw away" our vast prior scientific knowledge. The important point is that for many problems there is an existing theory which could be built upon, rather than starting from scratch. In a field traditionally based upon the tight interplay between well-defined theory and experiment, some believe that the limitations above render current ML approaches unacceptable.

These concerns have prompted the formation of a new and rapidly growing field, known as *scientific machine learning* (SciML) [Baker et al., 2019, Karniadakis et al., 2021, Willard et al., 2020, Cuomo et al., 2022, Arridge et al., 2019, Karpatne et al., 2017a]. The goal of SciML is to blend together existing scientific knowledge and ML, generating more nuanced ML algorithms which are informed by our prior knowledge, as shown in Figure 1.1. The key thesis of the field is that by doing so we will end up with more powerful methods for scientific research. Both traditional and ML methods have advantages and disadvantages, and their combination may prove more powerful than one or the other. For example, when carrying out data assimilation (e.g. in climate modelling), traditional physical modelling could be used to provide prior knowledge whilst ML could be used to account for data-dependencies and other unknown physics.

Expectations of the field are rapidly growing and a wide plethora of approaches with many innovative strategies for incorporating scientific knowledge into ML is currently being proposed and investigated. These approaches range from their

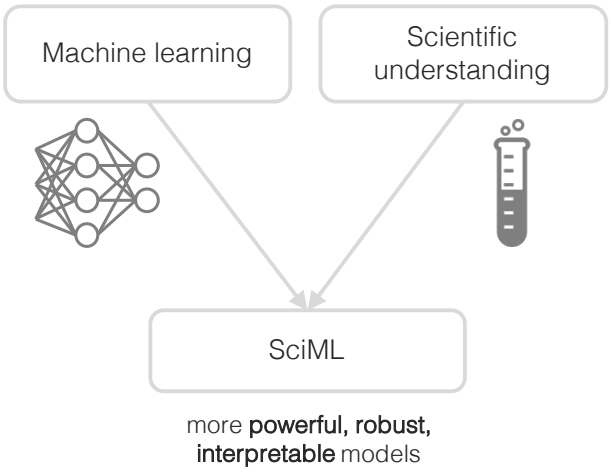


Figure 1.1: Scientific machine learning (SciML) overview. SciML aims to tightly combine ML with scientific knowledge in order to generate more powerful, robust and interpretable ML methods for scientific research.

intended scientific task (e.g. simulation, inversion, and governing equation discovery), to different ways to incorporate scientific principles (e.g. through the architecture of a deep neural network, its loss function, and the use of hybrid models), and the degree to which scientific principles are imposed (e.g. via hard or soft constraints). We present a detailed review of these approaches in Chapter 2. Many approaches use ideas from physics to inform their ML algorithms in a sub-field of SciML known as physics-informed machine learning (PIML) [Karniadakis et al., 2021].

So far, SciML is enjoying some early successes. It has helped us carry out powerful simulations [Raissi et al., 2019], discover the governing equations of complex physical systems [Kutz and Brunton, 2022], accurately invert for underlying parameters in inversion problems [Arridge et al., 2019], and seamlessly blend traditional workflows with learned components [Rackauckas et al., 2020, Thuerey et al., 2021] across a wide range of domains.

Despite its early promise, the field of SciML is still in its infancy and there are many important questions arising, for example; how should we enforce scientific principles? How should we balance the lack of interpretability of data-driven models with the clarity of existing theories? Are there any overarching SciML techniques which can be applied across scientific disciplines? Can SciML provide new perspectives and

ideas for the field of ML? How well do SciML techniques scale to complex, real-world problems? This thesis focuses on the last question, which is discussed in detail below.

1.2 Research questions

Much of the current research in the field of SciML is focused on developing new concepts and techniques for combining scientific principles with ML. This is an exciting phase of the field, and it is largely being driven by modern advances in ML.

Given that SciML is an emerging field, many of these techniques are not yet fully explored or understood. In particular, most works either demonstrate a proof-of-principle of a new technique, or its application in a specific domain. In both cases, a simplified or toy problem is typically used to test the effectiveness of the technique. This is usually because these problems are easy to understand and implement, and they often have analytical or well-studied solutions which makes it easy to assess how the technique is performing.

An essential next step is to shift focus towards solving more complex, real-world problems with SciML. Major challenges are usually encountered when scaling traditional methods to such problems and it is therefore crucial to understand how well SciML approaches scale in this setting. This is important because we ultimately want to use SciML to solve real-world problems. Indeed, the complexity of the scientific problems we are interested in studying today is increasing, and so being able to answer this question will only become more important over time.

By real-world problems, we mean problems which attempt to closely resemble the real-world and thus have tangible impact. Examples would be the simulation of the aerodynamics of an aeroplane to optimise its design, or the estimation of the location of an earthquake from real seismic recordings to better understand the Earth's crustal dynamics. In contrast, well-studied toy versions of these problems would be the simulation of the fluid flow over a cylinder, or the estimation of the location of a point source in a small, simplified medium.

Below is a list of some of the major challenges often encountered when scaling traditional scientific analysis methods from toy problems to real-world problems.

This list is not intended to be complete or exhaustive, but highlights some broad challenges and specific examples. These challenges have been the subject of decades of research and remain difficult to solve;

Incorporating more complex phenomena The real world usually contains much more complex physical phenomena than those studied in toy problems. Incorporating more complex physical phenomena is highly non-trivial and usually requires significant changes to the design of traditional algorithms. For example, when carrying out simulation, incorporating multi-scale, multi-physics phenomena can be very challenging. Finite difference and finite element methods usually require elaborate changes such as adaptive mesh refinement and sub-grid parameterisations to accurately simulate such systems. Furthermore, for many real-world problems it can be challenging to even know what the true underlying physical processes are. In such cases techniques such as data assimilation are required to ensure our physical models are well-calibrated with reality.

Computational intractability As the size and complexity of the problem grows, the computational requirements of traditional algorithms often become extremely demanding. For example, when simulating high frequency phenomena (or similarly, phenomena over long distance/time scales) the number of discrete elements (and hence computations) required by finite difference or finite element methods typically grows exponentially with the dimensionality of the domain. Thus, supercomputers with thousands of nodes are required to carry out such large-scale simulations. When solving inversion problems, more complex physical models can significantly increase the difficulty of the problem. In particular, much larger numbers of underlying parameters may need to be estimated, any forward simulations required may become much more computationally demanding, and the more complex model may make the problem significantly more ill-posed, i.e there exists many possible underlying parameter configurations for a given set of observations of the system. This often results in the problem being computationally intractable.

Real-world data Real-world data nearly always includes unwanted processes. These processes can be deterministic or random, and are usually labelled as noise. Noise can be very challenging to identify and disentangle from the underlying processes being studied. For example, noise often strongly affects inversion problems, by increasing the problem's ill-posedness and making it harder to solve. Often, a fully probabilistic framework is required to ensure an inversion algorithm is robust to noise. Another example is in the discovery of underlying laws. Here, alongside discovering underlying processes, algorithms must also be able to disentangle noise processes.

It is clear that transitioning from simplified problems to real-world problems is highly non-trivial for traditional methods, but less research has focused on assessing how well SciML, and in particular PIML, concepts and techniques scale to these problems. This issue is the central subject of this thesis. Our main research question is:

How well do PIML techniques scale to real-world problems?

For the purpose of this thesis the notion of scalability is defined in the broadest sense; any factor which could affect the performance of an algorithm when it is applied to a real-world problem is considered.

Many more sub-questions stem from this central question. For example, how do the challenges encountered when scaling PIML algorithms compare to those encountered when scaling traditional methods? What changes need to be made to PIML algorithms to help them scale better? Furthermore, given the plethora of different PIML concepts and techniques being researched, it is clear that the answers to these questions will vary significantly across different approaches. In this thesis we focus on a few specific sub-questions which are listed below.

Can PIML algorithms discover underlying physical principles from real data?

One goal of PIML is to provide more powerful algorithms which can discover underlying physical laws and principles from data. However, many of the current approaches are trained and tested using synthetic data generated from simplified

physical models. Furthermore, many only assume simple (e.g. Gaussian) noise models. Real data often contains significantly more complex physical and noise processes. Key questions are, how is the performance of these algorithms affected by real noise? How well do they scale to more complex physical systems?

Can PIML algorithms carry out inversion using real data?

A large variety of PIML concepts have been proposed to solve inversion problems. Yet, similar to above, many are only trained and tested using synthetic data generated from toy physical models. Typically, the difficulty of the inverse problem increases significantly for more complex systems and when using real data. Key questions are, how well do PIML algorithms cope with real-world noise? Can they carry out inversion for more complex physical systems? How do their computational requirements scale?

Can PIML algorithms simulate complex physical phenomena?

Another active area is the use of PIML for enhancing simulation. However, many PIML approaches are only verified by simulating simple physical systems with well-studied solutions. Usually, elaborate changes to traditional simulation methods are required to model more complex phenomena. Key questions here are, how well do PIML techniques scale to multi-scale, multi-physics phenomena? What changes should be made to improve their performance?

Can PIML algorithms carry out large-scale simulations?

Related to the above question, many PIML simulation algorithms are only tested using problems confined to small domains with low frequency solutions. Yet, simulating larger domains with traditional methods (i.e. higher frequencies, over longer distance/time scales) often becomes computationally intractable. Key questions are, how do the computational requirements of PIML techniques scale? How does this compare to traditional methods? How should the design of PIML algorithms change so they are scalable?

1.3 Thesis overview

1.3.1 Overview

In this thesis, we use two main approaches for studying the sub-questions above. Firstly, for the first three sub-questions, we use complex, real-world, domain-specific case studies to investigate the performance and scalability of multiple different PIML approaches. For each sub-question we present a case study, propose a PIML technique (or variety of PIML techniques) for solving it, and evaluate how well the technique scales to this setting. Secondly, for the last sub-question, we focus on a single general-purpose PIML technique and assess and improve its scalability.

Each of the first three sub-questions is studied in a separate chapter in this thesis (Chapters 3-5 respectively) and their case studies are taken from the fields of lunar science and geophysics. The last sub-question is investigated in Chapter 6. Finally, we discuss and summarise the implications of each chapter on our main research question in Chapter 7.

1.3.2 Detailed structure

In this section we give a more detailed overview of this thesis. First, Chapter 2 discusses the existing literature from the field of SciML. We review general strategies for incorporating scientific principles into ML, with a focus on deep learning-based PIML techniques, and then discuss general trends, challenges and future opportunities for the field.

Next, Chapter 3 considers the question, can PIML algorithms discover underlying physical principles from real data? More specifically, we train a variational autoencoder (VAE) to disentangle underlying factors of variation from real measurements of the lunar surface temperature taken by the Diviner instrument on-board the Lunar Reconnaissance Orbiter (LRO) satellite. First, a Gaussian process is used to model the noise in this dataset and interpolate its raw temperature measurements. Then, the VAE is trained using example temperature profiles extracted from multiple locations over the entire lunar surface. Its latent space is interpreted by comparing

it to parameters estimated from inversion using an existing thermophysical model. Global maps of the VAE latent variables are generated and used for anomaly detection on the lunar surface. We discuss the advantages of using the VAE alongside traditional thermophysical modelling and future ways to combine physical principles into the VAE.

Chapter 4 investigates the question, can PIML algorithms carry out inversion using real data? In particular, we present a PIML approach for enhancing extremely low-light images of permanently shadowed regions on the Moon. The underlying inversion problem is to estimate the mean photon count from real, noisy images captured by the LRO Narrow Angle Camera (NAC). Our physics-informed workflow uses two deep neural networks. Given an input noisy image, the first network estimates and removes the dark noise contained in the image, using the environmental meta-data (such as CCD temperature and orbit number) available at the time of image capture. Next, the flatfield and nonlinearity corrections taken from the existing calibration routine of the camera are applied. Finally, the second network estimates and removes all other noise sources, such as photon (shot) noise and image compression noise. We use a realistic physical noise model of the camera which combines both synthetic noise and real noise samples to generate training data. We compare our PIML approach to a naive end-to-end deep learning method, as well as the existing calibration routine of the camera.

Chapter 5 considers the question, can PIML algorithms simulate complex physical phenomena? More specifically, we propose, evaluate and discuss three different approaches for simulating increasingly complex seismic wave phenomena. First, we use a deep neural network with a physics-informed WaveNet architecture [van den Oord et al., 2016] to simulate the seismic waves generated by a point source propagating in simple 2D horizontally layered acoustic media. Next, we use a deep neural network with a conditional encoder-decoder architecture to simulate the same problem, but for more realistic media which contain geological faults and for a variable source location. Lastly, we use a fully-connected neural network with a physics-informed loss function to simulate the entire seismic wavefield in both simple 2D horizontally layered media

and strongly heterogeneous Earth-realistic media. We discuss in detail the remaining challenges of scaling our approaches to more realistic, complex, 3D media.

Chapter 6 researches the question, can PIML algorithms carry out large-scale simulations? Specifically, we investigate the scalability of physics-informed neural networks (PINNs) [Raissi et al., 2019, Lagaris et al., 1998] when solving differential equations with large domains. First, we assess the performance of PINNs when solving problems with high frequency solutions. Next, we present a general approach for scaling PINNs to large domains called finite basis physics-informed neural networks (FBPINNs). FBPINNs extend PINNs by combining them with domain decomposition, individual subdomain normalisation and flexible training schedules. The performance of FBPINNs and PINNs is compared across different simulation problems and their outstanding scalability challenges are discussed.

Finally, Chapter 7 concludes this thesis by discussing how each chapter contributes to our central research question. Common observations and challenges are identified and several areas of future research are recommended.

1.4 Contributions

In this section the contributions of each chapter are summarised. A more detailed discussion of their implications on our main research question is presented in Chapter 7.

For investigating whether PIML algorithms can discover underlying physical principles from real data (Chapter 3):

- We show that a VAE is able to disentangle physically meaningful factors of variation (such as effective albedo, thermal conductivity and solar onset) from real measurements of the lunar surface temperature.

- The VAE is able to invert for underlying physical parameters multiple orders of magnitude faster than traditional inversion using an existing physical model, allowing anomaly maps of the entire lunar surface to be generated.

- We show that carrying out traditional physical modelling alongside the VAE is synergistic; physical modelling allows us to interpret the latent space of the VAE, and the agreement of the VAE increases our confidence in the existing physical model.

- However, the complexity of the real-world dataset makes it challenging to fully disentangle some of the physical processes present and interpret parts of the model learned by the VAE and we discuss this further.

For investigating whether PIML algorithms can carry out inversion using real data (Chapter 4):

- We present a novel PIML algorithm for denoising real low-light images of permanently shadowed regions (PSRs) on the lunar surface.

- We show that our algorithm is able to accurately denoise these images, strongly outperforming the existing calibration routine of the camera and a naive end-to-end ML baseline, allowing us to reveal surface features in PSRs down to 3-5 m for the first time.

- However, we find the real-world noise contained in these images makes it challenging to validate the denoised images, generate realistic training data and ensure the algorithm generalises well to real images, and we discuss these limitations further.

For understanding whether PIML algorithms can simulate complex physical phenomena (Chapter 5):

- We present and evaluate various physics-informed approaches for simulating increasingly complex seismic wave phenomena. This includes a deep neural network with a WaveNet architecture, a deep neural network with a conditional encoder-decoder design and a fully-connected neural network with a physics-informed loss function.

- The first approach is able accurately to simulate seismic waves in horizontally layered media, the second in faulted media and the third in strongly heterogeneous Earth-realistic media.
- Once trained, all approaches are orders of magnitude faster than finite difference simulation.
- We encounter multiple challenges when scaling our approaches to more complex simulations, including the limited generalisation ability of the networks and the increasing computational resources required to train them, and we discuss these challenges further.

For understanding whether PIML algorithms can carry out large-scale simulations (Chapter 6):

- We show that PINNs struggle to solve differential equations with high frequency solutions and large domains, due to the increased complexity of their underlying optimisation problem.
- We present a general physics-informed deep learning framework which is able to alleviate this issue and scale PINNs to solve large, multi-scale problems relating to differential equations.
- Our framework outperforms PINNs in terms of accuracy and computational resources required.
- Even so, we find outstanding challenges of our approach compared to traditional simulation methods and discuss these further.

1.5 Publications

Below is the list of publications which have been generated as a direct result of the work in this thesis:

Peer-reviewed

1. **Moseley, B.**, Bickel, V., Lopez-Francos, I. G., & Rana, L. (2021). Extreme Low-Light Environment-Driven Image Denoising Over Permanently Shadowed Lunar Regions With a Physical Noise Model. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

2. Bickel, V., **Moseley, B.**; Lopez-Francos, I., & Shirley, M. (2021). Revealing the Geomorphology and Trafficability of Lunar Permanently Shadowed Regions with Deep Learning. Nature Communications.

3. **Moseley, B.**, Nissen-Meyer, T., & Markham, A. (2020). Deep learning for fast simulation of seismic waves in complex media. Solid Earth, 11(4), 1527–1549.

4. **Moseley, B.**, Bickel, V., Burelbach, J., & Relatores, N. (2020). Unsupervised Learning for Thermophysical Analysis on the Lunar Surface. The Planetary Science Journal, 1(2), 32.

Preprints

5. **Moseley, B.**, Markham, A., & Nissen-Meyer, T. (2021). Finite Basis Physics-Informed Neural Networks (FBPINNs): a scalable domain decomposition approach for solving differential equations. ArXiv.

6. **Moseley, B.**, Markham, A., & Nissen-Meyer, T. (2020). Solving the wave equation with physics-informed deep learning. ArXiv.

7. **Moseley, B.**, Markham, A., & Nissen-Meyer, T. (2018). Fast approximate simulation of seismic waves with deep learning. ArXiv.

Publication 4 forms the basis of Chapter 3, publications 1 and 2 form the basis of Chapter 4, publications 3, 6 and 7 form the basis of Chapter 5 and publication 5 forms the basis of Chapter 6.

In addition, much of the code used in this thesis is available open-source and can be found here: `https://github.com/benmoseley`.

2 Background

2.1 Scientific machine learning overview

The field of scientific machine learning (SciML) has been rapidly growing over the last few years. Seminal papers have attracted thousands of citations (e.g. Raissi et al. [2019], Chen et al. [2018a]), several SciML workshops have been hosted at leading AI conferences [NeurIPS, 2021a,b, ICLR, 2020, AAAI, 2021, 2020, ICERM, 2019], and significant amounts of funding have been announced[1]. SciML research is being carried out across a diverse range of disciplines and scientific objectives, for example in climate science [Rasp et al., 2018], fluid dynamics [Jin et al., 2021], biology [Jumper et al., 2021] and chemistry [Zhang et al., 2018], and expectations are rising for SciML to accelerate scientific discovery and address humanity's biggest challenges [Baker et al., 2019, Irrgang et al., 2021, Lavin et al., 2021].

In this chapter we aim to provide a snapshot of the latest SciML research. The field is already vast, and so our goal is to provide a representative overview rather than an exhaustive review. It is also fast-moving, and we aim to capture this by including some of the techniques which have gained recent popularity. There are many other reviews related to this field (e.g. Willard et al. [2020], Karniadakis

[1]For example, the US National Science Foundation recently founded a $20M Institute for Artificial Intelligence and Fundamental Interactions which aims to develop novel AI approaches that incorporate first principles from fundamental physics [NSF, 2020].

et al. [2021], Arridge et al. [2019], Karpatne et al. [2017a], Rai and Sahu [2020], Alber et al. [2019], Kutz and Brunton [2022]) from which we draw from. Most of our focus is placed on deep learning-based PIML research, which is driving a large portion of current SciML research.

Such a review could be approached from many different angles. One way is to focus on different classes of scientific tasks (e.g. simulation, inversion, equation discovery, etc) across different disciplines and describe how SciML can aid them. Another is to focus on the different ways scientific principles can be incorporated into ML algorithms (e.g. through architecture design, loss functions, hybrid approaches, etc) and describe example scientific applications. Here we choose the latter approach, centering around the SciML algorithms themselves.

A wide plethora of SciML approaches exist and Table 2.1 summarises the approaches covered in this review. The table classifies approaches by the way scientific knowledge is incorporated into their ML algorithm and their intended scientific task. We see that SciML research covers this entire plane, suggesting that SciML algorithms can be designed in many different ways for many different tasks. In addition, Figure 2.1 plots how "strongly" scientific knowledge is imposed for the approaches studied. We find that the whole spectrum is spanned, suggesting that there is no single best way to impose scientific constraints.

The remainder of this chapter is structured as follows. First, in Section 2.2 we introduce the typical scientific tasks for which SciML algorithms are being developed. Then, in Section 2.3 we review different SciML approaches, grouped by the way scientific knowledge is incorporated into their ML algorithm. Finally, in Section 2.4 we summarise our findings and identify general trends, challenges and future opportunities for the field, focusing on our central research question (how well do PIML techniques scale to real-world problems?).

2.2 Scientific tasks

In order to aid the discussion on SciML approaches we first provide a high-level overview of the types of scientific tasks that they are being designed for and the benefit

	Forward simulation	Inversion	Equation discovery
Architecture			
Adding physical variables	Daw et al.		
Encoding symmetries	Ling et al., Wang et al., Anderson et al., Schütt et al.		Udrescu et al.
Physics-inspired NAS	Ba et al., Panju and Ghodsi		
ML inspired by Koopman theory	Geneva and Zabaras, Lusch et al.		
Physically constrained GPs		Raissi et al., Raissi and Karniadakis	
Other approaches	Jumper et al., Mohan et al.		
Loss function			
Constraining physical quantities	Karpatne et al., Zhang et al., Benjamin Erichson et al., Xie et al., Brehmer et al.		
Encoding conservation laws	Beucler et al., Zeng et al.		Greydanus et al., Toth et al., Cranmer et al.
Auxiliary tasks	de Oliveira et al.		
Encoding governing equations	Raissi et al., Jin et al., Kharazmi et al., Yang et al., Wang et al., Wang et al., Li et al., Zhu et al., Geneva and Zabaras, Gao et al.	Jin et al., Chen et al., Yang et al.	Chen et al., Champion et al.
Hybrid approaches			
Residual modelling	Pawar et al.	Jiang et al.	
Differentiable physics		Ren et al., Minkov et al., Würfl et al., Zhang et al.	
Neural differential equations			Chen et al., Rackauckas et al., Long et al.
In-the-loop methods	Um et al., Rasp et al.	Adler and Öktem, Morningstar et al., Hammernik et al., Li et al., Lunz et al., Bora et al., Mosser et al.	

Table 2.1: SciML works reviewed in this chapter, classified by the way scientific knowledge is incorporated into their ML algorithm (rows) and their intended scientific task (columns). This table and taxonomy is inspired by Willard et al. [2020].

SciML can bring. These are fundamental and essential tasks carried out across science, and they traditionally have long-standing general and domain-specific challenges associated with them. Visual examples of these tasks are shown in Figure 2.2.

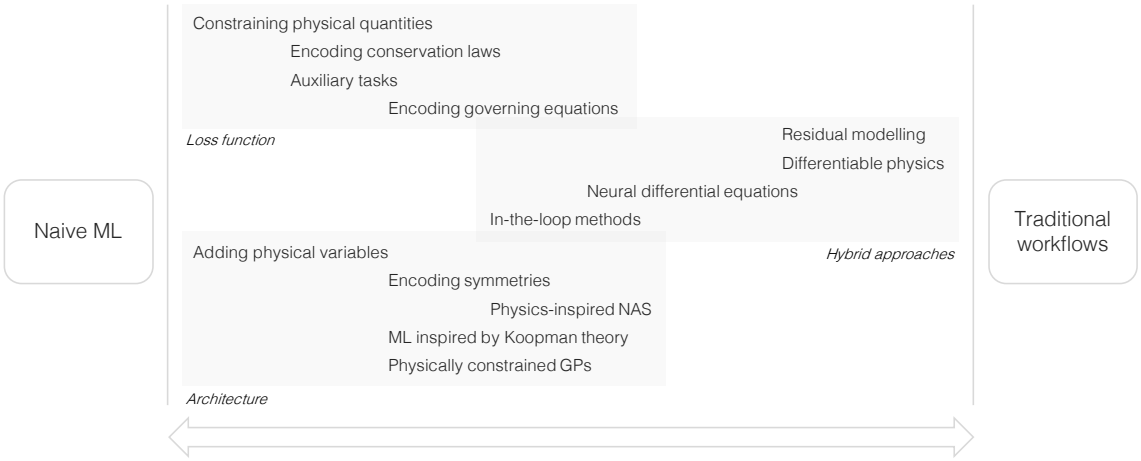


Figure 2.1: Spectrum of SciML approaches. This figure plots how "strongly" scientific knowledge is imposed for the different types of SciML approaches reviewed in this chapter. Note the strength of a scientific constraint is a fairly fuzzy concept; in this plot we define it as how close the SciML approach is to a traditional workflow. An approach in the middle would equally combine ML with some aspects of traditional workflows, for example in-the-loop methods which interleave traditional iterative solvers with ML models. Furthermore our assignment is somewhat subjective, and so this figure is only intended to convey the general trend.

2.2.1 Forward simulation

One major task is carrying out forward simulation. That is, predicting certain properties of a system given some input conditions of the system. At a very high level, one could write

$$b = \mathcal{F}(a) \ , \tag{2.1}$$

where a is a set of input conditions of the system, $\mathcal{F}$ defines a physical model of the system, and b is a set of resulting properties of the system given $\mathcal{F}$ and a. We note, depending on the specific problem, a and b could contain scalars, vectors, tensors, functions, or other descriptions of the system.

For many tasks, forward simulation amounts to solving a set of differential equations describing the system. In many cases, the system can be described by

$$\begin{aligned} \mathcal{D}[u(x);\lambda] &= f(x) \ , \quad x \in \Omega \ , \\ \mathcal{B}_k[u(x)] &= g_k(x) \ , \quad x \in \Gamma_k \subset \partial\Omega \ , \end{aligned} \tag{2.2}$$

for $k = 1, 2, ..., n_b$ where $\mathcal{D}$ is a differential operator, $\mathcal{B}_k$ is a set of boundary operators, $u \in \mathbb{R}^{d_u}$ is the solution to the differential equation, $f(x)$ is a forcing

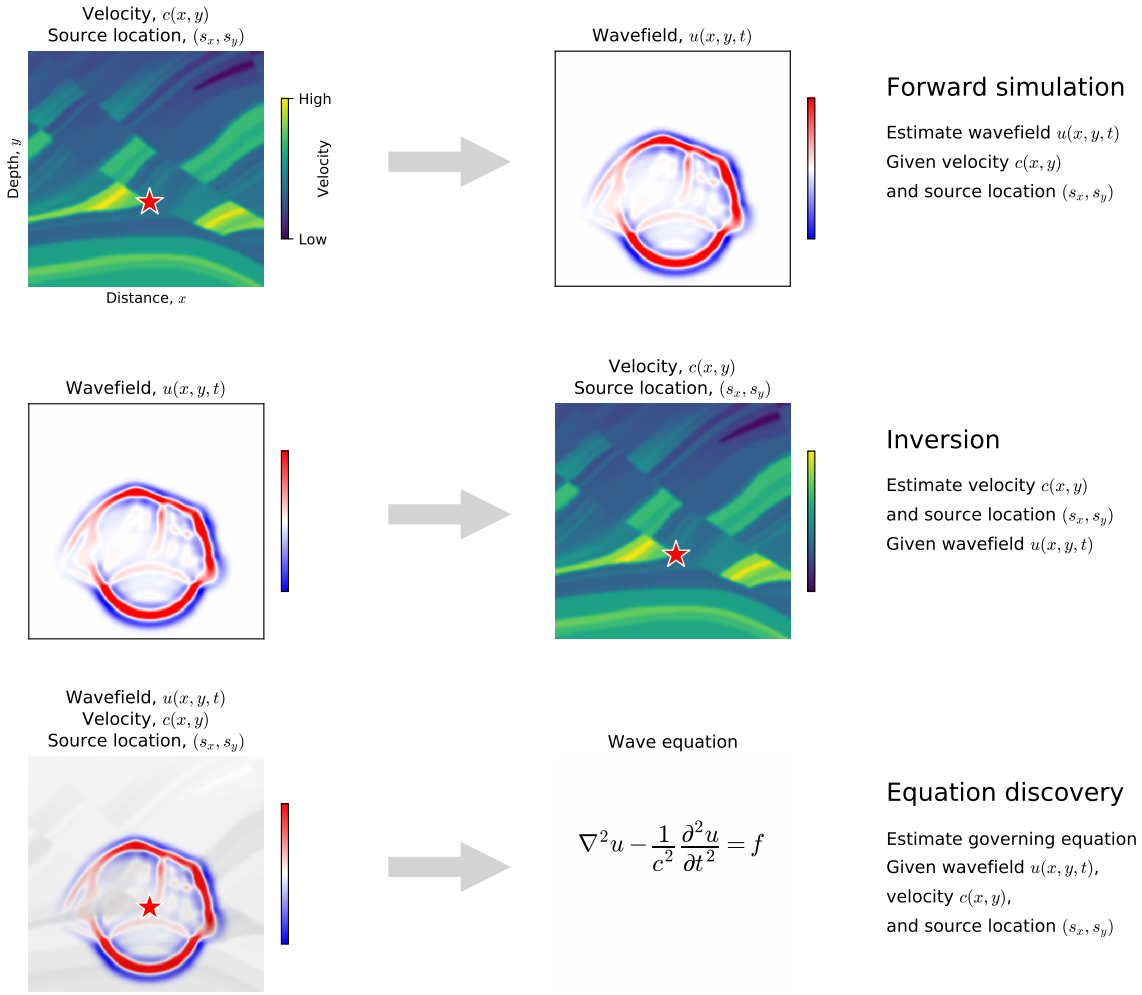


Figure 2.2: Types of scientific tasks SciML approaches are being designed for. Whilst their application domains can differ, the goal of many SciML approaches can be abstracted into one of three tasks, namely forward simulation, inversion and equation discovery. In this plot we visualise each of these tasks using the example of seismic waves propagating through the Earth as a result of an earthquake.

function, $g_k(x)$ is a set of boundary functions, x is an input vector in the domain $\Omega \subset \mathbb{R}^d$ (i.e. x is a d-dimensional vector), $\partial\Omega$ denotes the boundary of Ω and λ is a set of additional parameters of the differential operator. Many different differential equations can be represented in this form, including those with time-dependence or time-independence, and linear or nonlinear operators. In this particular setting, $a = \{f(x), g_1(x), ..., g_{n_b}(x), \lambda\}$, $b = \{u(x)\}$ and $\mathcal{F}$ represents some method for solving the equations.

There are many long-standing challenges associated with carrying out simulation. For most real-world applications, the greatest challenge is usually the extremely high

computational costs of simulating complex, multi-scale, multi-physics systems. Such simulations can require elaborate implementations and millions of CPU hours on the world's fastest supercomputers. Many general and domain-specific strategies exist for reducing computational cost, including adaptive mesh refinement, subgrid parameterisations, and reduced-order modelling. However, there is usually a compromise required between the level of approximation of the physical system and the computational cost of the simulator.

As we will see below, SciML is helping to alleviate these issues by allowing the possibility to learn from previous simulations, providing more powerful computational shortcuts whilst having less impact on the simulation fidelity. For example, SciML is being used to learn more efficient subgrid parameterisations [Rasp et al., 2018, Rackauckas et al., 2020], finer-resolution outputs from coarser-resolution simulations [Jiang et al., 2020, Um et al., 2020], and mesh-free methods which do not require elaborate discretisation schemes [Lagaris et al., 1998, Raissi et al., 2019].

2.2.2 Inversion

Inverse problems are tightly related to simulation and solving them is crucial for many real-world tasks. Here the goal is to estimate a set of latent, or unobserved parameters of a system given a set of real-world observations of the system. Under the framework described by Equation 2.1, it is the task of estimating a given b.

Inversion can be extremely challenging for a number of reasons. Firstly, inversion algorithms often require many forward simulations to be run in order to match the predictions of the physical model to the set of observations. Given the potentially high computational costs of forward simulation stated above, this can render many applications infeasible. Secondly, inversion problems often suffer from ill-posedness. This is the case where the information provided by b is not sufficient to uniquely determine a. In such cases, sophisticated regularisation schemes are required to restrict the space of possible latent parameters the inversion algorithm can explore. Finally, real-world inversion usually suffers from noise. This can be challenging

to identify, and usually increases the ill-posedness of the problem. Often a fully probabilistic framework is required to model such processes.

SciML is aiding inversion algorithms by addressing some of the challenges above. For example, SciML is providing fast surrogate models which allow many forward simulations to be carried out inexpensively [Wang et al., 2021b, Li et al., 2021, Zhu et al., 2019], stronger regularisers which learn from previous examples [Hammernik et al., 2018, Li et al., 2020a, Bora et al., 2017], and more tractable methods for estimating posterior distributions of unknown parameters in probabilistic settings [Brehmer et al., 2020].

2.2.3 Equation discovery

Finally, there are many cases where we do not fully understand the system itself. That is, we are unsure how to define our physical model $\mathcal{F}$ in Equation 2.1. For example, we still do not have a satisfactory model of fundamental particles, or of many processes in the Earth's climate system. Being able to learn about a system, for example by discovering its governing equations, is powerful as it can provide a general model of the system.

One could think of this as a different type of inversion problem where the entire physical model, or at least the unknown parts of it, are inverted for given some observations of the system. This is often very challenging, and inherits many of the difficulties encountered by inversion tasks described above.

Traditionally, discovering new theories about the world has relied upon remarkable human intuition. SciML is aiding this discovery by allowing us to automate the process and/or learn about complex processes which are hard to intuit [Chen et al., 2021, Champion et al., 2019, Long et al., 2019, Udrescu et al., 2020]. Furthermore, it is allowing us to cope with the increasingly large amounts of scientific data being generated by modern experiments, which is quickly becoming impossible for humans to sift through.

2.3 Ways to incorporate scientific principles into machine learning

In this section we review a snapshot of the current SciML approaches being researched. We primarily focus on the way scientific knowledge is incorporated into their ML algorithm, and the sections below are grouped by different high-level ways to do so; namely through the ML architecture, loss function and the use of hybrid approaches.

2.3.1 Architecture

The first set of SciML approaches considered are those which change the architecture of the ML algorithm so that it incorporates scientific constraints. Rather than treating a ML algorithm as a "black-box", we open up its design and change parts of it so that it obeys these constraints. For deep learning-based techniques, this usually means making changes to the neural network architecture itself. Incorporating scientific principles in this way can restrict the range of models the algorithm can learn, and result in more generalisable and interpretable models. From a machine learning perspective, we are introducing a strong inductive bias into the model.

2.3.1.1 Adding physical variables

One way to do this when using neural networks is to assign physical meaning to some of its neurons. For example, Daw et al. [2020] used a sequence of long short-term memory (LSTM) networks and a fully connected layer to model the time evolution of lake temperatures. Importantly, the LSTMs were used to predict an intermediate physical quantity, the density profile of the lake, before passing this to the fully connected layer to estimate its temperature. They also used a monotonicity-preserving LSTM design to preserve the fact that the density profile only increases with depth. All networks were trained end-to-end. They found both techniques improved the generalisation and physical consistency of the temperature predictions compared to a black-box end-to-end LSTM network.

2.3.1.2 Encoding symmetries

Another approach is to encode symmetries into the ML model. Indeed, symmetries such as translational and rotational invariance play a fundamental role in physics and dictate the form of the laws of nature [Gross, 1996]. And thus, it is likely that honouring symmetries in ML models will make them more physically consistent and generalisable.

For example, Ling et al. [2016] proposed rotationally invariant tensor basis neural networks for estimating the Reynolds stress anisotropy tensor in turbulent flows. They ensured the neural network's prediction stayed the same when the input axes of the flow were rotated, by noting that the anisotropy tensor lies on a basis of isotropic tensors and by using the neural network to predict the coefficients of the tensor in this basis. They found that this approach provided more accurate flow predictions than a generic neural network architecture.

Wang et al. [2021a] proposed a variety of methods for enforcing different types of symmetries (translation, rotation, uniform motion and scale) in convolutional neural networks (CNNs) when modelling dynamical systems. They showed that CNNs which included these constraints were more robust to different symmetry transformations and were better at conserving quantities such as energy than standard CNNs when used to solve the Navier-Stokes equations.

In molecular dynamics, Anderson et al. [2019] proposed a rotationally covariant neural network architecture for predicting the ground state properties and potential energy surfaces of molecules. Rotational and translational covariance were explicitly encoded by representing all activations in spherical tensor form. Similarly, Schütt et al. [2017] used neural networks to predict the potential energy surfaces of molecules, ensuring rotational invariance by using interatomic distances as inputs to their network. Furthermore, they proposed continuous convolutions that allowed these inputs to have arbitrary positions.

Finally, Udrescu et al. [2020] presented AI Feynman, a search algorithm for symbolic regression of physical functions which relied on "hidden simplicity" to factor the exponential search space of possible functions into multiple smaller search

spaces. Their algorithm worked by first fitting a neural network to observed data and recursively testing this mystery function for different types of modularity, including compositionality, symmetry and separability. Given this modular decomposition, a combination of polynominal fitting and brute-force search was used to find the remaining functional form. By doing so, their method was able to significantly outperform a state-of-the-art genetic algorithm [Udrescu and Tegmark, 2020].

2.3.1.3 Physics-inspired neural architecture search

Taking a different approach, Ba et al. [2019] explored the idea of physics-informed neural architecture search (NAS). Specifically, they considered the case of learning to model a physical system given training examples of its inputs and outputs. They used a neural network to model the system and whilst training the network they used a modified version of differentiable architecture search (DARTS) [Liu et al., 2018]. Importantly, the set of operators which DARTS selected from to design the network contained physically-motivated operators alongside standard neural network layers, including an incomplete (and differentiable) physical model of the system. Interestingly, they found the degree to which the physical operators were used depended on their completeness and the quality of training data available, and their approach outperformed standard residual modelling with a neural network. In a similar vein, Panju and Ghodsi [2020] used neural networks to learn symbolic solutions of differential equations. Each layer of the network consisted of a set of mathematical operators followed by a learnable softmax gate. Thus, during training the network learned to select a specific sequence of operators from which the symbolic solution could be interpreted. The network was trained by minimising the differential equation residual of its predicted solution over a set of random points in the domain. They found their method was able to provide approximate symbolic solutions to problems which have no simple analytical solutions.

2.3.1.4 ML inspired by Koopman theory

Another line of work is to constrain ML algorithms using concepts from Koopman operator theory [Brunton et al., 2021]. Koopman theory states that any (potentially

nonlinear) dynamical system can be represented in terms of an infinite-dimensional linear operator [Koopman, 1931]. This representation is powerful as it allows us to use linear analysis methods to efficiently predict, estimate and control a system, but finding such a representation is challenging. Encoding such linear constraints in a ML model could improve its performance [Kutz and Brunton, 2022].

For example, Geneva and Zabaras [2022] proposed a general transformer model [Vaswani et al., 2017] for predicting the temporal evolution of a dynamical system, where the input to the transformer was a learned embedding of the state which was constrained by Koopman dynamics. The embedding was defined as the latent vector of an encoder-decoder model, and it was constrained by pretraining the encoder-decoder to both reconstruct snapshots of the current state and to predict future timesteps by multiplying the embedding with a learnable Koopman matrix before decoding it. They found that such a representation improved the extrapolation accuracy of their model at later timesteps compared to a naive LSTM model when modelling the Lorenz system.

In a similar fashion, Lusch et al. [2018] used a modified encoder-decoder model to identify nonlinear coordinates of dynamical systems on which the dynamics are linear. Their encoder-decoder was trained to both reconstruct snapshots of the current state and to predict future timesteps by multiplying its latent vector with a Koopman matrix before decoding it. They parameterised their Koopman matrix by using a separate neural network to specify its eigenvalues given the current latent vector. In this way they were able to learn interpretable low-rank linear models of dynamical systems, including those with continuous spectra, such as a nonlinear pendulum and a high-dimensional nonlinear fluid flow. However, a limitation of both these approaches is that the dimensionality of the latent space is a hyperparameter which must be chosen.

2.3.1.5 Physically constrained Gaussian processes

Gaussian processes (GPs) are flexible probabilistic models which have a wide range of applications, and many approaches exist for asserting physical constraints within

them [Swiler et al., 2020]. For example, Raissi et al. [2017] proposed an approach for inverting parameters of linear differential equations using GPs. Importantly, the linear equations were directly incorporated into the design of the GP. Specifically, they noted that any linear transformation (including differentiation) of a GP is another GP with a suitably transformed covariance kernel, where any parameters of the linear operator become learnable hyper-parameters of the transformed kernel. Thus, training a combined GP to model noisy observations of both the solution and the forcing term of a linear differential equation allowed them to successfully estimate its underlying parameters. Furthermore, this approach circumvented the need for discretisation and naturally accounted for noise in the observational measurements. They later extended this concept to nonlinear differential equations too [Raissi and Karniadakis, 2018].

2.3.1.6 Other approaches

Many other approaches exist which ensure the architecture of the ML algorithm conforms with some prior scientific knowledge and some more examples are given below.

Jumper et al. [2021] presented AlphaFold, which is a deep learning algorithm for predicting 3D protein structures based on their amino acid sequences. Their algorithm incorporated many design features which were motivated by physical and biological knowledge about protein structure. Specifically, they focused on learning a pairwise representation of the residues in the protein which emphasised the learning of the 3D relationships between them, rather than a sequential representation based on their chain. Furthermore, they used various "triangular updates" to update these pairwise features using all features which formed length-3 cycles with them. This emphasised the fact that residuals should be constrained by their neighbours, for example the paths between residues should obey the triangle inequality. When computing the 3D rotation and position of each residue in the protein, a local frame of reference was used for each residue which ensured invariance of the prediction to the global frame. They showed that all of these domain-inspired design choices were required to achieve optimal performance.

Mohan et al. [2020] proposed a CNN for learning compressed representations of velocity fields in turbulent flows where the notion of an incompressible fluid was hard-coded into the CNN design. Specifically, they noted that the velocity v of an incompressible fluid should obey the continuity equation, $\nabla v = 0$. To ensure this condition, they took the output of the CNN, denoted as a, and defined its final velocity prediction to be $\tilde{v} = \nabla \times a$, where the spatial gradients of a were computed using a set of non-trainable finite difference convolutional filters. By noting that the divergence of the curl of any vector field is zero, one can see that the CNN prediction always obeys the continuity equation. In contrast, they showed that a CNN which predicted the velocity field directly significantly deviated from this constraint.

2.3.2 Loss function

The next class of approaches we consider are those where the loss function used to train the ML algorithm is modified so that it includes scientific constraints. As the ML model is trained, these constraints encourage the model to become consistent with some prior knowledge. This is usually considered a "soft" way of imposing constraints as the learned model typically only minimises the loss function and can therefore still disobey its constraining terms. From a machine learning perspective, such terms in the loss function can be seen as regularisers (or from a Bayesian modelling perspective, physical priors) which restrict the possible space of models which can be learnt. Including scientific knowledge in this way can significantly improve the convergence of ML models, allow them to generalise better and reduce the amount of training data required.

2.3.2.1 Constraining physical quantities

Modifying the loss function is a very general idea which can be achieved in a variety of ways. One way to do this is to constrain the values of certain physical quantities in the ML model. The hope is that by doing so these quantities become more physically consistent.

For example, Karpatne et al. [2017b] used a fully connected neural network to predict the temperature depth profile of lakes given a set of input drivers including wind speed, air temperature and the time of year. Importantly, they used a known empirical relationship to convert the predicted temperature profiles to density profiles, and then defined an additional ReLU-type loss function to encourage the changes in the density values to be positive, i.e. such that the density profile increased with depth, which is a known physical constraint. They found that using this loss term alongside a standard supervised loss both improved the accuracy of the network's predictions and made them more physically consistent compared to the same network trained without it.

In molecular dynamics, Zhang et al. [2018] trained a fully connected neural network to estimate the atomic energies of molecules given the positions of each atom in the molecule, using a loss function that not only constrained the atomic energy but also the resulting atomic force and virial tensor of the molecule. The resulting network was able to make accurate predictions across water, ice and organic molecules of different compositions and size.

Benjamin Erichson et al. [2019] used ideas from Lyapunov analysis to define a loss function which encouraged stability when training encoder-decoders to model the time evolution of dynamical systems. They assumed the dynamical system could be described by linear dynamics in a low-dimensional latent space z such that $z_{t+1} = \Omega z_t$. They learned the matrix Ω and the mapping from the state to this latent space by training an encoder-decoder network to both reconstruct snapshots of the current state and predict future timesteps by multiplying its latent vector with Ω before decoding it. Importantly, they encouraged the latent dynamics to be Lyapunov-stable (i.e, that all trajectories starting close to an origin state remain arbitrarily close at later times) by defining an additional loss term which encouraged the positive definiteness of the matrix P, where $\Omega^\top \mathrm{P}\, \Omega - \mathrm{P} = -\mathrm{I}$. They showed via Lyapunov's second method that this condition ensured stability if held. Including this loss term significantly improved the extrapolation ability of the encoder-decoder when modelling the fluid flow past a cylinder.

Xie et al. [2018] introduced TempoGAN, which is a generative adversarial network (GAN) [Goodfellow et al., 2014] for super-resolving coarse-resolution simulations of fluid flows. They used a standard conditional generator-discriminator setup, where the inputs to the generator were the spatially-varying velocity and density of the flow at a single snapshot in time, and the discriminator assessed whether the super-resolved density field output by the generator was fake or not. Importantly, they added a second discriminator which asserted that the super-resolved outputs were temporally coherent. Specifically, this discriminator assessed whether sequences of density fields output by the generator were fake or not. They found that adding this temporal discriminator improved both the temporal and spatial consistency of the super-resolved flows.

Finally, in the context of training neural network surrogates to model the outputs of stochastic simulators, Brehmer et al. [2020] showed that intermediate quantities extracted from within the simulator could be used to help train the surrogate model. In particular, they defined a "joint score" and a "joint likelihood ratio" based on the conditional transition probabilities between the latent states of the simulator, and showed theoretically that these could be used to constrain the gradient of the likelihood and the likelihood ratio estimated by the surrogate model when included in the loss function. Incorporating these constraints led to more accurate surrogate models which required much less training examples when modelling problems including the Lotka-Volterra system and the Galton board.

2.3.2.2 Encoding conservation laws

Another possibility is to encourage the ML model to obey certain conservation laws through its loss function. Similar to above, the hope is that this allows the ML model to be more physically consistent.

For example, in climate modelling, Beucler et al. [2021] trained a supervised neural network surrogate to emulate subgrid cloud processes, where the input to the network was a set of variables describing the large-scale thermodynamic state of the atmospheric column and the output was a set of variables describing the

rate at which subgrid convection vertically redistributes heat and water. They added appropriate combinations of these inputs and outputs as an extra term in the loss function such that the conservation of energy, mass and long- and short-wave radiation was encouraged. Taking this a step further, they also proposed an approach for analytically conserving nonlinear constraints of a network's inputs and outputs by rewriting them as linear constraints over auxiliary variables and using the network to solve the subsequent linearly constrained optimisation problem. They showed that both approaches achieved similar prediction performance to a naive neural network whilst achieving greater physical consistency.

Zeng et al. [2021] used additional terms in their loss function to enforce mass conservation when training GANs to model potential flow velocity fields. Specifically, they added a loss term which imposed that the divergence of their generator output was zero, using finite differences computed over its output. They found that this constraint both improved the physical consistency of the generator output and accelerated the convergence of the GAN.

Greydanus et al. [2019] proposed Hamiltonian neural networks (HNNs) for modelling the time evolution of dynamical systems, where instead of using the neural network to directly predict the evolution of the state, the neural network was used to predict its Hamiltonian, $\mathcal{H}(p, q)$, assuming a set of canonical coordinates describing the state, (p, q), were available, i.e. $NN(p, q; \theta) \approx \mathcal{H}(p, q)$. The network was trained using a loss function which matched its derivatives to observations of the temporal derivatives of the coordinates such that its output satisfied Hamiltonian's equations, i.e. $\mathcal{L}(\theta) = ||\frac{\partial NN}{\partial p} - \frac{dq}{dt}||^2 + ||\frac{\partial NN}{\partial q} + \frac{dp}{dt}||^2$. This learned Hamiltonian was then numerically integrated to predict the time evolution of the coordinates. By construction, these networks naturally learned and respected conservation laws within the training data. For example, they showed that a HNN could more accurately predict the time evolution and conserve the total energy of two bodies under the influence of gravity compared to a baseline network trained to directly predict the temporal derivatives of the coordinates. Toth et al. [2020] extended this idea by showing that the canonical coordinates of the system could also be learnt by adding trainable

encoder and decoder layers to map raw observations (such as images of two bodies moving under the influence of gravity) to these coordinates. Cranmer et al. [2020] proposed Lagrangian neural networks, which followed an analogous idea by modelling the Lagrangian of the system. These networks also learned and respected conservation laws but acted on generalised coordinates rather than canonical coordinates.

2.3.2.3 Auxiliary tasks

Rather than using scientific terms in the loss function which constrain the current task, another approach for encoding scientific knowledge into ML algorithms is to train them in a multi-task setting [Crawshaw, 2020]. Here, the ML model is trained to learn multiple tasks simultaneously. The additional tasks may provide the ML model with more awareness of the context surrounding the main task and allow the model to learn shared representations, which can improve its generalisation and result in improved sample efficiency.

In particle physics, de Oliveira et al. [2017] trained a GAN using multi-task learning to generate jet images, which are 2D representations of the energy deposited from particles interacting with a calorimeter inside a particle collider. The GAN was trained using many example jet images, and importantly its discriminator was tasked with not only classifying whether the generated images were fake or real, but also whether the jet images originated from high energy W bosons or generic quark and gluon events. Their workflow followed the auxiliary classifier GAN proposed by Odena et al. [2017], with some further modifications to help model the sparsity of the jet images. They showed that their GAN was able to realistically generate jet images, honouring multiple physically-relevant statistics across their distribution. However, they noted that the GAN may have favoured generating images which were very distinguishably W-like or quantum chromodynamics (QCD)-like jet images because of the additional discriminator task, and thus under-represented those images which were hard to distinguish.

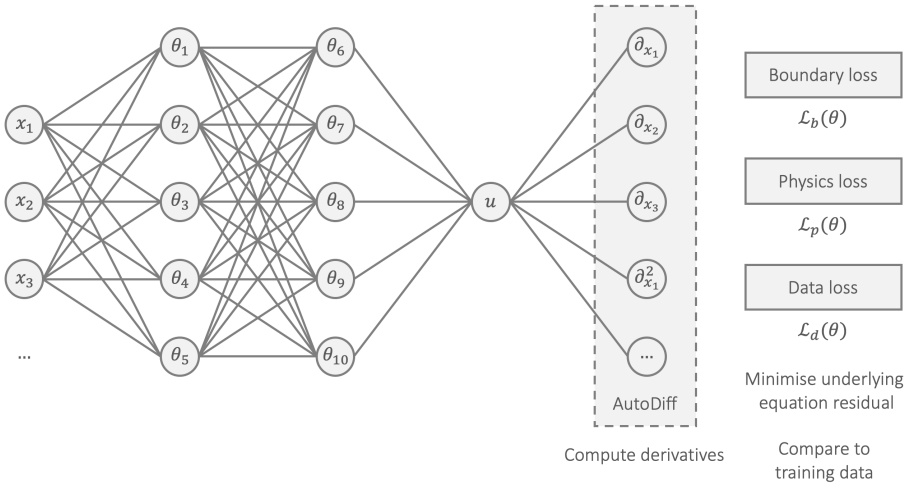


Figure 2.3: Schematic of a physics-informed neural network (PINN). A PINN is a neural network, $NN(x;\theta)$, with trainable parameters θ, which directly approximates the solution, $u(x)$, to a differential equation, i.e. $NN(x;\theta) \approx u(x)$. To train the PINN, a loss function is used which is composed of two terms, one termed the "boundary" loss which tries to match the PINN solution to the known solution (and/or its derivatives) along the boundaries of the domain, and another termed the "physics" loss which tries to minimise the residual of the underlying equation at a set of locations within the domain. The derivatives of the PINN solution with respect to its inputs required by the boundary and physics loss are obtained using autodifferentiation. PINNs can also be used for inverse problems, in which case an additional "data" loss is added which compares the PINN solution with known solution values at additional locations within the domain, and parameters of the underlying equation are jointly optimised alongside θ.

2.3.2.4 Encoding governing equations

Instead of just adding constraints on the physical quantities of a system into the loss function, a more powerful constraint may be to include knowledge of the underlying governing equations of the system themselves. By doing so, one could hope that the ML model learns to obey these equations in general.

This idea has led to multiple general-purpose SciML approaches, each of which are rapidly turning into their own sub-field of SciML. Furthermore, it has provided new perspectives on how to carry out key scientific tasks such as simulation, inversion and equation discovery.

Physics-informed neural networks

One recent and popular way of doing this is to use physics-informed neural networks (PINNs), which are designed to solve problems related to differential equations. The

starting ideas behind PINNs were developed in the 1990s by Lagaris et al. [1998] and others, and they were recently reintroduced and extended by Raissi et al. [2019] using modern deep learning techniques. Since these works PINNs have quickly become their own sub-field of SciML [Karniadakis et al., 2021, Cuomo et al., 2022].

A generic schematic of a PINN is shown in Figure 2.3. At a high level, a PINN is designed to model the solution, $u(x)$, of a differential equation by using a neural network, $NN(x;\theta)$, with learnable parameters θ, to directly approximate the solution, i.e. $NN(x;\theta) \approx u(x)$.

Importantly, the PINN learns the solution by using a loss function which directly penalises the residual of the underlying equation. More precisely, the loss function takes the form

$$\mathcal{L}(\theta) = \mathcal{L}_b(\theta) + \mathcal{L}_p(\theta) \ , \tag{2.3}$$

where

$$\mathcal{L}_b(\theta) = \sum_k \frac{1}{N_{bk}} \sum_j^{N_{bk}} || \, \mathcal{B}_k[NN(x_{kj};\theta)] - g_k(x_{kj}) \, ||^2 \ , \tag{2.4}$$

$$\mathcal{L}_p(\theta) = \frac{1}{N_p} \sum_i^{N_p} || \, \mathcal{D}[NN(x_i;\theta);\lambda] - f(x_i) \, ||^2 \ , \tag{2.5}$$

given all of the definitions and nomenclature from Equation 2.2[2]. The first term, denoted the "boundary loss", $\mathcal{L}_b(\theta)$, encourages the network's prediction to match the true solution (and/or its derivatives) at a subset of known locations, $\{x_{kj}\}$, in the domain (typically along the boundaries). The second term, denoted the "physics loss", $\mathcal{L}_p(\theta)$, tries to ensure that the PINN solution obeys the underlying differential equation of the system by minimising the residual of the differential equation at a set of points, $\{x_i\}$, sampled over the entire domain. Intuitively, the physics loss pushes the neural network to learn a solution which is consistent with the underlying differential equation, whilst the boundary loss attempts to ensure the solution is unique by matching the solution to the known boundary conditions.

[2]Note other classes of differential equations can be incorporated too, e.g. Pang et al. [2020, 2019].

Thus, training a PINN amounts to solving a differential equation. Compared to a naive neural network trained to model the solution using only example solution points, a PINN requires much less training data and is able to extrapolate away from these example solution points. Furthermore, PINNs provide potential advantages over traditional methods when used to solve differential equations. Firstly, they provide a continuous, mesh-free solution, and therefore one does not have to design discretisation schemes used in traditional approaches such as finite difference and finite element modelling. Secondly, they provide analytical and tractable gradients with respect to their inputs, which are valuable for tasks such as inversion and sensitivity analysis. Finally, PINNs can be extended to carry out inversion tasks. In this setting, a "data loss" is usually added which penalises the difference between the network solution and the solution at a set of observed points within the domain, and inversion parameters (such as the parameters of the differential equation, λ) are jointly optimised alongside the network solution.

PINN applications

PINNs have been utilised in a wide range of applications spanning many domains [Cuomo et al., 2022]. In their original paper, Raissi et al. [2019] showed that PINNs can accurately solve a range of canonical equations including the Burgers equation, Schrödinger equation and Navier-Stokes equations. Two more detailed examples are described below.

Jin et al. [2021] proposed NSFnets, which consisted of two PINNs designed for solving the Navier-Stokes equations, one using the velocity-pressure formulation of the equations and the other using the vorticity-velocity formulation. They showed that both designs were able to accurately solve various 2D and 3D laminar, unsteady and turbulent flow problems with a Reynolds number of 1,000. However, they noted that the computational cost of training their PINNs was 5-10 times higher than traditional computational fluid dynamics (CFD) solvers. Transfer learning was used to alleviate this issue, by initialising the weights of the PINN using a separate PINN pretrained on a lower Reynolds number. They showed that this significantly improved both the training time and accuracy of the higher Reynolds number PINN.

They also showed that the PINNs could be used in an inversion setting to estimate unknown Reynolds numbers, and that adaptive weighting between the two terms in the PINN loss function could be used to improve performance.

Chen et al. [2020] used PINNs in an inversion setting to solve inverse scattering problems related to photonic meta-materials and nano-optics. More specifically, they used a PINN to estimate the unknown, spatially-varying effective electric permittivity, $\varepsilon(x, y)$, of arrays of dielectric nanocylinders given measurements of the electric field strength, $E(x, y)$, resulting from their transverse magnetic wave excitation. The electric field strength was governed by the Helmholtz equation, and this was used in the PINN loss function. They found that the PINN could accurately model the electric field strength and recover the effective permittivity of the nanocylinders, for periodic and aperiodic arrays of nanocylinders. Furthermore, they used their approach to design nanocylinders which could serve as invisible cloaking devices, achieving a 75% scattering abatement when the size of the cylinder was comparable to the wavelength of the incoming radiation.

PINN extensions

Furthermore, many extensions of PINNs have been proposed. A few notable examples are discussed below; they include using alternative formulations of the underlying equation, incorporating uncertainty and discovering the underlying equations themselves.

Kharazmi et al. [2019] proposed variational PINNs (VPINNs) which instead trained PINNs using the variational form of the underlying differential equations. Their neural network was still used to approximate the solution of the differential equation, but it was combined with a set of analytical test functions to compute the residual of the variational form of the equation in its physics loss term. Furthermore, they used quadrature points to estimate the corresponding integrals in the variational loss, rather than random collocation points. They found that the VPINN was able to solve differential equations including Poisson's equation with similar or better accuracy to a PINN trained using the strong form, whilst requiring less collocation points to train. However, a downside of their approach is that additional

approximation errors were introduced through the use of finite sets of test functions and numerical integration in the variational loss. Kharazmi et al. [2021] later extended their approach by using domain decomposition over the set of test functions, more closely emulating traditional finite element method (FEM) solvers.

Yang et al. [2021a] presented Bayesian PINNs (B-PINNs) which combined PINNs with Bayesian inference. First, they assumed that one had access to a set, $\mathcal{D}$, of noisy measurements of $u(x)$, $f(x)$ and $g_k(x)$ in Equation 2.2, and that these measurements were independently and identically Gaussian-distributed around their true values. Next, they assumed that the solution to the differential equation was approximated by a Bayesian neural network (BNN), $BNN(x;\theta) \approx u(x)$, where they assumed the parameters of the network, θ were random variables with some prior distribution, $p(\theta)$. Given these assumptions, they could write the likelihood of the measurements, $p(\mathcal{D} \mid \theta)$, as a product of Gaussians, where $f(x)$ and $g_k(x)$ were rewritten in terms of $BNN(x;\theta)$ according to Equation 2.2. They then used standard Bayesian inference methods, such as Hamiltonian Monte Carlo (HMC) and variational inference (VI), to infer the posterior distribution of the parameters of the BNN, $p(\theta \mid \mathcal{D}) \propto p(\mathcal{D} \mid \theta)\, p(\theta)$. This approached allowed them to both solve the differential equation and provide uncertainty estimates on the solution, whilst naturally accounting for noisy training data. Furthermore, they extended their approach to inverse problems by assuming priors over inversions parameters, such as λ in Equation 2.2. They found that in general HMC provided more robust predictions and uncertainty estimates than VI for the problems they considered (Poisson equation, Allen-Cahn equation), and that B-PINNs provided more accurate solutions than standard PINNs in the presence of large amounts of noise. However, a limitation of their approach was that the cost of HMC typically scales poorly to large dataset sizes and they only tested cases with up to several hundred measurements.

Chen et al. [2021] combined PINNs with sparse regression such that their algorithm not only learned the solution of differential equations but also discovered the differential equations themselves too. They used an optimisation strategy that alternated between carrying out sparse regression over an extensive library

of candidate functions of the PINN solution and its gradients in order to guess the underlying differential equation, and using this guessed equation and example solution points in a standard PINN loss function to update the weights of the PINN. In this way, the algorithm set up a positive feedback loop between the PINN and the sparse regression. They showed that their method was able to robustly discover a variety of differential equations, including the Burgers equation, Kuramoto-Sivashinksy equation, nonlinear Schrödinger equation, Navier-Stokes equations and reaction-diffusion equation.

PINN limitations

Finally, PINNs also have a number of important limitations, and multiple works have attempted to address them. These are discussed below.

A major source of difficulty is that PINNs can struggle to converge. PINNs often contain millions of free parameters and it is unclear how best to optimise them. Indeed, in contrast to traditional numerical methods, their theoretical convergence properties are not well understood. Some work has started to understand the theoretical properties of PINNs, but it is in its early stages [Mishra and Molinaro, 2022, Shin et al., 2020, Wang et al., 2022, 2021c].

This has led to many works proposing (mostly empirical) strategies for improving their convergence. For example, Wang et al. [2022] proposed a loss balancing strategy based on insights from neural tangent kernel theory, Anitescu et al. [2019] and Lu et al. [2019] proposed adaptive refinement methods where more training points were added where the physics loss was high, Jagtap et al. [2020a,b] used learnable activation functions, Avrutskiy [2020] and Son et al. [2021] included higher-order derivatives of the underlying equation in the loss function, and Lyu et al. [2022] used auxiliary variables to rewrite higher-order differential equations into first-order equations before using them in the loss function. All of the above methods, at least for the various cases studied, showed significant convergence improvements over standard PINN implementations.

Another related limitation is that PINNs often struggle to scale to problems which have larger domains and/or multi-scale solutions. This limitation is the central focus of Chapter 6 and discussed in detail in Section 6.2.1, and so we refer the reader there.

Lastly, when used purely for forward simulation tasks, PINNs are usually less efficient than traditional approaches such as finite difference and finite element methods, because of their large optimisation costs and the fact they need to be retrained for each simulation [Karniadakis et al., 2021]. However, PINNs can outperform traditional approaches when used for inversion problems, where traditional inversion algorithms often require thousands of forward simulations to converge.

Physics-informed neural operators

A related set of approaches which incorporate governing equations into their loss function are physics-informed neural operators. These are neural networks which are similar to PINNs in that they are designed to learn the solution to differential equations, but instead of learning a single solution they learn an entire family of solutions by adding certain inputs of the differential equation (such as those contained in a in Equation 2.1) as inputs to the network. Thus, they do not need to be retrained to carry out new simulations, and during inference they offer a fast surrogate model. From a mathematical standpoint, the goal is to learn an operator to map function spaces to function spaces, rather than just a single function. Indeed, there is a growing sub-field of ML concerned with learning operators [Lu et al., 2021, Li et al., 2020b], and we discuss some of the physics-informed approaches here.

Wang et al. [2021b] proposed physics-informed deep operator networks, which essentially conditioned PINNs on input functions of the differential equation by using the DeepONet architecture introduced by Lu et al. [2021]. Their DeepONet consisted of two sub-networks, a "branch" network which encoded the input function (by taking discretised samples of the input function at fixed locations as input), and a "trunk" network which encoded a set of input coordinates. The solution of the differential equation was then approximated by merging the outputs of both of these sub-networks. The network was trained using a loss function that extended the PINN loss function (Equation 2.3) by averaging over many random samples

of the input function. They showed that this approach was able to learn fast and accurate surrogate models of the solution to a nonlinear diffusion-reaction system given the source term of the equation as input, and of the solution to Burgers equation given the initial condition as input.

In a related approach, Li et al. [2021] proposed physics-informed neural operators (PINOs), which used a physics-informed loss function when training Fourier neural operators (FNOs) introduced by Li et al. [2020b]. Similar to DeepONets, FNOs learn an operator to map between function spaces. This is achieved by using a series of stacked Fourier layers, where the input to each layer is Fourier transformed and truncated to a fixed number of Fourier modes. This truncation allows the model to learn mappings which are invariant to the number of discrete points used in its inputs and outputs. Their PINO was trained using a similar loss function to Wang et al. [2021b], evaluated by averaging over many random samples of the input function. They showed that this approach was able to learn efficient and accurate surrogate models for a range of systems, including the Burgers equation, Darcy flow and chaotic Kolmogorov flow.

Physics-informed neural networks on meshes

Another way to include governing equations in the loss function when training neural networks to approximate their solutions is to use finite difference filters to evaluate the residual of the differential equation [Zhu et al., 2019]. This is useful when the neural network is used to output a discretised version of the solution on a mesh of points, rather than represent a functional approximation of the solution. The advantage of this approach is that architectures which take advantage of the spatial relationships between their inputs and outputs can be easily used, such as CNNs. Similar to PINNs, this approach provides a way to solve differential equations with very little training data.

Zhu et al. [2019] showed this approach could be used to learn single solutions to differential equations by providing a fixed arbitrary input to the network, or a surrogate model by providing discretised versions of an input to the differential equation as input and averaging over these in the loss function during training. Geneva

and Zabaras [2020] extended this approach by proposing physics-informed auto-regressive networks for solving time-dependent differential equations, and Gao et al. [2021] extended it to learn solutions on non-rectangular meshes by first transforming the problem onto a rectangular mesh and then learning the solution and evaluating the physics loss in this transformed domain.

Other approaches

In a different approach, Champion et al. [2019] used a loss function which included high-level assumptions about the underlying equation of a dynamical system to both learn the underlying equation and a coordinate system in which it was parsimonious. More specifically, they used an autoencoder to learn a coordinate transform from the observed state of a system to lower-dimensional latent representation of the state. The autoencoder was trained by using a loss function which tried to reconstruct example snapshots of the state and match the temporal derivatives of the latent coordinates to a library of candidate functions via sparse regression, in a similar fashion to the SINDy algorithm [Brunton et al., 2016]. They showed that their approach was able to learn a low-dimensional coordinate transform and a corresponding parsimonious underlying equation for a nonlinear pendulum given high-dimensional 2D images of the system as training data, and for a reaction-diffusion system given high-dimensional snapshots of its solution discretised on a grid.

2.3.3 Hybrid approaches

Finally, we consider hybrid approaches, where rather than using a ML algorithm to solve a scientific task by itself, the ML algorithm is tightly integrated with a traditional algorithm. The combined result can be thought of as a learnable system, strongly informed by our prior knowledge but flexible enough to adapt to the data at hand. Designing a hybrid approach can accelerate and improve the accuracy of traditional algorithms and provide more interpretable ML outputs. However, hybrid approaches require deep knowledge of ML, the scientific task at hand and the traditional methods for solving it. This is the broadest category presented and

such approaches vary significantly by their intended scientific task and by how their traditional and learned components interact.

2.3.3.1 Residual modelling

Perhaps the simplest way to combine ML with traditional algorithms is to carry out residual modelling. This is where a ML model is trained to correct the outputs of a traditional algorithm, and it only sees its inputs and outputs, i.e. the traditional algorithm is treated as a "black box". Depending on the scientific task at hand, there are many reasons why residual modelling is useful; for example it can be used to correct traditional workflows where they are inexact due to computational restrictions, or unable to account for some unknown physics.

For example, Pawar et al. [2021] designed a neural network to predict the forces acting on an aerofoil given the aerofoil shape, physical parameters of the flow and predictions of the forces from a simplified theory (specifically, the Hess-Smith panel method) as inputs. They inputted the force predictions by concatenating them to intermediate layers of their network rather than the input layer, and used an ensemble of networks to quantify the uncertainty in the prediction. They found that their approach provided more accurate predictions with less uncertainty than a naive neural network trained to predict the forces given only the physical parameters of the flow as input, whilst remaining faster than full CFD simulation.

Super-resolution can be thought of as type of residual modelling, and in a more involved example, Jiang et al. [2020] proposed MeshfreeFlowNet, which learned to super-resolve turbulent flows in a Rayleigh-Bénard convection problem given the output from a coarse CFD solver. They used two neural networks applied sequentially, where the first network generated a latent context grid given the output of the CFD solver. Then, the second network estimated the super-resolved flow parameters, given a query coordinate and the neighbourhood of latent context values around this location. In addition to a standard supervised loss function, they regularised the output of the second network by minimising the residual of the underlying equations in a similar fashion to the PINNs discussed above. They found that MeshfreeFlowNet

strongly outperformed a standard convolutional super-resolution architecture across a range of physical metrics including total energy, dissipation rate, and eddy turnover time, with the physics loss helping to improve its accuracy.

2.3.3.2 Differentiable physics

A potentially more powerful hybrid approach is to open up the black box of a traditional algorithm and tightly integrate ML models within it. This allows a more granular way of balancing the two paradigms; ML can be inserted where we are unsure how to solve a problem, or where the traditional workflow is computationally expensive, and traditional components can be kept where we require robust and interpretable outputs. Often, the performance of the traditional workflow is improved whilst the ML components are easier to train, are more interpretable, and require less parameters and training data compared to a naive ML approach.

A general approach for doing so is to use concepts from the field of differentiable physics [Thuerey et al., 2021, Ramsundar et al., 2021, Belbute-Peres et al., 2018, Monga et al., 2021]. In particular, the central realisation of this field is that many traditional scientific algorithms can be written as a composition of basic and differentiable mathematical operations (such as matrix multiplication, addition, subtraction, etc), and that modern automatic differentiation and differential programming languages [Baydin et al., 2018] make it easy to track and backpropagate the gradients of these outputs with respect to their inputs. This unlocks the possibility of inserting and training gradient-based ML components (such as neural networks) within traditional workflows, whereas otherwise it may have been difficult to do so. From a ML perspective, one could imagine the combined workflow as a new type of neural network, where some of its operations are fixed and others are learnable. This is a powerful and flexible concept, and it is an essential part of many of the remaining approaches studied in this chapter.

A simple way to start is to re-implement a traditional workflow inside a modern differentiable programming language. For example, Ren et al. [2020] re-implemented full waveform inversion (FWI), an algorithm used to invert for an Earth model

given its seismic response, using the PyTorch framework [Paszke et al., 2019] and showed that computing gradients of the seismic response with respect to the Earth model using automatic differentiation was mathematically equivalent to traditional adjoint-state methods. Furthermore, they showed that using modern deep learning optimisation strategies (such as using the Adam optimiser [Kingma and Ba, 2015]) within the FWI algorithm gave competitive performance compared to standard FWI.

In a similar fashion, Minkov et al. [2020] implemented traditional mode-expansion methods for simulating the optical response of photonic crystals in an automatic differentiation framework and then harnessed automatic differentiation to optimise their design. In this case, computing gradients of the traditional methods by hand was challenging because of the complexity of the simulation method and so automatic differentiation opened up a new class of inversion algorithms.

Once a traditional algorithm is implemented, its design can be altered by treating certain parameters as learnable, or by inserting new learned components. For example, Würfl et al. [2018] reframed traditional filtered back-projection-type algorithms for carrying out 3D tomographic reconstruction as neural networks. They took this a step further by learning key hyperparameters of the algorithms, such as their compensation weights and apodization windows, by optimising the network's performance over many example reconstructions. They showed that their approach consistently outperformed the traditional algorithms while keeping the same test-time computational complexity by design. Furthermore, the network was easy to interpret as its learned hyperparameters could be directly compared to those used in traditional algorithms.

Similarly, Zhang et al. [2019a] reframed a traditional iterative prox-linear solver for estimating the voltages across a power grid by unrolling the solver over a fixed number of iterations into a ResNet-like neural network architecture. Then, they treated certain matrices in the solver as learnable, relaxing its formal definition, and optimised the network using many example outputs. They showed that this network significantly outperformed a naive fully connected network and a more sophisticated Gauss-Newton solver.

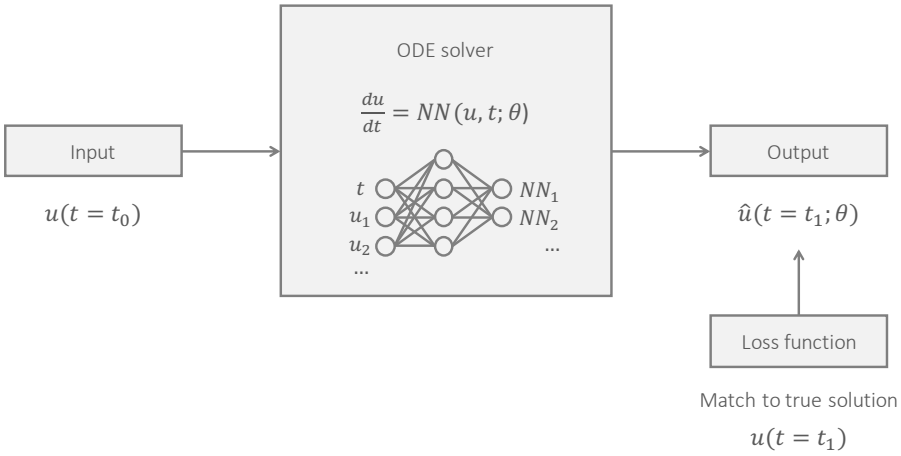


Figure 2.4: Schematic of a neural ordinary differential equation (ODE). The goal of a neural ODE is to learn the right-hand side term of an unknown ODE. A neural network $NN(u, t; \theta)$ is used to represent this term, which is trained by using many examples of the solution of the ODE at two times, $u(t = t_0)$ and $u(t = t_1)$. More specifically, a standard ODE solver is used to model the solution of the ODE, $u(t = t_1)$, at time $t = t_1$ given the solution at time $t = t_0$ and evaluations of the network where needed. Then, the network's free parameters, θ, are updated by matching this estimated solution with the true solution and differentiating through the entire ODE solver.

2.3.3.3 Neural differential equations

The approaches above are simple use cases of differentiable physics in that they only learn existing parameters in the traditional workflow. However, differentiable physics is more flexible than this; it also allows more complex ML modules (such as neural networks) to be inserted into traditional algorithms, and for these to be learnt by training the entire algorithm end-to-end.

An example of this are neural differential equations [Chen et al., 2018a, Rackauckas et al., 2020], which combine neural networks with traditional numerical solvers in order to discover terms in underlying equations. Neural differential equations were first proposed in the field of ML by Chen et al. [2018a] in the context of continuous-depth models. More precisely, they used a neural network to learn the right-hand side of the ordinary differential equation (ODE)

$$\frac{du}{dt} = NN(u, t; \theta) \ , \tag{2.6}$$

where $u \in \mathbb{R}^d$, $t \in \mathbb{R}^1$ and θ were the free parameters of the neural network.

The network was trained using a standard ODE solver. More specifically, it was iteratively updated using a loss function which matched the output of the ODE solver, $\hat{u}(t = t_1)$, at time $t = t_1$, given the solution $u(t = t_0)$ at time $t = t_0$ and evaluations of the network where needed, to the true observed solution of the ODE, $u(t = t_1)$, and backpropagating through the entire ODE solver. A schematic of this workflow is shown in Figure 2.4.

From a differentiable physics perspective, this workflow inserts a neural network into a traditional ODE solver in order to learn the underlying equation. From a ML perspective, the workflow offers a novel class of trainable, continuous-depth models, which do necessarily need to relate to a physical system, where the input to the model is the solution at time t_0 and its output is the solution at time t_1. The simplest ODE solver is an Euler integrator, which discretises the solution in time. In this case Chen et al. noted the resulting model was analogous to a ResNet [He et al., 2016].

Rackauckas et al. [2020] extended this idea to learning arbitrary terms in more complex differential equations, such as partial differential equations (PDEs). In a similar fashion, a neural network was used to represent a term (or set of terms) in the equation and was trained by differentiating through a numerical solver. They also presented efficient adjoint methods for computing the gradients required to train the network, and a strategy for converting the learned terms into symbolic expressions via sparse regression. They showed that their approach was able to accurately discover unknown terms in the Lotka-Volterra equation, Boussinesq equation and a reaction-diffusion system.

Long et al. [2019] proposed a similar approach to Chen et al. [2018a] and Rackauckas et al. [2020], but they instead used a symbolic neural network to learn a symbolic representation of the PDE. Specifically, they proposed PDE-Net, which was designed to discover governing equations with the generic form $u_t = F(u, u_x, u_y, u_{xx}, u_{xy}, u_{yy}, ..)$ where $u(x, y, t)$ was the solution to the equation and $(x, y, t) \in \mathbb{R}^3$. First, given a discrete mesh of solution observations at a starting timestep $t = t_0$, they used a series of convolutional filters to estimate the spatial derivatives of the solution. Next, a symbolic neural network was used to define the

nonlinear function F given these derivatives. The structure of the neural network was defined such that only multiplicative and summative relationships between its inputs could be learnt, allowing a symbolic representation of the underlying equation to be easily extracted. A discrete time-stepping Euler scheme was used to solve for the solution a later timesteps using the neural network, and the network was trained by matching these predictions to solution observations at later timesteps. They showed their method was able to accurately discover a range of (2+1)D PDEs including the Burgers equation and the diffusion equation.

2.3.3.4 In-the-loop methods

The final class of approaches for combining ML with traditional algorithms we consider is one we define as "in-the-loop" methods. Many traditional algorithms are fundamentally iterative, and in these approaches an ML component is used in the inner loop of the algorithm[3]. Using an ML component in this way can lead to less iterations being required, increasing the efficiency of the algorithm, and result in more accurate outputs. A selection of approaches for both forward simulation and inversion tasks are discussed below.

Forward simulation

Forward simulation algorithms (such as finite difference and finite element methods) are often iterative and many works exist which combine them with ML components. For example, Um et al. [2020] used a neural network in the inner loop of a coarse numerical solver to correct the discretisation error in its solution at each timestep. Importantly, rather than training this network using many example high-resolution and low-resolution solution pairs in a naive fashion, they unrolled their hybrid algorithm over a fixed number of timesteps and differentiated through the solver in order to train the network. This "solver-in-the-loop" strategy ensured that the correction network experienced states which were not purely on the manifold of coarse solver states, and that the network learned to correct its own corrections.

[3] A related but slightly narrower concept is the "unrolling" of traditional iterative algorithms, where these algorithms are unrolled and their parameters are learned through end-to-end training [Monga et al., 2021, Gregor and LeCun, 2010] (e.g. see Zhang et al. [2019a] described above).

They found that this strategy was able to more accurately correct the coarse solver over longer roll-outs than the same network trained naively, for simulating a range of problems including unsteady wake flow and an advection-diffusion system.

Learning subgrid parameterisations can be viewed as an in-the-loop method, where a ML model is used to predict the effect of a subgrid process at every timestep within a numerical solver. There is a significant amount of work in this area, and one example is Rasp et al. [2018], who trained a deep neural network to represent atmospheric processes in a global climate model. Their network was allowed to freely interact with the simulator and they showed it was able to produce simulations which were more stable and better reproduced key aspects of variability such as extreme precipitation than a traditional subgrid parameterisation, although it struggled with out-of-sample climates.

Inversion

Many inversion algorithms are iterative and there are many ways they can be combined with ML [Arridge et al., 2019]. We discuss two major themes below.

Learned gradient descent

One option is to use ML to learn gradient descent steps [Adler and Öktem, 2017, Andrychowicz et al., 2016]. Many inversion algorithms use gradient descent to iteratively estimate the unknown parameters of a system based on the gradient of some misfit function, although, depending on the difficulty of the problem, gradient descent can suffer from some well-known limitations such as trapping in local minima and unstable convergence. Instead, given the current state of the inversion algorithm (for example the current parameter values and the gradient of the misfit function with respect to the parameters), a ML model can be trained to estimate the best direction to move in at each iteration of the algorithm. The ML model can be inserted into the inner loop of the inversion algorithm, then the entire algorithm unrolled and the network trained end-to-end using many example inversion problems.

Adler and Öktem [2017] used this strategy when carrying out 2D inversion of images of human heads given their projection data from simulated computed

tomography (CT) scans, using a training dataset of similar images and their CT scans. They found that their learned gradient scheme outperformed a standard primal-dual iterative algorithm with total variation regularisation, providing more accurate reconstructions whilst being significantly faster. In a similar fashion, Morningstar et al. [2019] proposed a learned gradient scheme for correcting distorted images of gravitationally-lensed galaxies and showed that it was significantly more accurate than linear inversion.

Learned regularisers

Another option is to use ML to learn a regularisation strategy. Most iterative inversion algorithms use regularisers within their inner loop to help guide the algorithm towards a likely solution, although defining one by hand can be very challenging. Instead, ML can be used to learn one.

For example, Hammernik et al. [2018] proposed to learn key parameters of a fields of experts regulariser when using it within the loss function of the Landweber iterative algorithm to carry out magnetic resonance imaging (MRI). Specifically, they learned the filter kernels, activation functions and data term weights of the regulariser by unrolling the iterative algorithm over a fixed number of steps and training end-to-end using many example inversion problems. They found using this learned regulariser led to more accurate reconstructions than standard algorithms over a wide range of images.

Li et al. [2020a] took this idea a step further and used a neural network to define the regularisation term in the loss function when using iterative Tikhonov regularisation-based inversion algorithms, in a strategy they called network Tikhonov (NETT). The network was trained to estimate artefacts in the solution given the current solution as input, using a dataset of previous inversion runs. Lunz et al. [2018] used a similar strategy, but instead used an adversarial network to define the regulariser, which was trained to discriminate between ground truth and estimated solutions from previous inversion runs. Both works found their regularisers led to accurate inversion algorithms.

Finally, Bora et al. [2017] used generative models to define a regularisation strategy when using iterative inversion algorithms. Specifically, they trained a generative model (e.g. a VAE or GAN) to generate plausible solutions using a dataset of many example solutions and then used this generator to parameterise the solution in the inversion problem. This transformed the inversion problem from one of estimating the solution to one of estimating the latent values of the generator. They showed this approach outperformed standard baselines across a range of compressive sensing tasks. In a similar fashion, Mosser et al. [2018] showed this strategy could be successfully applied to seismic inversion tasks.

2.4 Summary

Innovation in the field of SciML is happening at a remarkable pace and a wide range of SciML approaches have been proposed across many different domains and scientific objectives. In this section we summarise these approaches and identify general trends, challenges and future opportunities for the field. Finally, we motivate our central research question (how well do PIML techniques scale to real-world problems?).

2.4.1 General trends

Whilst a plethora of different SciML approaches exist, some general trends emerge from our analysis. Firstly, and perhaps most importantly, incorporating scientific knowledge nearly always improves the performance of ML algorithms. The SciML approaches studied consistently outperform their naive ML counterparts in terms of accuracy, speed, interpretability and physical consistency. This is generally true across all the approaches, regardless of their application domain and scientific objective. Adding scientific knowledge restricts the space of models the ML algorithm can learn, introduces stronger inductive biases, and allows the ML algorithm to focus on only the parts of the scientific workflow that need to be improved, which in turn alleviates some of the well-known flaws of ML algorithms (and in particular, deep learning), including their poor generalisation, their difficulty of optimising, their lack of interpretability and the large amounts of training data required. This observation is encouraging

as it gives strong evidence in favour of the central thesis of the field, which is that incorporating scientific principles into ML leads to more powerful models.

Secondly, SciML approaches can be applied across a wide range of domains and scientific objectives. The approaches studied cover fields such as fluid mechanics, computational biology, seismology and particle physics, and they are applied to tasks ranging from the prediction of protein structures to the design of photonic devices. Whilst their applications differ, the core algorithmic goal of many of these approaches can be abstracted into one of the fundamental scientific tasks discussed in Section 2.2, and a host of SciML techniques are available for each, as summarised by Table 2.1. Some techniques are general-purpose (for example, incorporating underlying equations into the loss function [Raissi et al., 2019]), whilst some remain highly domain-specific (for example, using a curl representation to ensure incompressibility in fluid mechanics [Mohan et al., 2020]). It is an open and interesting question to whether the most powerful SciML techniques will end up being domain-specific or general-purpose; currently both approaches are powerful.

Thirdly, scientific knowledge can be incorporated across the entire ML pipeline. It can be added through careful choice of training data, the architecture of the algorithm, the way the algorithm is trained, and by incorporating the algorithm more tightly into an existing scientific workflow. Furthermore, as summarised by Figure 2.1, these strategies vary significantly in how strongly they impose scientific knowledge. Generally, approaches which modify the loss function assert the weakest constraints, whilst hybrid approaches assert the strongest. Similar to above, it is an open and interesting question to whether SciML will converge on a certain strategy; currently it appears the most appropriate highly depends on the scientific task at hand. We also note it is possible to use multiple strategies at once, for example Jiang et al. [2020] who combined a physics-informed loss function with residual modelling.

Finally, it is clear that SciML can significantly enhance the performance of traditional workflows. In some cases, SciML dramatically improves the efficiency and accuracy of traditional algorithms, and adds new scientific insights. SciML appears most useful where the underlying processes are not fully understood, or where the

traditional approach is computationally expensive, or where we are unsure of the best solution strategy. In such cases, we can take advantage of the large amounts of data we have to learn solutions to these tasks.

2.4.2 Future challenges and opportunities

Even though a plethora of SciML techniques have been proposed, the field is burgeoning and many opportunities exist. For example, many of the approaches above have only been applied to a restricted number of domains and are likely to strongly benefit others. There are sections in Table 2.1 which appear relatively unexplored, such the use of architectural constraints when carrying out inversion or equation discovery, which may lead to novel approaches. Combining multiple strategies for incorporating scientific principles at once is also relatively unexplored and could lead to more powerful models. Alongside SciML, the field of ML is constantly generating new ideas which could be incorporated into SciML.

There are challenges too. For example, the field lacks clear benchmarking and comparison across these approaches, and future research is required to recommend which SciML approaches perform best for a given problem. Even though SciML helps to improve the interpretability of ML models, for the most part they still remain challenging to interpret and understanding how best to balance this lack of interpretability with the clarity of traditional models is an open question. Furthermore, in contrast to traditional approaches, many of the SciML approaches above lack theoretical grounding, yet theoretical guarantees may be crucial for safety-critical or resource-limited applications. Studying these properties could lead to better performance and new insights.

Finally, whilst most SciML research is primarily focused on helping the sciences, a growing amount of research is being carried out in the other direction, using scientific insights to improve our understanding of ML [Ruthotto and Haber, 2020, Chaudhari et al., 2018, Chang et al., 2019, Wright et al., 2022, Hughes et al., 2019]. This work "closes the loop" and will hopefully accelerate research and cross-fertilisation in both fields.

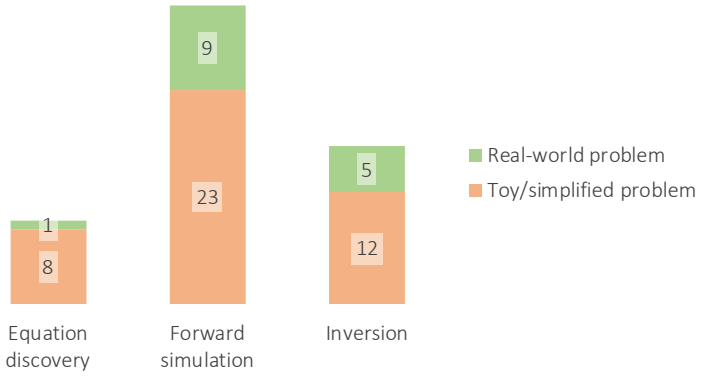


Figure 2.5: Number of works studied in this chapter which focus on solving real-world versus toy/simplified problems, broken down by scientific task. Please see the main text for a description of how these categories were assigned.

2.4.3 Scaling to real-world problems

Alongside the above, and as discussed in Chapter 1, a major outstanding challenge is to understand how well SciML approaches scale to more complex, real-world problems. To highlight this, Figure 2.5 plots the number of works studied in this chapter which focus on solving real-world problems compared to the number which focus on solving toy/simplified problems, broken down by scientific task. This assignment is fairly subjective, and for this plot we define a real-world problem to be one which attempts to closely resemble the real-world and thus has tangible impact. For example, for a work which carries out forward simulation, a real-world problem would be carrying out modelling of a complex, real-world environment (for example, a multi-scale, multi-physics global climate simulation), and for a work which carries out inversion or equation discovery, a real-world problem would be inverting for underlying parameters using real data captured from a real-world environment (for example, carrying out MRI using real clinical data). A work considered in the real-world category does not necessarily have to offer an effective solution to such a problem, but it must at least place significant focus on the real-world problem and assess how well its approach scales.

Under this definition, we find that most works focus on the latter, using only simplified problems to validate their SciML approaches. Whilst this useful, given

the increasing complexity and scale of the scientific problems we wish to study, it is clear that a crucial next step is to understand how well these approaches scale to more complex, real-world problems. This is essential for them to become widely applicable and have significant impact.

As presented in Section 1.2, understanding how well SciML (and in particular, PIML) approaches scale is the central question of this thesis. In the next chapter we will address our first sub-question on this topic, which is, can PIML discover underlying physical principles from real data?

3

Physically-interpretable unsupervised learning of lunar thermodynamics

3.1 Introduction

In this chapter we investigate the question, can PIML algorithms discover underlying physical principles from real data? In particular, we assess whether a variational autoencoder (VAE) is able to learn about the underlying physical processes affecting the lunar surface temperature from real temperature measurements captured by the Diviner Lunar Radiometer Experiment on board the Lunar Reconnaissance Orbiter (LRO) satellite.

The LRO has been orbiting around the Moon since 2009 and its main goal is to map as much of the Moon as possible in order to support future human and robotic missions there. Of particular interest to this chapter is the data recorded by LRO's Diviner instrument, which has been recording the Moon's surface temperature since the LRO's launch. This dataset contains rich information about the Moon's thermophysical environment and understanding the underlying processes contained within it is of key interest to scientists and mission planners alike.

Aside from its use in lunar science, this dataset is a good candidate for assessing how well PIML techniques for discovering physical principles scale to real-world problems, for a number of reasons. Firstly, the thermal behaviour of the lunar

surface is complex and therefore many different physical processes are contained within the dataset. Secondly, its measurements contain real-world noise which is more complex in character than e.g. synthetic Gaussian noise. Thirdly, over its 12 year lifetime the Diviner instrument has captured billions of temperature measurements, providing significant amounts of data to learn from. Thus, learning from this dataset presents significant differences to learning from a synthetic dataset generated using a simplified physical system.

In this chapter we specifically assess whether a VAE is able to learn about the underlying physical processes in this dataset. The VAE defines a fully probabilistic and data-driven model of the lunar surface temperature. The model is physics-agnostic and only makes broad assumptions about the physical system; the most important being that at each point on the surface the observed temperature variations can be described by a small set of independent latent variables. The VAE learns the relationship between these latent variables and the observed temperature variations by learning to reconstruct the dataset. To cope with the dataset's noise and irregularity a Gaussian process is used to interpolate the raw surface temperature measurements onto a regular grid before using them to train the VAE.

Once trained, we interpret the model learned by the VAE by comparing its latent variables to underlying physical parameters estimated during traditional inversion using an existing thermophysical model of the lunar surface. By doing so, we find that the VAE is able to disentangle five different thermophysical processes from the data, including 1) the solar thermal onset delay caused by slope aspect, 2) effective albedo, 3) surface thermal conductivity, 4) topography and cumulative illumination, and 5) extreme thermal anomalies. Furthermore, compared to traditional inversion, the VAE is extremely efficient, requiring orders of magnitude less computational power to invert for underlying physical parameters. This allows us to generate global thermal anomaly maps of the lunar surface.

Whilst the VAE is effective at learning about the underlying physical processes in this dataset, multiple challenges are uncovered. Firstly, because the VAE is fully data-driven, it is difficult to interpret its latent space. Carrying out traditional physical

modelling alongside the VAE is symbiotic in this sense; it both helps us to interpret the VAE and increases our confidence in the existing physical model. Secondly, the VAE struggles to fully disentangle all of the underlying physical processes contained in the dataset, and this is discussed further. Finally, our workflow is only very weakly physics-informed in the sense that a traditional physical model is used to interpret the latent space of the VAE. Incorporating physical principles directly into the design of the VAE could bring additional advantages and we discuss this further.

3.2 Background

3.2.1 Understanding the lunar surface

The lunar surface is a fascinating environment to study. It is the result of the Moon's turbulent geologic history and allows us to understand many of the endogenic and exogenic processes that have shaped the Moon over billions of years. It also allows us to understand the evolution of our Sun and the solar system [Spudis and D., 1996, Fagents et al., 2010, Joy et al., 2016]. For example, the spatial distribution and heterogeneity of surface characteristics, such as composition and relief, can be used to infer some of the fundamental processes that control the evolution of the Earth - Moon system. Famously, the frequency of occurrence and diameters of impact craters can be used to estimate the age of lunar regions as well as of planetary surfaces all across the solar system (e.g. Shoemaker [1962], Baldwin [1964], Hartmann [1965]).

As well as informing science, studying the lunar surface is essential for the continued exploration of the Moon. For example, maps of the surface temperature (e.g. Vasavada et al. [2012], Williams et al. [2017]) and rock abundance [Bandfield et al., 2011] are vital for planning future crewed and uncrewed missions. Furthermore, with NASA's goal of establishing a permanent and sustainable human presence on the lunar surface [NASA, 2020], maps of in-situ resources such as oxygen, water, rare Earth elements, metals and solar power are essential [ESA, 2019].

However, because of its remoteness, studying the lunar surface can be challenging. Remote sensing missions provide the main source of data, but such data can be noisy, sparse, incomplete, and require careful calibration. Furthermore, many of

the scientific products derived from this data rely on a physical model to interpret the data and invert for underlying physical quantities of interest. These models are typically either theoretically derived or have been developed during extensive laboratory studies, but they can be difficult to verify; humans have only been to the Moon a small number of times and we therefore lack ground truth observations. This can result in bias and incorrect interpretation. Finally, increasingly large amounts of data are being returned by space exploration missions, and it is becoming challenging to absorb and analyse all of it.

An example remote sensing mission which is of particular interest to this chapter is the Diviner Lunar Radiometer Experiment (DLRE) or Diviner instrument on board the LRO which has been acquiring visible and infrared radiance measurements of the lunar surface since 2009 [Paige et al., 2010, Williams et al., 2017, Mazarico et al., 2018]. Its main goal is to study the Moon's thermophysical environment and over its history it has revealed many complex thermal phenomena. It was the first instrument to create high-resolution ($\sim$ 150 m spatial resolution) day and night surface temperature maps of the Moon, which revealed significant variations in temperature which depend on location and time of lunar day [Williams et al., 2017]. It characterised rock abundance [Bandfield et al., 2011], discovered the presence of anomalous lunar cold spots, which are extensive regions of low thermal inertia, highly insulating surface material that usually surround small, recent impact events [Bandfield et al., 2014], and discovered craters and other regions with anomalously high thermal conductivity, which could suggest the presence of metals on or close to the surface [Hayne et al., 2017]. The instrument has also been used to study the temperature of permanently shadowed regions, which are among the coldest places in the solar system [Sefton-Nash et al., 2019], and the mineralogy of the Moon's surface [Greenhagen et al., 2010]. However, as noted above, many of these studies rely upon traditional physical modelling and/or laboratory studies to analyse and interpret the raw temperature measurements recorded by the instrument, which may introduce biases.

One potential solution is to place the focus on learning from data, rather than relying on traditional model-based analysis. This could allow weaker assumptions to

be made about the data and could therefore result in new underlying processes to be discovered. Furthermore, machine learning may be able to help absorb the increasing amounts of data by providing more automated and efficient inference algorithms.

3.2.2 Related work

Machine learning is starting to see increased use in lunar and planetary science. For example, Silburt et al. [2019] developed a deep learning-driven tool to detect and measure impact craters on the Moon and Mercury in digital elevation model data. Kodikara and McHenry [2020] showed that machine learning can be used to classify the physical and mineralogical properties of lunar regolith. Bickel et al. [2019, 2020] trained convolutional neural networks to automatically extract the locations and sizes of lunar rockfalls from LRO's image archive with more than 2 million entries and Wu et al. [2019] presented a neural network-driven approach to improve the localisation accuracy of rovers on planetary surfaces by matching ground-based and space-borne imagery. However, whilst these works concentrate on detection and classification, little work has been done on using ML to learn about and model the underlying physical processes on the Moon.

3.2.3 Instrument overview

The Diviner instrument collects radiance measurements across 9 different wavelength channels in the visible-infrared band (0.3 - 200 microns). Each channel is defined by a linear array of 21 thermopile sensors (i.e., the instrument has a total of 189 sensors) and due to the limited field of view (FOV) of the sensors, Diviner's swath width is very narrow, resulting in quasi-point measurements along LRO's ground track (nadir push broom mapping). Thus, multiple orbits are required to completely cover the entire surface of the Moon and the number and effective FOV of the available measurements governs the spatial and temporal resolution of Diviner's final image or map products. The dominant influence on the size of the effective FOV of each point measurement is the LRO's altitude, which has varied between $\sim 20 - 200$ km over Diviner's operational period; at 30 km altitude it is expected to be an ellipse approximately

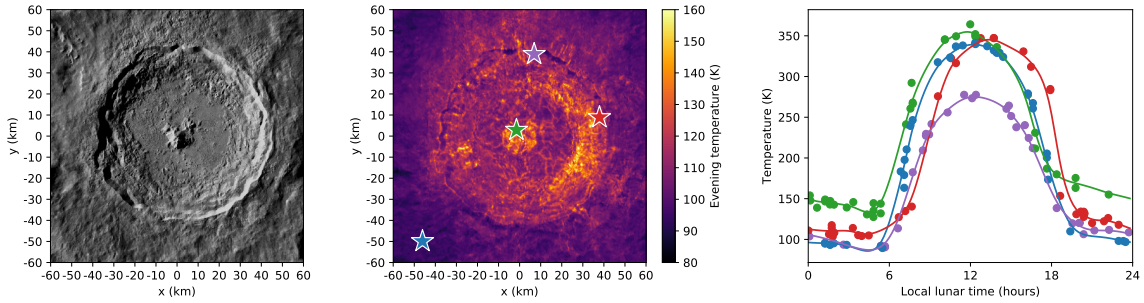


Figure 3.1: Example pre-processed data. Left: optical image of Tycho (LRO Wide Angle Camera (WAC) Global Morphologic basemap [Speyerer et al., 2011]) (348.7° lon./ -43.3° lat.). Middle: evening temperature map extracted from the pre-processed Diviner dataset, plotting the average temperature over 12 am-4 am in local lunar time. Right: four example temperature profiles extracted over Tycho. Points show temperature measurements, solid lines show the Gaussian process fits of the profiles used for interpolation. Colour coded stars in the middle plot indicate the locations of the profiles.

160 × 120 m in size with its major axis oriented in the in-track direction, whilst at 200 km altitude these dimensions are expected to double or triple [Sefton-Nash et al., 2017]. Since its launch, Diviner has obtained billions of point measurements of the lunar surface, allowing its temporal and spatial temperature variations to be studied.

3.3 Methods

3.3.1 Data pre-processing

There are multiple Diviner datasets at different levels of processing publicly available from NASA's Planetary Data System (PDS). For this study we collect the Diviner Level 1 Reduced Data Record (RDR) data [Paige et al., 2011], which consists of tables of calibrated radiance point measurements and their derived surface temperature values, as well as their associated ephemeris and geometry information, including the local lunar time at which each point measurement was recorded. We choose this dataset because it is minimally processed from the raw Diviner telemetry data and contains the point measurements before any gridding is applied.

We download data from Diviner's entire operational period January 2010 - March 2019, which contains data recorded during LRO's nominal science mission phase as well as its three extended science mission phases. We use all available data for a number of reasons. Firstly, because we are using a data-driven approach, we

want to absorb as much data as possible and do not wish to bias our analysis to a narrow observational window. Secondly, as noted above, including more orbits increases the spatial-temporal resolution of the data.

A number of pre-processing steps are carried out before inserting this data into our unsupervised learning workflow. Firstly, for each RDR data table, we only keep measurements which have instrument activity flags equal to 110 (indicating "on moon" orientation, standard nadir observation and "nominal" instrument mode), calibration flags equal to 0 (indicating an in-bounds interpolated measurement), geometry flags equal to 12 (indicating tracking data was used to generate geometry information), miscellaneous flags equal to 0 (indicating no miscellaneous observations) and emission angles less than 10° (angle between the vector from the surface FOV centre to Diviner and the "normal" vector to the Moon's surface). This step removes 23.4% of the point measurements available. Secondly, for simplicity, in this study we only use data from channel 7 which records radiance in the 25-50 micron band, is the most sensitive to temperature values in the range 69-178 K and has a high signal-to-noise ratio over the majority of surface temperatures [Williams et al., 2017, Vasavada et al., 2012, Feng et al., 2020]. Finally, we sort the data into $0.5° \times 0.5°$ latitude and longitude bins, which allows temperature values at each point in space to be easily extracted in our subsequent workflows. This pre-processing step takes considerable computational resources; 400 CPU cores over approximately 2 days are used to process over 425 billion point measurements contained in 35 TB of uncompressed Diviner RDR data. Example pre-processed data are shown in Figure 3.1.

3.3.2 Overview of VAE workflow

Our basic idea is to use a VAE to learn the underlying factors of variation in the Diviner surface temperature measurements. The input to the VAE is the observed surface temperature variations over the lunar day at each point on the surface and we learn a latent representation which is able to reconstruct this input. In doing so we learn a model of the thermophysical data, without the use of an a prior physics model. We choose a VAE framework (as specified in Equation A.3-A.6) because

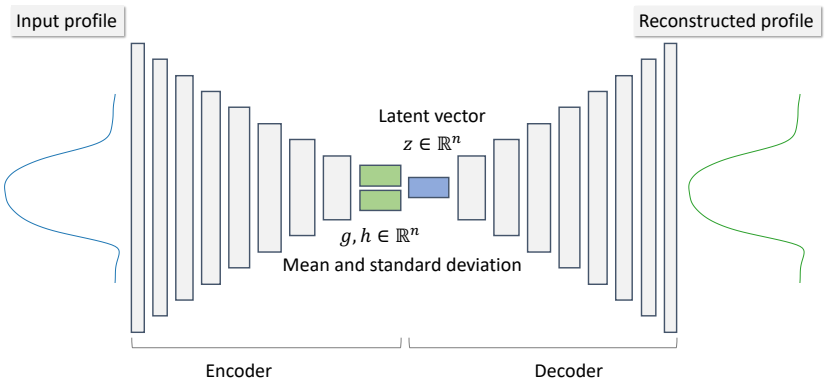


Figure 3.2: VAE workflow. We train a VAE to compress the observed surface temperature variations over the lunar day into a sparse set of latent values. The input to the VAE is the 1D profile of the surface temperature variation over the lunar day extracted at a single point on the lunar surface. A convolutional encoder computes the mean and standard deviation of the VAE's approximate latent posterior distribution, given these observations. During training a sample from this posterior distribution is fed into a convolutional decoder, which outputs a reconstruction of the input profile.

it attempts to learn a model which 1) accurately reconstructs the input profiles, 2) disentangles independent factors of variation in the surface temperature and 3) has a latent representation which varies smoothly over the input space. We stress that these are very general assertions and therefore this method could be applied to other similar space science and exploration tasks.

A major benefit of this approach is that it is fully data-driven. In general, methods which rely on physical models to analyse physical data may be unreliable if one is not confident on the accuracy of the model (e.g. Hayne et al. [2017]). Furthermore, we learn salient features directly from the data; other traditional approaches which uses hand-crafted features (such as averaged or evening temperature maps, e.g. Bandfield et al. [2011]) to interpret physical data risk discarding important information. We interpret the VAE by comparing its latent variables to parameters estimated using inversion with an existing thermophysical model. Once trained the VAE is run over the entire lunar surface to map its latent variables and these maps are used to search for thermal anomalies.

Our VAE workflow is shown in more detail in Figure 3.2. The input to the VAE is the 1D profile of the surface temperature variation over the lunar day extracted at a single point on the lunar surface. The VAE has multiple "encoder" convolutional

layers which reduce this profile into its latent representation and multiple "decoder" convolutional layers which reconstruct it, and is trained using example profiles from many different locations on the lunar surface.

3.3.3 Training data generation

To train the VAE, millions of example 1D profiles of the surface temperature variation over the lunar day are extracted from 48 Areas Of Interest (AOI). We curate the AOIs so that they encapsulate a wide range of surface temperature variation. The AOIs include impact craters of different ages, large pyroclastic deposits and swirls (magnetic anomalies), with areas ranging from 1 - 10,000 km^2. They are listed in Table A.1 and described in more detail in Appendix A.2. The locations of all the AOIs are shown in Figure A.1.

We extract the 1D temperature profiles by locally projecting and binning the observed Diviner surface temperature measurements at each AOI onto a 200×200 m grid and sorting the points in each bin by the local lunar time at which they were recorded, obtaining a profile at each bin location. An orthographic projection centred over each AOI with a radius of $R_{\text{moon}} = 1737.4 \times 10^3$ m is used to define each grid. To ensure sufficient sampling of each temperature profile, we reject profiles whose maximum spacing in local lunar time between points is less than 4 hours. We also reject some profiles from background areas to ensure they do not dominate the training set. This workflow results in 1,985,693 training profiles extracted over all AOIs and examples of extracted profiles are shown in Figure 3.1.

Before inputting the profiles into our VAE workflow, we interpolate their temperature measurements onto a regular grid by using Gaussian Process (GP) regression [Rasmussen and Williams, 2005]. A GP with a Matern-1.5 kernel, maximum length scale of 6 hours and an assumption of 10 K standard deviation of uncertainty in each temperature measurement is fit to each profile. We sample every 0.2 hours, resulting in a 1D input profile with 120 samples. Example GP interpolations are also shown for the profiles in Figure 3.1.

3.3.4 VAE architecture and training

A convolutional encoder-decoder design is used for our VAE architecture, shown in Figure 3.2, which consists of 9 convolutional layers in the encoder and 8 convolutional layers in the decoder. The encoder takes an input profile of dimensionality 120×1 and slowly reduces it to 2 output vectors each of dimensionality $1 \times n$ which represent the mean, g, and standard deviation, h, of the approximate latent posterior distribution in Equation A.5. The number of latent variables, n, is a hyperparameter of the VAE model which we test. The decoder takes a sample from the latent posterior distribution and slowly expands it to output a reconstruction of the input profile.

We use 32 hidden channels in the encoder and decoder and batch normalisation [Ioffe and Szegedy, 2015] after all hidden layers to help training converge. For $n = 4$ the total number of trainable parameters in the model is 28,713. A complete specification of the network is given in Table A.2 and the network architecture is described in more detail in Appendix A.2. We design the network architecture by hand to maximise its reconstruction performance.

During training, the mean and standard deviation vectors are used to generate a single sample (or vector of latent values) from the approximate posterior distribution which is fed into the decoder so that the expectation term in the loss function (Equation A.6) can be estimated. During inference, only the mean vector is passed to the decoder and used to define the latent values. We use a β-value of 0.2 and normalise the training profiles to have a collective mean of 0 and standard deviation of 1 before inputting them to the VAE. The network is trained using the Adam stochastic gradient descent optimisation algorithm [Kingma and Ba, 2015], using an exponentially decaying learning rate starting at $1\text{x}10^{-3}$ and a mini-batch size of 200 profiles, reserving 20% of the example dataset for cross validation. We use the PyTorch framework [Paszke et al., 2019] to define and train the VAE.

3.3.5 Physics modelling and interpretation

A downside of our approach is that the VAE model lacks a clear physical interpretation. To aid its interpretation, we compare the VAE's latent representation to a physics-

based inversion. We modify the thermophysical model proposed by Hayne et al. [2017] which consists of numerically solving the heat equation over time t and depth z in a one-dimensional solid medium

$$\rho(z,t)\, C_p(z,t) \frac{\partial T(z,t)}{\partial t} = \frac{\partial}{\partial z}\left(K(z,t) \frac{\partial T(z,t)}{\partial z} \right) , \tag{3.1}$$

where $\rho(z,t)$ is the mass density, $C_p(z,t)$ is the specific heat capacity, $T(z,t)$ is the temperature and $K(z,t)$ is the thermal conductivity. The boundary conditions and functions $\rho(z,t)$, $C_p(z,t)$ and $K(z,t)$ must be known in order to solve this differential equation for the temperature profile. We make a few important alterations to the model used by Hayne et al. [2017]. First, we assume that the density ρ is constant and equal to the surface density ρ_s given by Hayne et al. [2017]. Second, we use the phonon conductivity K_c as a free fitting parameter, rescaled with respect to the surface conductivity K_s given by Hayne et al. [2017]. For this purpose, we introduce the thermal conductivity factor as K_c/K_s. Third, we assume that the solar incidence angle approximately coincides with the hour angle and absorb the effect of latitude and slope on solar incidence in an effective Bond albedo, which we use as another free fitting parameter. Hence, our model treats the lunar regolith as a solid medium with a constant density, but with a varying effective albedo and thermal conductivity factor.

We run inversion using this model by tuning the thermal conductivity factor, albedo and onset time of solar radiation to best match each input interpolated profile. Note the model predicts the temperature as a function of depth and time, $T(z,t)$, and so the temperature at the surface is used when matching to the input profile. Bayesian optimisation with an expected improvement acquisition function and a L2 loss function is used as the inversion algorithm [Frazier, 2018]. We then compare the estimated parameters from the inversion to the latent variables generated by the VAE. Note, in contrast to this model, the VAE only models the temporal dependence and not the depth dependence of the surface temperature.

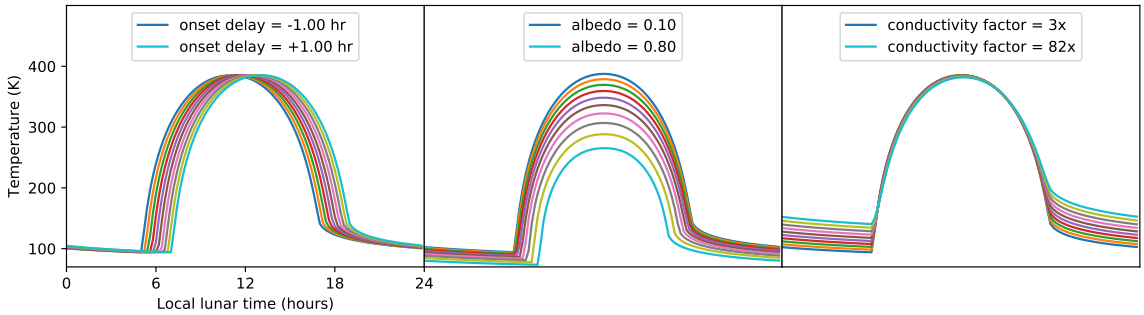


Figure 3.3: 1D numerical simulations of the lunar surface temperature against local lunar time. Our baseline model uses thermal properties estimated for the lunar regolith. The three plots show the sensitivity of the baseline model when varying the onset delay, albedo and thermal conductivity factor independently.

3.3.6 Generation of global maps

As a final step, we run the trained VAE with $n = 4$ over the entire lunar surface and generate maps of its latent variables to search for thermal anomalies. This allows us to compute global statistics of the learned latent variables and interpret these new features alongside existing physical maps. The details on how we generate these maps are provided in Appendix A.3.

3.4 Results

3.4.1 Data pre-processing

In total 425,745,195,120 Diviner point measurements are extracted, of which 326,190,541,311 (77%) remain after the first pre-processing filtering step and 36,483,372,924 remain after filtering channel 7, requiring 2.9 TB of storage. We plot the number of points after pre-processing in each latitude-longitude bin in Figure A.4. There is a higher point density closer to the poles, which is expected because LRO's orbit means the Diviner instrument samples the poles more frequently than the equator.

Example data after pre-processing over the Tycho crater are plotted in Figure 3.1. We observe a wide variation in temperature around this crater. Outside of the crater, the surface temperature increases to around 350 K at midday and decreases to 100 K at night. On the Eastern slope of the crater the profiles are shifted in local time by approximately 2 lunar hours, which is consistent with the slope's shadow

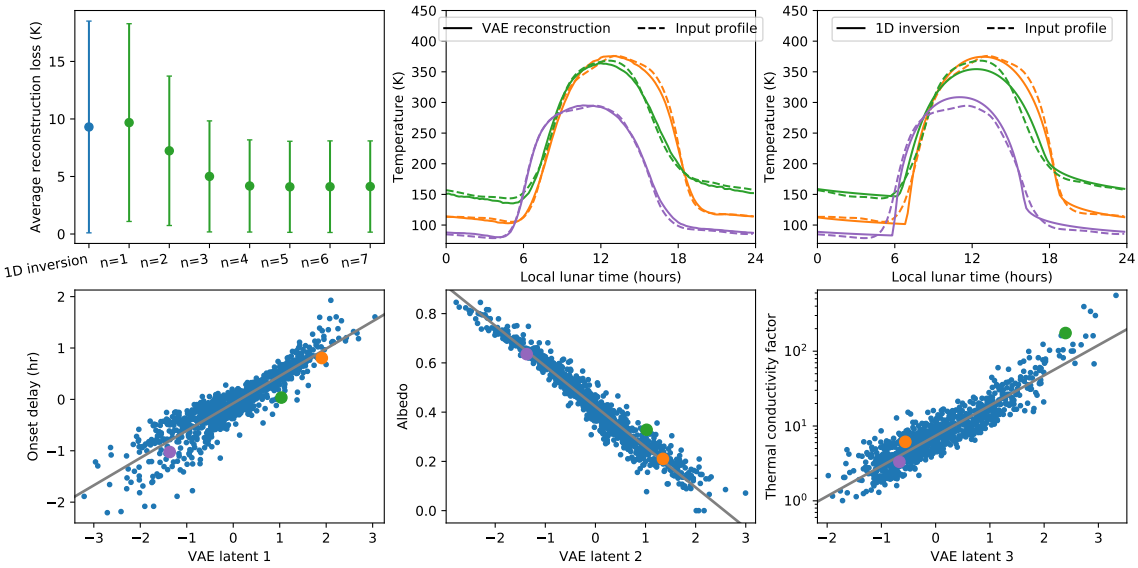


Figure 3.4: VAE results and physical interpretation. Top left: Average L1 reconstruction loss of VAEs with varying latent vector lengths n and the 1D physics inversion, over 1000 profiles from the validation set. Error bars show ± 1 standard deviation. Top middle and right: Example reconstructions for the VAE with $n = 4$ and the 1D physics inversion, for 3 selected profiles in the validation set. Bottom: plots showing the correlation between the first 3 VAE latent values when $n = 4$ and the estimated parameters from physics inversion, over 1000 profiles from the validation set. Thermal conductivity factor is relative to the regolith. The larger coloured-coded points correspond to the profiles shown in the top middle and top right plots.

delaying the incidence time of the Sun's radiation. On the Northern slope the peak temperature is lower at 270 K, which is consistent with the slope receiving less thermal radiation per unit area from the Sun's elevation at Tycho's latitude. In the centre of the crater we observe a profile which has a higher evening temperature around 150 K. Figure 3.3 (right) plots our physics model simulation when varying thermal conductivity factor; one explanation is that the material at Tycho's center has a higher thermal conductivity than its surroundings, although it should be noted that its central peak has complex topography and surface roughness which is likely to influence the measured temperature profiles.

3.4.2 VAE results and physical interpretation

Whilst training the VAE the training and validation reconstruction loss values converge to similar values, suggesting the VAE generalises well and does not over-fit to the training dataset. Training takes approximately 1 hr using one NVIDIA

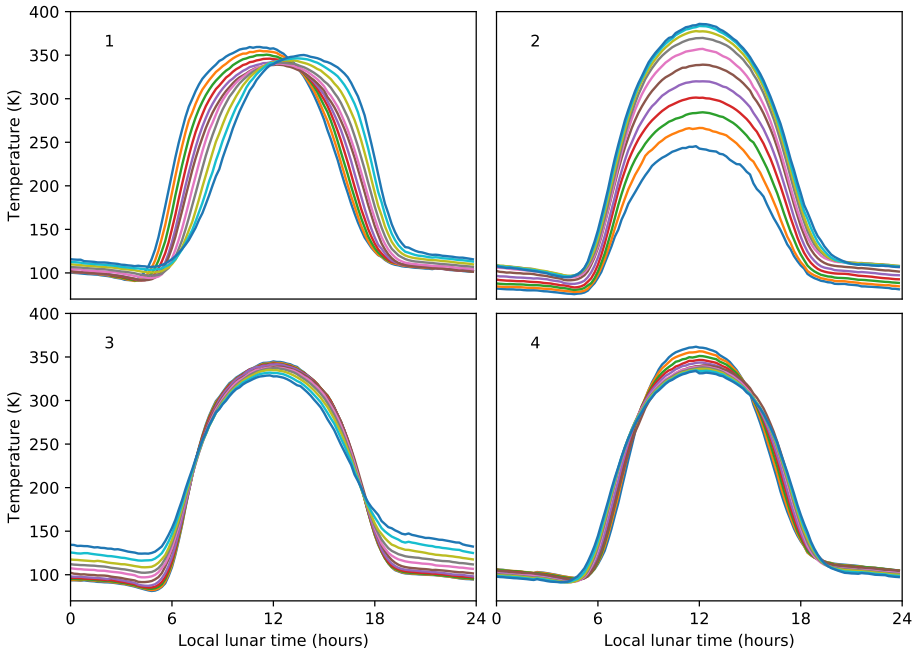


Figure 3.5: Reconstructed profiles generated from the VAE with $n = 4$ when sampling each latent variable independently across its range and fixing the other latent variables to zero.

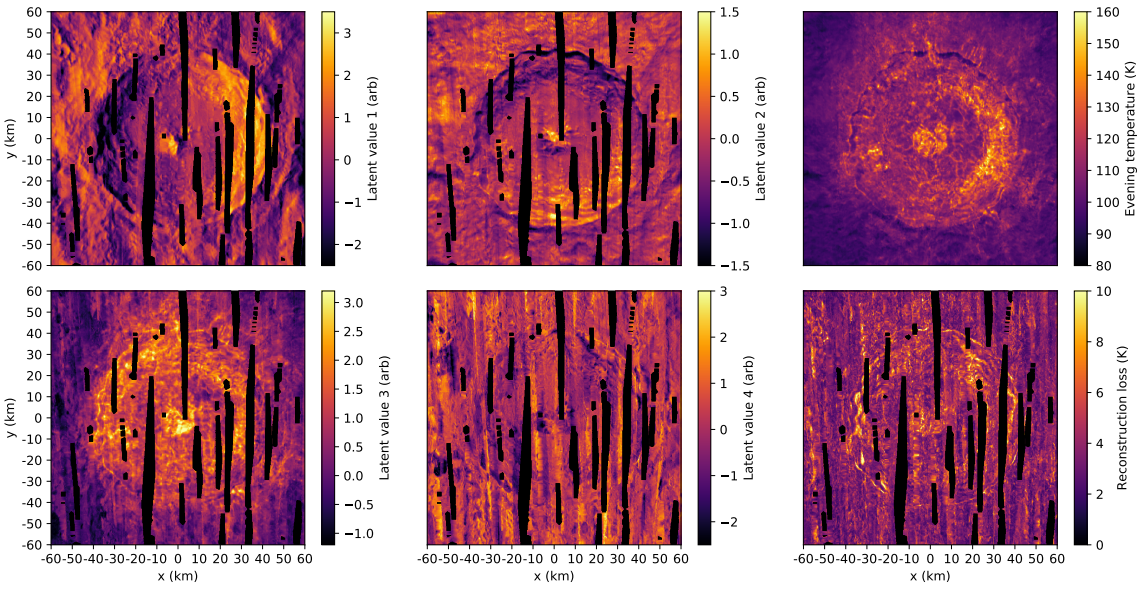


Figure 3.6: Maps of the VAE latent variables over Tycho, using the VAE with $n = 4$. Left two columns show maps of the four VAE latent variables, without any transforms applied. Top right is a repeat of the Diviner evening temperature map shown in Figure 3.1. Bottom right shows a map of the L1 reconstruction loss of the VAE.

Tesla T4 GPU. Example reconstructed profiles from the validation dataset after training are plotted in Figure 3.4 (top middle). We find that the VAE is able to

accurately reconstruct the profiles from the validation dataset. Figure 3.4 (top left) shows the average reconstruction loss over the validation dataset after training the VAE when varying the number of latent variables. The reconstruction loss asymptotes at $n = 4$, suggesting that only 4 latent variables are needed to accurately represent the input profiles.

Figure 3.5 shows example profiles generated by the VAE when varying each latent variable independently and fixing the others to zero, for $n = 4$. We observe that latent variable 1 responds to the onset of the profile, latent 2 responds to the peak temperature, latent 3 responds to the evening temperature "sidelobe" and latent 4 responds to the width of the temperature profile. To aid our interpretation of the VAE further we compare its latent variables with estimated parameters from the physics inversion. The VAE and physics inversion are run over the validation dataset and the latent values are plotted against the estimated physics parameters in Figure 3.4 (bottom). We find that latent variables 1, 2 and 3 have a strong linear correlation with the onset of solar radiation, effective albedo and the logarithm of the thermal conductivity factor respectively. This correlation can also be seen in Figure 3.3; scanning the physics simulation across the physical parameters result in similar profile variations to those seen when scanning the latent variables in Figure 3.5.

A straight line fit is shown for each correlation plot in Figure 3.4 (bottom). As thermal inertia is proportional to the square root of thermal conductivity, we expect that applying the transform $\hat{I} = e^{mz_3/2}$ to the latent 3 values will obtain a quantity which is linearly correlated with the thermal inertia at constant density and temperature, under the assumptions of our physics model. Here m is the gradient of the latent 3 and log thermal conductivity factor correlation and is estimated to be 0.93.

Maps of the latent variables and the VAE L1 reconstruction loss over Tycho are plotted in Figure 3.6. The VAE takes approximately 0.001 s to compute latent values for each profile when running on a single CPU, whilst each physics inversion takes of the order of 60 s to run using the same CPU. This significant efficiency improvement (factor of 60,000 or 4 orders of magnitude) allows the VAE to be run over large areas.

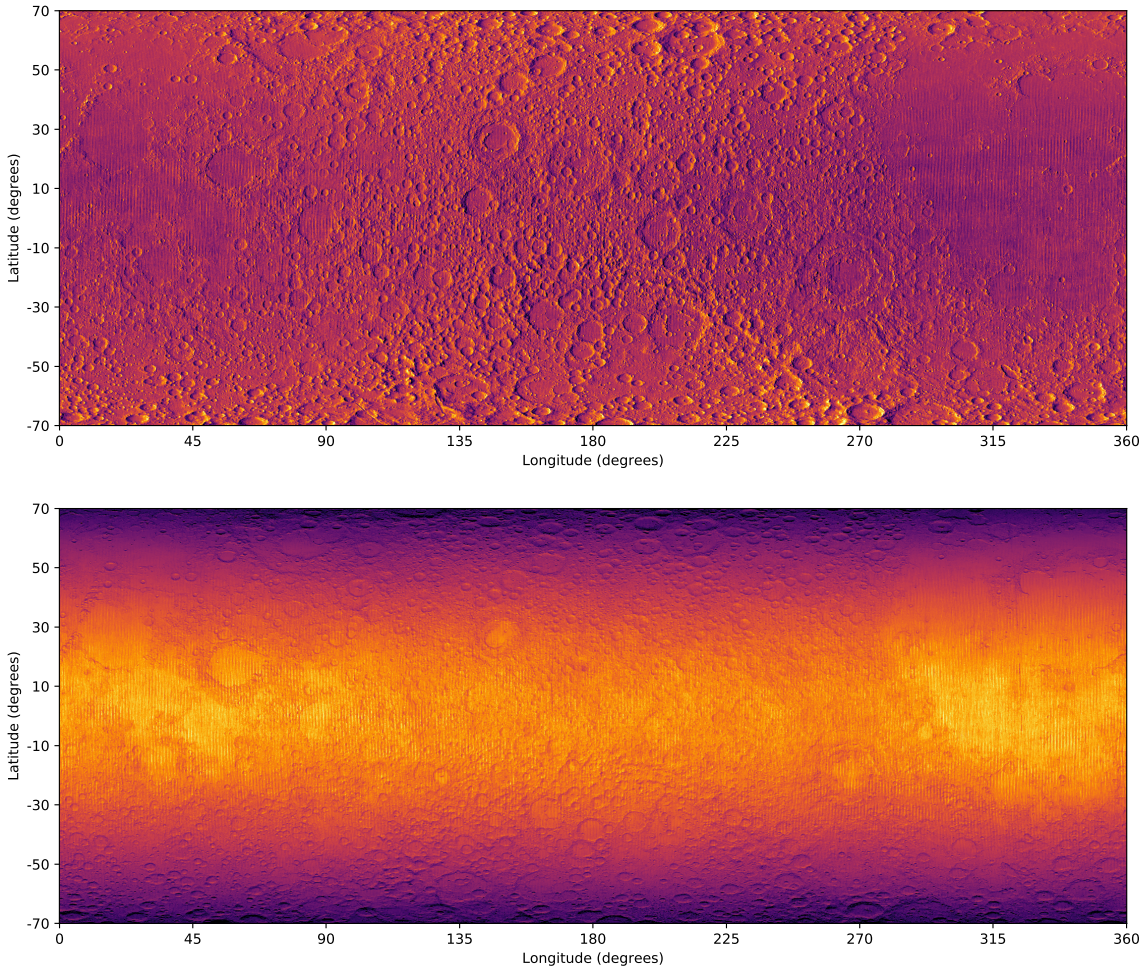


Figure 3.7: Global maps of the VAE latent variables 1 and 2. Top map shows latent variable 1 and bottom shows latent variable 2.

3.4.3 Global maps

The global maps produced by our mapping workflow are shown in Figures 3.7 and 3.8. We find that our observations of the latent variables are consistent on a global scale; the latent 1 map consistently shows higher values on the Eastern sides of craters and lower values on their Western sides, indicating that latent 1 responds to the onset delay of solar radiation as function of slope (East-West) aspect. The latent 2 map shows higher values at the equator and lower values closer to the poles, suggesting that it represents a combination of effective incidence angle and surface albedo. The latent 3 map mostly shows a background value with some craters and recent impact basins being either anomalously bright (e.g. Tycho crater) or showing a brightness level slightly above the background (e.g. Mare Orientale), and could

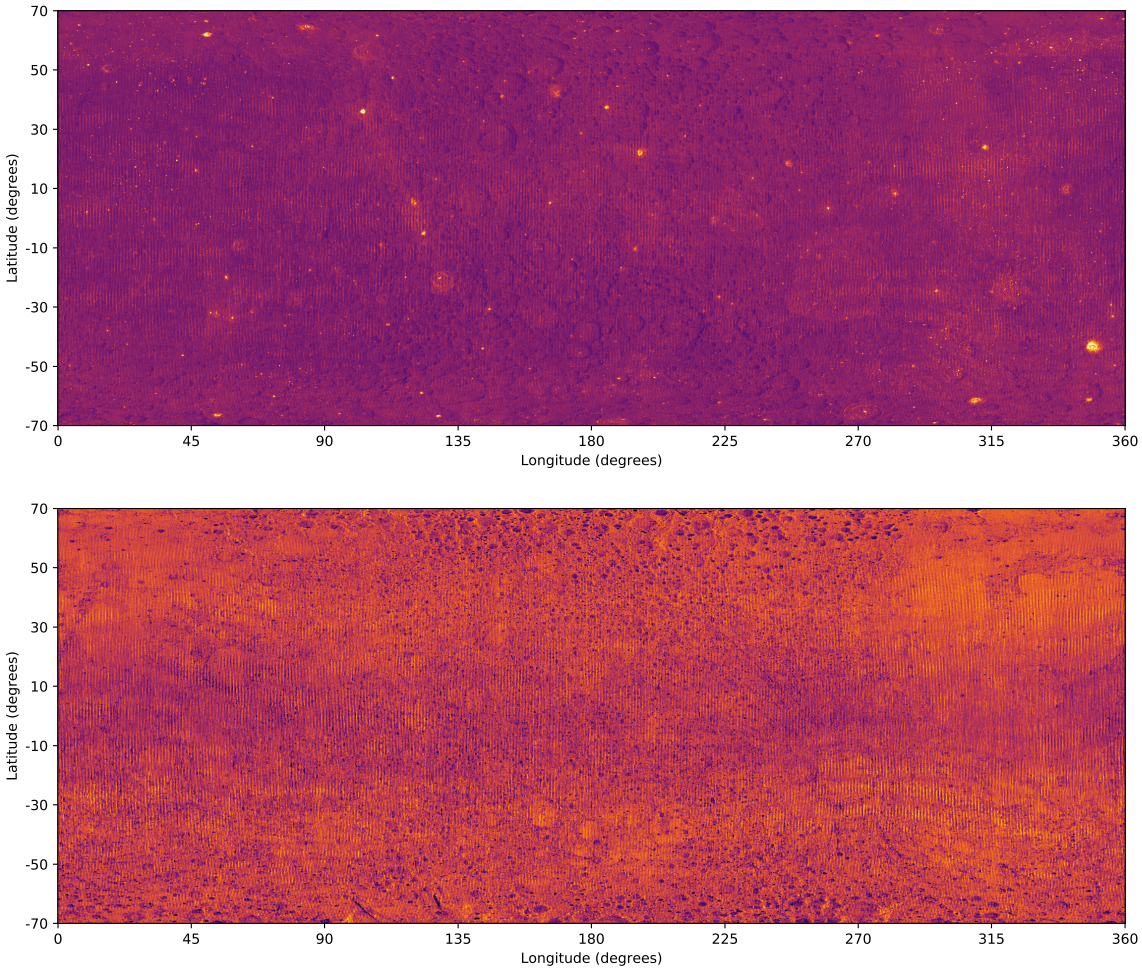


Figure 3.8: Global maps of the VAE latent variables 3 and 4. Top map shows latent variable 3 with the thermal inertia transform ($\hat{I} = e^{0.93 z_3/2}$) applied and bottom shows latent variable 4.

plausibly indicate changes in thermal conductivity. The latent 4 map is generally darker within smaller and narrower craters, particularly towards the poles. Besides their dominant trends, both the latent 1 and 2 maps show minor, local variations in the equatorial regions, for example around the main maria (centred at 315° lon./ 10° lat. or 45° lon./ 0° lat.), that could represent the influence of terrain roughness on latent 1 and varying albedo on latent 2. We define an anomalously high value of the latent 3 variable to be one that is greater than 2 standard deviations from its global mean and find that approximately 4% of the surface area of the Moon (within -70° to 70° latitude) contains these values.

The global map of the VAE L1 reconstruction loss is shown in Figure 3.9. The

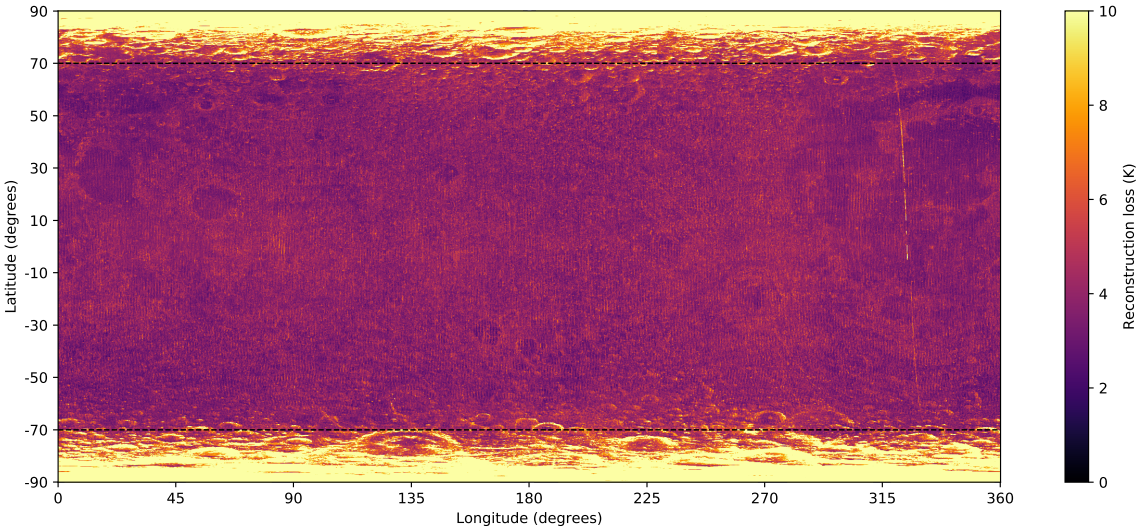


Figure 3.9: Global map of the VAE L1 reconstruction loss. Dashed lines show the -70° to 70° latitude cutoff used when generating the latent maps shown in Figures 3.7 and 3.8.

loss is low (typically less than 10 K) in the range -70° to 70° latitude, whilst closer to the poles it increases dramatically. In general, the loss slightly increases along topographic gradients such as the rims of craters and in the more heavily-cratered highland regions. Locally, the loss increases at East-West facing slopes of some impact features such as the Chaucer or Rydberg craters. A regional exemption is the north-western and central portion of Mare Smythii (85° lon./ 0° lat.) which show higher than average loss values. Some observations in the loss map are discussed in more detail in Appendix A.3.

Approximately 80% of the surface area of the Moon is covered by our extraction workflow, the missing data being where the extracted profiles do not meet our minimum temporal sampling criteria, which is mostly around the equator where the density of point measurements is lowest due to LRO's near-polar orbit.

3.5 Discussion

To analyse the latent representation of the VAE further, we compare its latent 3 values to the H-parameter values estimated by Hayne et al. [2017] at 4 different crater locations in Figure 3.10. The H-parameter has a one-to-one mapping with

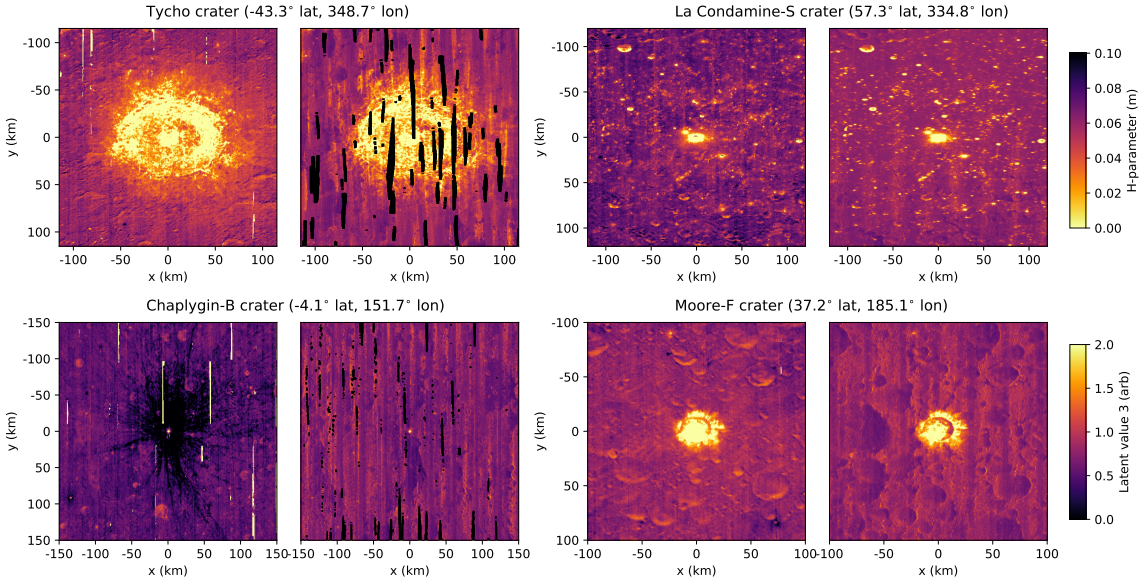


Figure 3.10: Comparison of the H-parameter map generated by Hayne et al. [2017] to our latent 3 variable map at 4 example crater locations. For each crater, left shows the H-parameter map and right shows the latent 3 variable map. The latent 3 variable map has the thermal inertia transform applied. Top left: high thermal inertia at the Tycho crater, Bottom left: Cold spot around Chaplygin-B crater unrecognised by latent 3, Top right: latent 3 slightly less affected by noise, Bottom right: H-parameter and latent 3 exhibit North/South and East/West coherent noise respectively.

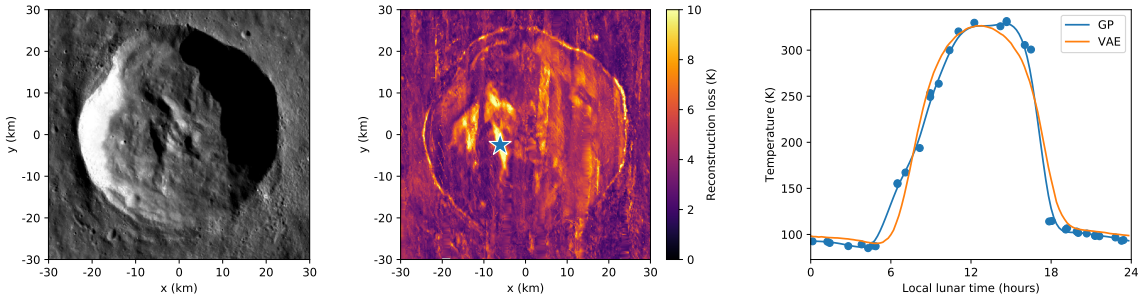


Figure 3.11: Example anomaly in our VAE reconstruction loss map, located within the Rydberg crater (263.6° lon./ -46.5° lat.). Left shows an optical image of the crater (LRO WAC Global Morphologic basemap [Speyerer et al., 2011]), middle shows a map of the L1 reconstruction loss of the VAE and right shows an example interpolated profile and its resulting VAE reconstruction extracted at the star location in the loss map. The loss anomaly is most likely caused by geometric errors, here (predominantly) the East-West facing slope of Rydberg's central peak.

thermal inertia at a constant temperature and is estimated using a thermophysical model where it governs the assumed density variation with depth.

There are multiple similarities and differences between the latent 3 map and the H-parameter map at each crater. At Tycho, the maps are largely similar within

the crater, suggesting a strong thermal inertia anomaly. At Chaplygin-B, there is a "cold spot" in the H-parameter map and no apparent anomaly in the latent 3 map. Bandfield et al. [2014] define cold spots as extensive regions of low thermal inertia, highly insulating surface material that usually surround small, recent impact events. This appears to be a limitation of the VAE, which struggles to differentiate between the subtler colder temperatures which produce these types of anomalies. At both Tycho and Chaplygin-B there is an acquisition footprint in the latent 3 map (vertically striped lines), which is likely a result of the differences in the point density of each LRO orbit affecting the GP interpolation and VAE reconstruction of each temperature profile. At La Condamine-S the latent 3 map is in strong agreement with the H-parameter map, and arguably has a lower noise content; this may be because the higher density of point measurements at higher latitudes makes the VAE inference more stable.

At Moore-F, we notice that both the H-parameter map and latent 3 map show coherent noise outside of the crater. Whilst the H-parameter map appears to erroneously correlate to the North-South slope, the latent 3 map correlates to the East-West slope. Looking carefully at the sensitivity of our latent variables in Figure 3.5 and comparing them to the physical model sensitivity in Figure 3.3, we notice that the VAE latent representation does not manage to fully disentangle the effect of solar onset with thermal conductivity factor and albedo, which could explain this observation. This may be a result of the fundamental assumptions of the VAE model (for example the Gaussian constraints on the posterior and prior latent distributions), or hint at some other physical process(es) affecting the latent representation, and further research is required to understand this effect.

As shown in Figure 3.7 the latent 1 variable is sensitive to the onset of solar radiation and thus depends on the aspect of the terrain, where a West-facing slope would heat up earlier in the day, whilst an East-facing slope would heat up later but also cool down later (resulting in a shifted thermal profile). The thermal behaviour of North and South-facing slopes is represented by the latent 2 variable, which is sensitive to the peak temperature or effective albedo of the lunar surface. Closer to

the equator, its behaviour is governed by the albedo of the surface, whereas closer to the poles it is increasingly controlled by topography as the angle between the incoming solar illumination and the surface increases. Interestingly, both the latent 1 and 2 map feature local variations close to the equator which indicates different physical properties of the surface in these regions (such as surface roughness), and is particularly evident when comparing equatorial mare and highland regions. Such local variations are less abundant in the latent 3 map, highlighting how it is largely decoupled from geometric artefacts caused by illumination and topography.

The latent 4 map shown in Figure 3.8 is generally darkest inside small and narrow craters and brighter for flatter regions. One possible explanation is that it is sensitive to cumulative illumination, i.e., to the relief or a large-baseline surface roughness, as a narrow crater would be illuminated for a shorter period of time than a flat mare region. Another explanation is that it is sensitive to a dependence of solar incidence angle on albedo, as modelled by Vasavada et al. [2012]. As observed in the latent 1 and 2 maps, the latent 4 map also exhibits minor local variations; in particular it has generally lower values close to the equator. This appears counter-intuitive, as the equatorial region would receive more direct sunlight on average; this variable seems to be sensitive to and is not fully decoupled from the effective albedo, to some extent.

The loss map generated by the VAE shown in Figure 3.9 indicates the input profiles which the VAE has struggled to reconstruct. These profiles are likely to be outside of the VAE's training distribution and the representative set of temperature variations observed on the lunar surface, and therefore this map is a useful resource because it can be used to search for extreme thermal anomalies. We show an example of a high loss anomaly within the Rydberg crater in Figure 3.11, along with a temperature profile extracted from within this anomaly. We find that the profile has a very gradual heating in the morning which the VAE is unable to model. There are multiple explanations for this profile; it could likely be a result of complex topology/geometry being integrated over Diviner's FOV and our 200×200 m bin size, or from extreme underlying thermal parameters. We observe another regional loss anomaly in Mare Smythii, on the lunar nearside. The reason for the slightly

above-average loss values in this region is unknown, but could represent anomalous thermophysical behaviour. Much more analysis could be done with this loss map.

There are many possible ways to extend this work. Firstly, it could be powerful to more tightly combine traditional physical models and the VAE together, to combine their relative strengths. For example, one could manipulate the latent representation of an input profile before reconstructing it to remove unwanted factors of variation, such as effective albedo, and then use an existing thermophysical model to invert for underlying thermal parameters. This may enable a simpler physical model to be used and combine the strengths of both approaches. Traditional thermophysical models (such as Bandfield et al. [2011] and Hayne et al. [2017]) either require terrain modelling to remove the effect of slopes and isolate thermophysical effects, or restrict themselves to low slope angles [Feng et al., 2020]. Our model is able to remove the effect of slopes very efficiently without the need for terrain modelling. Going the other way and using existing derived Diviner products, such as soil temperature, instead of the raw temperature measurements could help the VAE disentangle different thermophysical effects. Physical constraints could also be encoded directly into the VAE, for example by changing the assumed prior distributions of the latent variables in Equation A.4 to those expected of thermophysical parameters on the lunar surface, or by adding physics constraints directly into the VAE's loss function.

Another direction would be to incorporate the rest of the wavelength channels from the Diviner instrument into the VAE workflow. The VAE framework is readily extendable in this regard and it may be able to disentangle cross-channel effects such as anisothermality to estimate rock abundance (channels 6-8) [Williams et al., 2016, 2017, Bandfield et al., 2011] or to recognise the Christiansen feature (channels 3-5) that is indicative of bulk silicate [Greenhagen et al., 2010]. We discarded the standard deviation vector of the approximate latent posterior distribution produced by the VAE when carrying out inference, but this could also be used to obtain uncertainty estimates of the underlying thermal parameters. The ability of the VAE to carry out fast inference may mean it is useful for real-time inference, for example in detecting transient temperature anomalies from fresh impacts.

It would also be useful to investigate how different preprocessing of the raw Diviner data affects the model learned by the VAE. In particular, we found that the selection of the 1D training profiles had a significant effect on the performance of the model. The VAE was only able to disentangle the thermophysical factors when the training profiles had a large variation, and thus we hand-selected 48 AOIs to extract training profiles from to ensure this. However, an automated approach which randomly selects profiles across the lunar surface, whilst still ensuring diversity in the training set, could be less biased.

Finally, as our method is physics-agnostic it could be applied to other similar lunar science and exploration datasets. For example, other useful applications may be to use the VAE to model differences in optical illumination of the lunar surface over the lunar day or to localise regions with over-night temperature flattening that might indicate multi-layer sub-surfaces, such as cryptomare.

3.6 Summary

We have shown that ML can learn about the underlying physical processes in real data. Starting from some only very general assumptions about the physical system, our VAE was able to disentangle multiple different thermophysical processes on the lunar surface, even though these processes had complex characteristics and the dataset contained real-world noise. Furthermore, the VAE provided a much faster approach for analysing the thermal characteristics of the lunar surface compared to traditional thermophysical modelling. However, the complexity of the dataset and the "black-box" nature of the VAE's neural networks meant that it was challenging to interpret the model learned by the VAE, and it was not able to fully disentangle some of the physical processes. Future work is required to understand how best to improve the disentangling ability of the VAE and to build physical constraints directly into the ML workflow.

4

Combining deep learning with a physical model for denoising lunar imagery

4.1 Introduction

In the previous chapter we assessed how well PIML is able to discover underlying physical principles from real data. In this chapter we consider a related scalability question which is, can PIML algorithms carry out inversion using real data? In particular, we evaluate the ability of a PIML algorithm for denoising extremely low-light images of permanently shadowed regions (PSRs) on the Moon taken by the Narrow Angle Camera (NAC) on board the Lunar Reconnaissance Orbiter (LRO) satellite.

The NAC is another instrument on board the LRO and its main purpose is to deliver high resolution optical images of the lunar surface. These images have many important applications across lunar science and exploration, with one being the study of PSRs. These regions are among the coldest places in the solar system and are likely to host water, making them key scientific targets for future missions.

However, NAC images of these regions contain significant amounts of noise, which makes them extremely challenging to interpret. This is fundamentally due to the very low numbers of photons that are reflected from these regions, and also in part because the instrument was not originally designed to operate in such low-light conditions.

Removing this noise would provide a significant benefit to the lunar community, as it would allow us to peer into these regions with high resolution for the first time.

Alongside its benefit to lunar science, this problem is an excellent candidate for evaluating how well PIML methods for inversion scale to real-world problems. The underlying inversion problem is to estimate the mean photon count arriving at the camera given a noisy image. Compared to carrying out inversion on a toy version of this problem (for example, learning to remove synthetic photon noise from images), this problem is significantly harder because of the real and complex nature of the multiple noise sources contained in the images.

Many different noise sources are present. One is photon (shot) noise, which is due to the inherent randomness of the photons arriving at the camera. Another is CCD-related dark noise, which depends on multiple environmental factors and has a complex character. Other noise sources include nonlinearities in the camera's response and compression noise when the image is downlinked to Earth. When combined together, modelling and removing this noise is a challenging task.

To denoise these images, we propose a purpose-built PIML algorithm that combines both traditional and learned processing steps. The workflow is physics-informed in the sense that each element of the algorithm targets different types of noise in the image (i.e., it is physically-motivated) and traditional denoising elements are included (i.e., it is a hybrid approach). First, a deep neural network is used to estimate and remove the dark noise contained in the image, given a set of environmental meta-data (such as CCD temperature and orbit number) available at the time of image capture. Next, the existing flatfield and nonlinearity corrections taken from the standard calibration routine of the camera are applied. Finally, another deep neural network is used to estimate and remove all other noise sources contained in the image, such as photon noise and image compression noise. Furthermore, to train the networks we generate training data by developing a hybrid noise model of the camera which combines both synthetic noise and real noise samples from dark calibration frames.

We find that our PIML algorithm is able to significantly enhance images of shadowed regions on the lunar surface. For the first time, surface features such as

boulders and craters within PSRs down to 3-5 m in size can be seen. The approach is compared to the existing calibration routine of the camera and a naive end-to-end ML baseline and it significantly outperforms both across all metrics tested. These images offer a step change for lunar exploration and science and are likely to have a significant impact, for example by aiding the identification of surface water-ice and reducing uncertainty in rover and human traverse planning into PSRs.

Whilst effective, limitations are encountered when using our PIML approach. Firstly, because of the complexity of the noise contained in the images, significant effort is required to ensure our physical noise model of the camera generates realistic training data. This is important to ensure our algorithm generalises well to real images. Secondly, because of the lack of interpretability of the neural networks and the ill-posedness of the inverse problem, it is difficult to verify the outputs of our algorithm. Both of these limitations are discussed further.

4.2 Background

4.2.1 Peering into permanently shadowed regions

PSRs are craters and other topographic depressions near the north and south poles of the Moon that have not received any direct illumination for millions of years and as a consequence are extremely dark and cold places (25 K to 70 K) [Arnold, 1979, Fisher et al., 2017]. These enigmatic environments are believed to have trapped and accumulated water and other volatiles in the form of ice from processes such as solar winds, cometary impacts and volcanic outgassing during the evolution of our Solar System. For this reason, PSRs are of key interest to lunar science and exploration, as they could both provide clues about the Moon's past and host water which is essential for sustaining our human presence [NASA, 2020, ESA, 2019]. Multiple missions are planned to study PSRs in the near future; for example NASA's Volatiles Investigating Polar Exploration Rover (VIPER) rover aims to drive into a PSR to search for water in 2023 [Colaprete et al., 2021].

However, studying the interiors of PSRs is very challenging. Fundamentally, this is because their darkness makes them very difficult to image. Whilst no direct sunlight

enters PSRs some photons can arrive into them after being reflected from their nearby sunlit surroundings, which is known as secondary illumination. Therefore, images of PSRs can be captured, but in this extremely low-light setting these images are dominated by noise. Furthermore, the amount of secondary illumination depends heavily on the surrounding topology, and therefore the signal-to-noise ratio (SNR) of these images can vary.

The current state-of-the-art images of the lunar surface are captured by the NAC onboard the LRO [Chin et al., 2007, Robinson et al., 2010]. This is a panchromatic optical camera which is acquiring images with a nominal $\sim 0.5-2$ m spatial resolution. The camera launched in 2009 and over its lifetime it has captured over 2 million images of the lunar surface, enough to cover the surface multiple times over. Whilst its images have high spatial resolution, its short exposure times and the lack of photons means that CCD-related and photon noise dominates these images over shadowed regions, making any meaningful interpretation of the small-scale features in PSRs challenging. Currently, the only way to capture images of some PSRs with reasonable SNR is to significantly increase the exposure time of the camera, at the expense of significantly reducing spatial resolution [Cisneros et al., 2017]. Figure 4.1 (column 1) shows example raw (normal, short exposure) NAC images captured over 3 PSRs at the lunar south pole.

In this chapter we present a PIML approach for denoising these images, allowing us to peer into lunar PSRs with high resolution for the first time. We name our approach Hyper-Effective Noise Removal U-Net Software, or HORUS[1].

4.2.2 Related work

Whilst the NAC is a scientific camera operating in a remote environment, many of its noise sources (e.g. camera-related noise and photon noise) are the same as those found in other low-light imaging applications. Low-light environments occur in many different important applications, such as night-time photography, astronomy

[1]The name HORUS represents our interest in complementing the existing LRO NAC image processing pipeline, named Integrated Software for Imagers and Spectrometers (ISIS) [Gaddis et al., 1997]. Isis also happens to be the mother of Horus in Egyptian mythology.

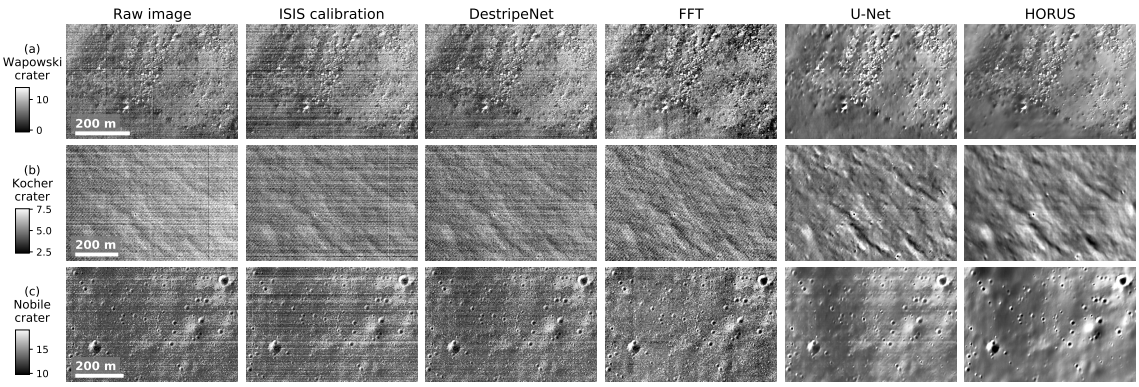


Figure 4.1: Examples of HORUS applied to 3 real PSR images, compared to the baseline approaches. Grayscale colorbars show the estimated mean photon count S in DN for each HORUS plot. Raw/ISIS image credits to LROC/GSFC/ASU.

and microscopy [Chen et al., 2018b, Chromey, 2016, Zhang et al., 2019b], and the images captured usually have very poor quality.

Direct approaches for removing noise include increasing the exposure time or by using burst photography and aligning and combining the images during post-processing. However, these approaches have significant downsides; for example longer exposure times can cause blurring and burst photography can still be sensitive to dynamic scenes [Hasinoff et al., 2016, Karadeniz et al., 2020]. Instead, techniques from the field of image denoising can be applied, for which there exists a rich literature [Buades et al., 2005, Goyal et al., 2020, Fan et al., 2019].

For denoising low-light images, simple traditional methods include histogram equalisation and gamma correction [Cheng and Shi, 2004], whilst more sophisticated methods include using wavelet transforms [Łoza et al., 2013], retinex-based methods [Park et al., 2017, Xu et al., 2020a, Guo et al., 2017] and principle component analysis [Salmon et al., 2014]. More recently, deep learning based methods have become hugely popular [Xu et al., 2020b, Wei et al., 2020, Maharjan et al., 2019, Zhang et al., 2019b, Chen et al., 2018b, Ren et al., 2019, Remez et al., 2017]. In these approaches, deep neural networks are used to learn an implicit representation of signal and noise. For example, Chen et al. [2018b] proposed an end-to-end network with a U-Net [Ronneberger et al., 2015] architecture to convert short-exposure images to their corresponding long-exposure images, whilst Xu et al. [2020b] proposed

a frequency-based enhancement scheme. Such approaches have shown significant improvements over popular traditional methods.

Whilst promising, a major disadvantage of using deep learning methods for image denoising is that they often require large amounts of labelled training data to prevent overfitting, which can be expensive to collect, or unavailable [Abdelhamed et al., 2018, Chen et al., 2018b, Zhang et al., 2019b]. An alternative approach is to synthetically generate noisy images, although many works use simple additive Gaussian noise models or similar, leading to poor performance on real images [Plötz and Roth, 2017]. A potential improvement is to incorporate both synthetic and real images into the training set [Guo et al., 2019]. For low-light images, Wei et al. [2020] focused on improving the realism of their synthetic data by incorporating a comprehensive set of noise sources based on a physical noise model of CMOS sensors and showed that this performed as well as a deep neural network trained on real clean-noisy image pairs.

Specifically for low-light PSR LRO NAC images, Sargeant et al. [2020] attempted to remove noise using a Canny edge detector and a Hough transform with local interpolation. Their method worked sufficiently well for images with relatively high photon counts but failed for low photon counts, i.e., the vast majority of high-latitude PSRs. The current calibration routine of the camera also attempts to remove noise in the images, but is not specifically designed for images of PSRs [Robinson et al., 2010, Humm et al., 2016].

In this work we extend existing learning-based methods in several respects. Firstly, in order to avoid overfitting and improve generalisation performance, we combine a realistic physical noise model of the camera with real noise samples from dark frames to generate realistic training data. We also use 3D ray tracing to select training image scenes which best match the illumination conditions expected in PSRs. Secondly, instead of using the noisy image as the only input to the learned model as is typically done [Chen et al., 2018b, Wei et al., 2020, Xu et al., 2020b], we condition our model on the camera's environmental metadata available at the time of image capture, allowing us to account for the effect of external factors such as camera temperature on the noise. Finally, whilst many recent works focus on images from CMOS sensors

[Chen et al., 2018b, Wei et al., 2020, Xu et al., 2020b, Maharjan et al., 2019, Szeliski, 2010], we formulate a physical noise model for a CCD sensor.

4.2.3 Instrument overview

The NAC consists of two nominally identical cameras ("left" and "right") which capture publicly-available[2] 12-bit panchromatic optical images [Robinson et al., 2010, Humm et al., 2016]. Each camera consists of a 700 mm focal length telescope, which images onto a Kodak KLI-5001G 5064-pixel CCD line array, providing a 10 μrad instantaneous field of view. The array is oriented perpendicular to the direction of flight of the satellite and 2D images are captured by taking multiple line scans as the spacecraft moves. Because of this motion, the in-track spatial resolution of the images depends on the exposure time as well as the spacecraft altitude and typically both the in-track and cross-track resolution are in the range 0.5 – 2.0 m. Each camera has two operating modes, "regular" and "summed", where the summed mode typically uses twice the exposure time and sums adjacent pixels during image capture. This is frequently used to maximise the signal received over low-lit regions of the Moon (such as PSRs), at the expense of halving the spatial resolution. For every image the NAC also records over 50 fields of environmental metadata at the time of capture, which for example includes the camera's temperature and orbit number, and the values of 60 physically masked pixels located on the edges of the CCD.

4.3 Methods

4.3.1 Physical noise model

In this extremely low-light setting these images are dominated by noise. As part of HORUS, we propose a physical model of this noise. The model is informed by the model developed by Humm et al. [2016], who carried out comprehensive characterisation of the instrument before and during its deployment, as well as

[2] lroc.sese.asu.edu/data

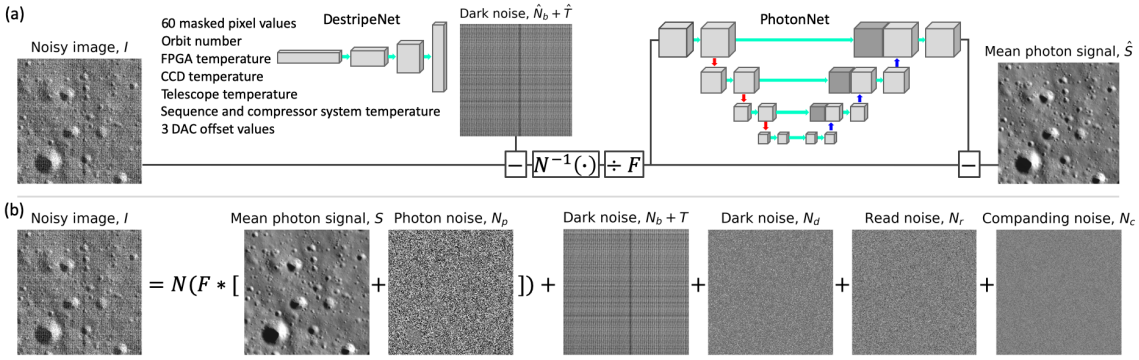


Figure 4.2: HORUS denoising approach and physical noise model. (a) HORUS denoising workflow. The input to the workflow is the raw, noisy low-lit image I. First a model of the deterministic elements of the dark noise, namely the dark bias N_b and the mean dark current T, is subtracted from the image, which is estimated using a convolutional decoder from the environmental and masked pixel data available at the time of image capture. Next, the inverse nonlinearity and flatfield corrections are applied. Lastly, residual noise sources including the photon noise N_p, stochastic dark current noise N_d, read noise N_r, and companding noise N_c are estimated and subtracted using a network with a U-Net architecture. (b) Physical noise model. We assume that the raw image I contains companding noise, read noise, dark current and bias noise, photon noise and a nonlinearity and flatfield response. The quantity we wish to recover is the mean photon signal, S. S image credit to LROC/GSFC/ASU.

standard CCD theory [Chromey, 2016], and is given by

$$I = N(F * (S + N_p)) + N_b + (T + N_d) + N_r + N_c \ , \tag{4.1}$$

where I is the raw image detector count in Data Numbers (DN) recorded by the camera, N is the nonlinearity response, F is the flatfield response, S is the mean photon signal (which is the desirable quantity of interest), N_p is photon noise, N_b is the dark bias, $T + N_d$ is the dark current noise, N_r is read out noise and N_c is companding noise. A depiction of our noise model is shown in Figure 4.2 (b). In the following we describe each noise source in greater detail.

The *photon noise* N_p is due to the inherent randomness of photons arriving at the CCD, and obeys

$$(S + N_p) \sim \mathcal{P}(S) \ , \tag{4.2}$$

where $\mathcal{P}$ denotes a Poisson distribution. The strength of this noise depends on the mean rate of photons hitting the CCD, and it represents a fundamental limitation for any sensor.

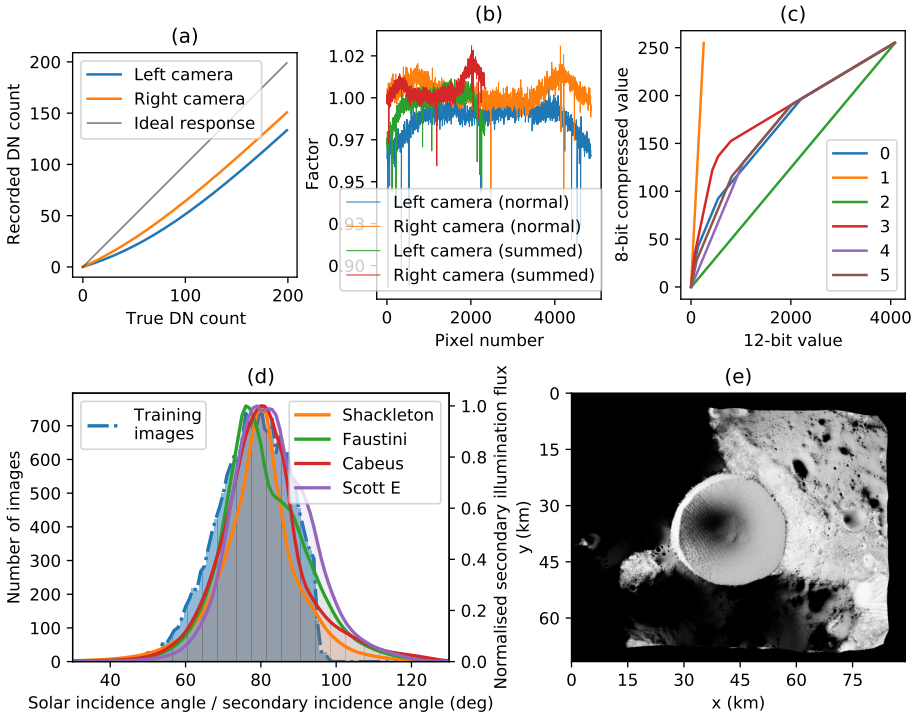


Figure 4.3: NAC characteristics and training image selection. (a) Nonlinearity response. Separate responses are used for the left and right cameras. (b) Flatfield response. Separate flatfields are used for each camera and operating mode. (c) The six compression schemes commandable for compressing NAC images from 12-bit values to 8-bit values. (d) Distribution of solar incidence angles in the training set for HORUS compared to the distribution of secondary incidence angle of scattered rays using 3D ray tracing over 4 PSRs. (e) Render of the secondary illumination radiance from ray tracing of the Shackleton crater PSR at the lunar south pole (white indicates increased secondary illumination).

The *dark bias* N_b is a deterministic noise source which is due to an artificial voltage offset applied to the CCD to ensure that the Analogue-to-Digital Converter (ADC) always receives a positive signal, and varies pixel-to-pixel, manifesting itself as vertical stripes in the image. Different offsets are commanded depending on the temperature of the NAC, and we also consider the possibility that the bias changes over the camera's lifetime as it degrades.

The *dark current noise* is generated by thermally-activated charge carriers in the CCD which accumulate over the exposure duration and obeys

$$(T + N_d) \sim \mathcal{P}(T) \ , \tag{4.3}$$

where T is the mean number of thermally-activated charge carriers which depends on the CCD temperature. This noise source varies pixel-to-pixel and with each image

line and introduces horizontal and vertical stripes in the image.

The *read out noise* N_r is stochastic system noise introduced during the conversion of charge carriers into an analogue voltage signal and is estimated by Humm et al. [2016] to have a standard deviation around 1.15 DN for both cameras.

A perfect camera is expected to have a linear response between the recorded DN value and the mean photon signal S. However during laboratory characterisation Humm et al. [2016] showed that the response of the NAC becomes nonlinear for DN counts below 600 DN. They proposed an empirical *nonlinearity correction* to correct for this effect, given by

$$N^{-1}(x) = (x + d) - \frac{1}{ab^{(x+d)} + c} \ , \tag{4.4}$$

where x is the recorded pixel DN value and $N^{-1}(x)$ is the estimated true DN value. Here a, b, c and d are free parameters which Humm et al. [2016] estimated through experimental calibration. In this work we use the average parameter values reported by Humm et al. [2016] for each camera, using the same values across all pixels. For computing the forward function $N(x)$ we are not aware of an analytical inverse of Equation 4.4 and instead use numerical interpolation. The nonlinearity correction curves used are shown in Figure 4.3 (a).

The *flatfield correction* is used to correct for pixel-to-pixel variations in the sensitivity of the camera, which may be due to the optics or the detector, such as vignetting or particulates on the detector. The forward response is modelled by multiplying the flatfield F, which is a vector of gain values, point-wise with each pixel. Humm et al. [2016] experimentally measured separate flatfields for each camera and each operating mode and we use the same, shown in Figure 4.3 (b).

Finally, the NAC images are compressed (companded) from 12-bits to 8-bits using a lossy compression scheme before downlink to Earth [Robinson et al., 2010]. This introduces *companding noise* and represents a fundamental limit on DN resolution. Six different compression schemes can be commanded; for PSR images scheme 3 in Figure 4.3 (c) is frequently used, which more accurately reconstructs lower DN

values, leading to a maximum absolute reconstruction error after decompression of 2 DN for counts below 424 DN.

Compared to the noise model proposed by Humm et al. [2016], our noise model explicitly considers photon noise, stochastic dark current noise, read noise and companding noise.

4.3.2 Training data generation

4.3.2.1 Hybrid noise generation

To train HORUS we generate a large dataset of clean-noisy image pairs. Noisy images are generated by combining our physical noise model (Equation 4.1) with noise sampled from real dark calibration frames. An example of a real dark calibration frame is shown in Figure 4.4 (d). These are captured by the NAC whenever the LRO passes over the non-illuminated side of the Moon and nominally receive no sunlight. We therefore assume that they sample the following terms in our physical noise model[3],

$$D \simeq N_b + (T + N_d) + N_r \ . \tag{4.5}$$

Over 70,000 dark calibration frames have been captured, which cover the entire lifetime of the instrument over a wide range of camera states and environmental conditions; the histogram of orbit numbers and CCD temperatures at the time of image capture over the entire set are shown in Figure 4.4. We assume that these frames are a representative and sufficient sample of the dark noise distribution of the camera.

Given an input clean image representing the mean photon signal $\tilde{S}$, we generate its corresponding noisy image $\tilde{I}$ using the following equation,

$$\tilde{I} = N(F * (\tilde{S} + N_p)) + D + N_c \ , \tag{4.6}$$

where D is a randomly selected real dark calibration frame, N_p is synthetic, randomly generated Poisson noise and N_c is the noise generated by compressing and decompressing the noisy image through the NAC companding scheme 3. By using a hybrid noise model (i.e. a model which combines real and synthetic noise components),

[3]Dark frames are compressed using scheme 1 in Figure 4.3 (c), which is a lossless one-to-one mapping and so N_c is not included.

Equation 4.6 does not entirely rely on the assumptions of our physical noise model and we ensure that the distribution of noise in our training data is as close as possible to the real noise distribution. To be explicit, the synthetically generated components of our training images $\tilde{I}$ in Equation 4.6 are $N(\cdot)$, F, N_p, and N_c, whilst D is derived from real dark noise frames and $\tilde{S}$ is derived from real sunlit images (as explained below).

4.3.3 Image pre-processing and selection based on 3D ray tracing

Clean images $\tilde{S}$ are generated by extracting millions of randomly selected 256×256 image patches cropped from NAC images of the lunar surface in normal sunlit conditions. We define a sunlit image as any image with a median DN > 200 and our assumption is that in this regime the noise sources in Equation 4.1 become negligible, such that we can use sunlit images as a proxy for S. An example sunlit image is shown in Figure 4.2 (b).

An important difference between sunlit images and the test-time images of PSRs are the illumination conditions; whilst sunlit images are illuminated by direct sunlight, PSRs are only illuminated by secondary (back-scattered) light. In order to match the illumination conditions of the training data to the test-time data as best we can, we select sunlit images with similar solar incidence angles to the secondary illumination incidence angles expected over PSRs. 3D ray tracing is performed over 4 example PSRs using 30 m/pixel spatial resolution LRO Lunar Orbiter Laser Altimeter elevation data [Riris et al., 2017] and a Lambertian bidirectional reflection function. The solar incidence angle is available as metadata for each sunlit image, and we match the distribution of solar incidence angles in our training images to the distribution of secondary illumination angles from ray tracing, shown in Figure 4.3 (d).

Before using the sunlit images as training data we re-scale their DN values to those expected for PSR images. Each image patch is divided by its median DN value and multiplied by a random number drawn from a uniform distribution over the range 0-60 DN, which is what we expect for typical PSRs. Images are sampled from

the entire lunar surface with no preference on their location and in total over 1.5 million image patches $\tilde{S}$ are extracted for each camera in each operating mode.

4.3.4 Denoising workflow

The workflow HORUS uses to denoise images is shown in Figure 4.2 (a). The input is the noisy low-lit image, I, after decompression. First a convolutional decoder called DestripeNet is used to predict the dark bias and mean dark current, given a set of environmental metadata and the masked pixel values at the time of image capture, which is subtracted from the input image. Next, the inverse nonlinearity and flatfield corrections are applied. Lastly, a network with a U-Net [Ronneberger et al., 2015] architecture called PhotonNet is used to estimate the residual noise sources in the image, which are subtracted from the image.

The input to DestripeNet is a vector of 8 selected environment metadata fields available at the time of image capture (listed in Figure 4.2 (a)), concatenated with the masked pixel values. Our assumption is that this vector is sufficient to predict the dark bias and mean dark current for an image. DestripeNet is trained using the real dark calibration images as labels and a L2 loss function. Under a L2 loss the network learns to predict the mean of its output variable conditioned on its inputs [Smyth, 1993], such that, under the assumptions of Equation 4.5, it estimates the quantity $(N_b + T)$.

Given the DestripeNet prediction $(\hat{N}_b + \hat{T})$, the input to PhotonNet is

$$J = N^{-1}(I - (\hat{N}_b + \hat{T}))/F \ . \tag{4.7}$$

We train PhotonNet using the synthetic image pairs $(\tilde{I}, \tilde{S})$ described in Sections 4.3.2.1 and 4.3.3. Before inputting the noisy training images $\tilde{I}$ into the network, Equation 4.7 is applied using their DestripeNet prediction, and we use $\tilde{J} - \tilde{S}$ as training labels. Thus, PhotonNet learns to estimate the (transformed) residual noise sources in the image, namely the photon noise, stochastic dark current noise, read noise, companding noise, and any residual dark bias and mean dark current noise which DestripeNet failed to predict.

DestripeNet is trained to predict each image line separately and is comprised of 13 1D transposed convolutional layers with a filter length and stride of 2 and ReLU activation functions. The number of hidden channels starts constant at 512 and then halves every layer from 512 to 16 in the last layer. Each metadata field is independently normalised before input. PhotonNet operates on 2D image patches and uses a standard U-Net architecture [Ronneberger et al., 2015] with 4 downsampling steps and 1 convolutional layer per scale with LeakyReLU activation functions with a negative slope of 0.1. The number of hidden channels doubles after every downsampling step from 32 to 512. DestripeNet is trained before PhotonNet and its predictions used for training PhotonNet are pre-computed. Both networks are trained using the Adam stochastic gradient descent algorithm [Kingma and Ba, 2015] with a L2 loss function, using a batch size of 400 image lines for DestripeNet and 10 image patches for PhotonNet. Learning rates of 1×10^{-5} and 1×10^{-4} are used for DestripeNet and PhotonNet respectively. We divide the synthetic image dataset $(\tilde{I}, \tilde{S})$ into a training, validation and test set (72:18:10). Finally, to account for possible differences in the two cameras and their two operating modes, separate networks are trained for each possible configuration.

4.3.5 Baselines

We compare our approach to a number of baselines;

Current NAC calibration routine (ISIS). We use the Integrated Software for Imagers and Spectrometers (ISIS) [Gaddis et al., 1997], which is the current calibration routine for NAC images. The calibration removes an estimate of the dark bias and mean dark current noise by fitting a cyclic trend to the masked pixel values, and also applies the nonlinearity and flatfield corrections [Humm et al., 2016].

DestripeNet only. For this case, we only subtract the DestripeNet prediction from the image and then apply the nonlinearity and flatfield correction, i.e. PhotonNet is not used.

Fast Fourier Transform (FFT) filtering. We develop a hand-crafted FFT denoising algorithm. The 2D image is transformed into the frequency domain and its

		ISIS	DestripeNet	FFT
Mode	**Camera**	L1 / PSNR / SSIM		
Normal	Left	5.68 / 31.12 / 0.61	5.58 / 31.33 / 0.62	4.93 / 33.38 / 0.80
	Right	5.09 / 32.23 / 0.67	4.99 / 32.44 / 0.68	4.61 / 34.20 / 0.83
Summed	Left	5.45 / 31.51 / 0.65	5.44 / 31.59 / 0.65	4.50 / 34.04 / 0.83
	Right	4.96 / 32.48 / 0.70	4.92 / 32.59 / 0.70	4.21 / 34.82 / 0.85

		U-Net	HORUS
Mode	**Camera**	L1 / PSNR / SSIM	
Normal	Left	1.64 / 42.50 / 0.96	**1.30** / **44.63** / **0.97**
	Right	1.43 / 44.08 / 0.96	**1.23** / **45.35** / **0.97**
Summed	Left	1.69 / 42.34 / 0.95	**1.52** / **43.33** / **0.96**
	Right	1.60 / 42.93 / **0.96**	**1.43** / **44.02** / **0.96**

Table 4.1: Synthetic test set performance of HORUS, compared to the baseline approaches. All metrics compare the ground truth image $\tilde{S}$ to the estimated denoised image and the average performance over all images is reported. Higher is better, apart from L1 error.

zero-frequency components are replaced with the mean of their nearest neighbours to remove horizontal and vertical dark noise stripes in the image. A mild low-pass filter using a 2D Gaussian kernel with a standard deviation of 0.25 m^{-1} and the nonlinearity and flatfield corrections are also applied.

End-to-end U-Net. Instead of two separate networks, we train a single U-Net to directly estimate $\tilde{S}$ given $\tilde{I}$ as input, without using any metadata. This end-to-end strategy is typical in many existing works [Chen et al., 2018b, Maharjan et al., 2019, Wei et al., 2020]. The U-Net has the same architecture and training scheme as PhotonNet.

All relevant baselines are trained on and/or have their hyperparameters selected using the same training data as HORUS, and all are tested on the same data as HORUS.

4.4 Results

4.4.1 Results on synthetic images

Table 4.1 shows the quantitative performance of HORUS compared to the baselines across the synthetic test set of images. We find that HORUS gives the strongest

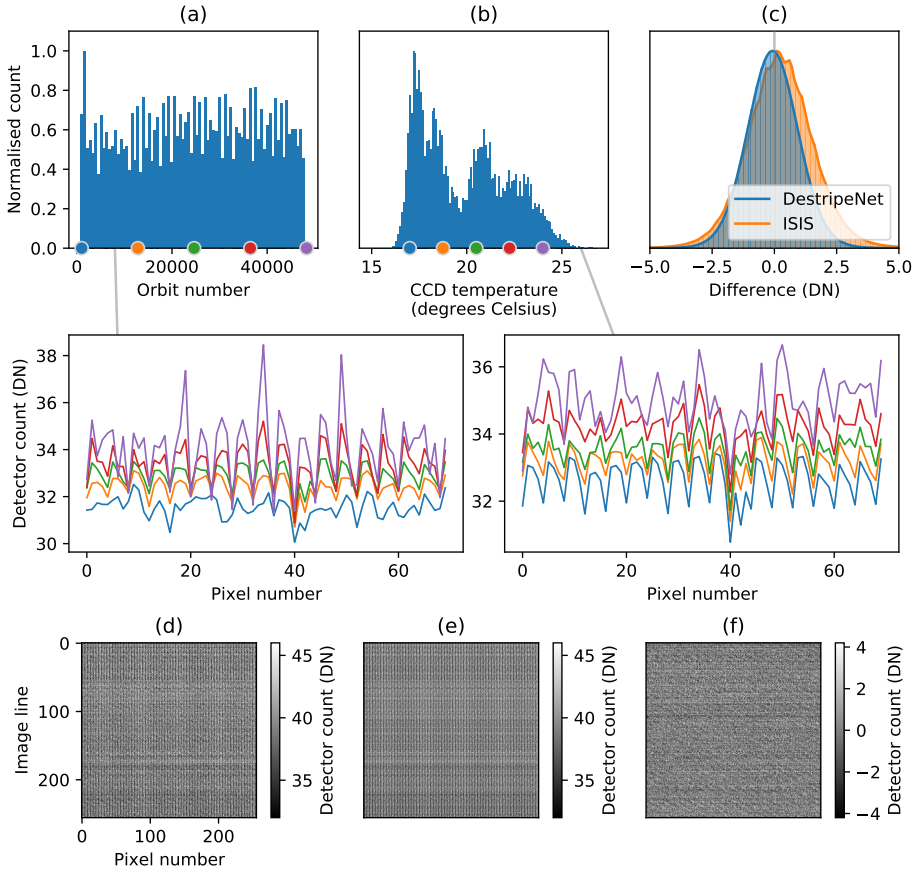


Figure 4.4: Example dark noise predictions generated by DestripeNet. (a) Histogram of orbit numbers over the dark calibration frames. The linked line plot shows the predicted dark level using DestripeNet for the first 70 pixels of the right camera in summed mode when varying the orbit number and fixing the other metadata inputs to those of a randomly selected dark frame. Colour-coded dots in the histogram show the value of the orbit number for each line. (b) Similar set of plots for the CCD temperature. (c) Histograms of the difference between the DestripeNet and ISIS dark frame predictions and the ground truth dark calibration frame over the test set of summed mode images. (d) Example portion of a real dark calibration frame. (e) Corresponding DestripeNet dark level prediction. (f) Difference between (d) and (e).

performance across all metrics for this dataset, and significantly outperforms the ISIS, DestripeNet-only and FFT baselines.

Example HORUS denoising of a synthetic test image with varying mean photon counts are shown in Figure 4.5. We also plot the mean absolute percentage error between the ground truth image $\tilde{S}$ and the denoised image, binned by median DN value of the ground truth image, for HORUS and the baselines over the test set of summed mode images. We find that the performance of all the methods degrades with lowering photon counts, however HORUS gives the lowest error across all DN

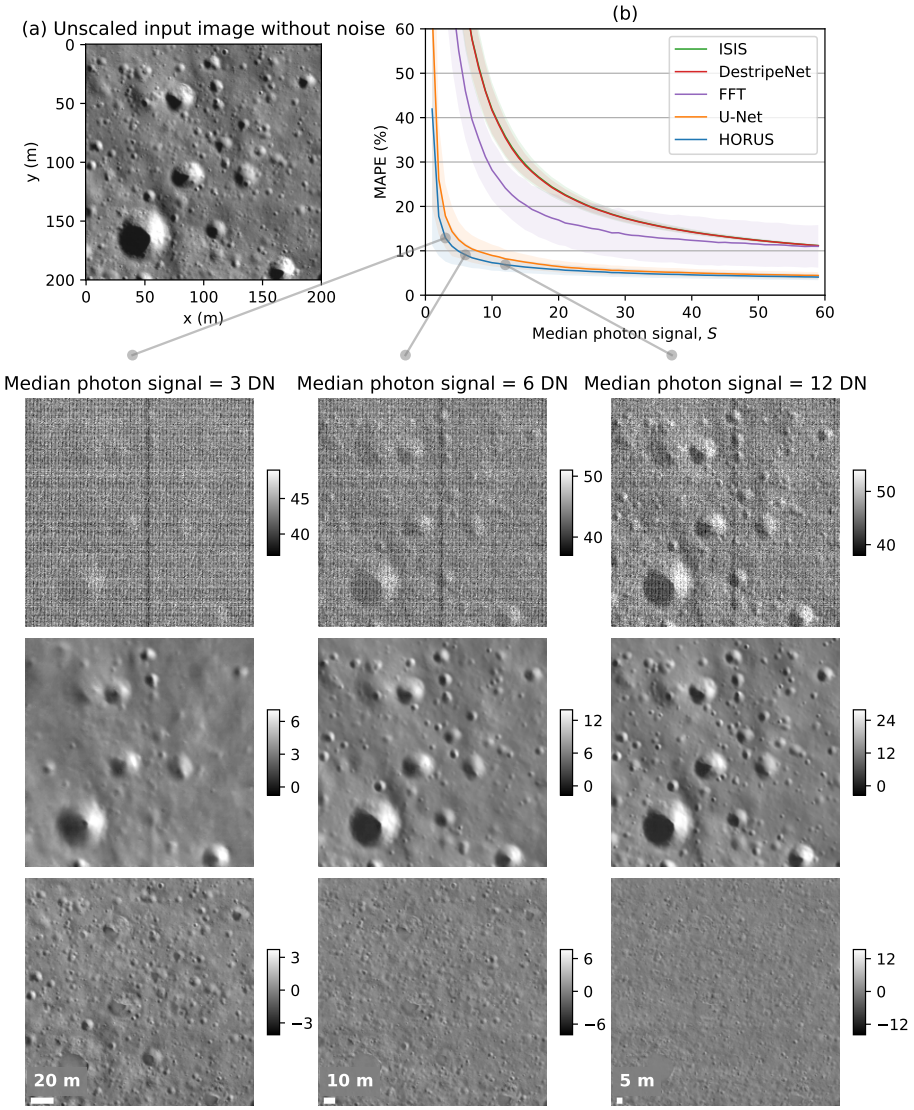


Figure 4.5: Synthetic test set performance of HORUS with varying signal strengths. (a) Unscaled ground truth image S used below. (b) Mean absolute percentage error between the ground truth image and the denoised image, binned by median DN value of the ground truth image, for HORUS and the baselines over the synthetic test set of summed mode images. Filled regions show ± 1 standard deviation. The image grid shows the ground truth image from (a) scaled to different median photon counts with noise added according to our physical model (top row), HORUS denoising of this image (middle row) and the difference between HORUS and the ground truth image (bottom row). Colorbar shows DN counts. Raw image credits to LROC/GSFC/ASU.

values. The example denoised images suggest that HORUS can denoise synthetic images with median photon counts as low as 3-6 DN. The difference plots suggest that the minimum feature size HORUS can reliably resolve correlates strongly with the photon count, and for median photon counts of $\sim$12 DN this is $\sim$5 m. Further

qualitative examples of HORUS denoising on synthetic images with varying photon counts are shown in Figure B.1.

4.4.2 DestripeNet results

The histogram of the error between the DestripeNet prediction and the real dark calibration frame over the calibration frame test set is shown in Figure 4.4 (c). We find that DestripeNet is typically able to reconstruct the dark frames to within 1 DN. We plot scans of its prediction over the orbit number and CCD temperature whilst keeping the other metadata inputs fixed. We find that these predictions are physically interpretable: increasing the CCD temperature increases the dark DN level, which is expected, and the variance of the prediction increases with orbit number, possibly indicating a degradation of the instrument over time.

4.4.3 Results on real images

Qualitative examples of HORUS and the baselines applied to 3 real images of different PSRs are shown in Figure 4.1. We find that HORUS gives the most convincing results on these images; compared to the ISIS, DestripeNet-only and FFT baselines it removes much more of the high frequency stochastic noise in the images, and compared to the U-Net it removes more of the residual dark noise stripes in the image and introduces less low-frequency noise. Further qualitative examples of HORUS denoising on real images with varying latitude are shown in Figure B.2.

4.5 Discussion

4.5.1 Validation of real images

The quantitative and qualitative results above suggest that HORUS is able to significantly enhance noisy images of shadowed regions on the lunar surface and strongly outperform the existing calibration routine of the camera and other baselines.

However, whilst we thoroughly assess the performance of HORUS on synthetic images above, it remains important to validate the performance of HORUS on real images. This is because, even though we make every effort to match our training

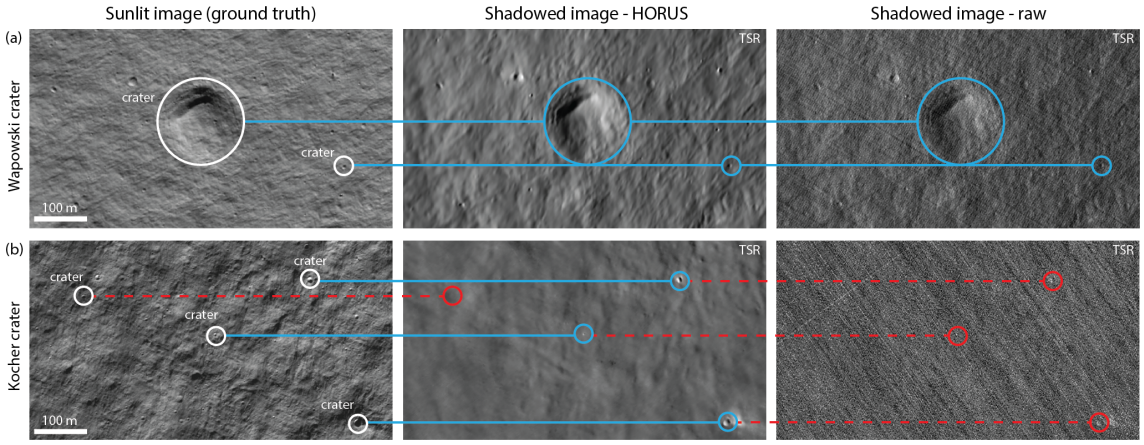


Figure 4.6: Qualitative verification of HORUS using map-projected sunlit (left) and shadowed image pairs in (a) Wapowski crater and (b) Kocher crater; HORUS-denoised frames (center) are compared with their raw input frames (right). Some features are resolved in both the HORUS and raw frames (blue marks), but HORUS resolves a significantly larger number of features. With decreasing photon counts (example (b)), HORUS struggles to resolve smaller features (here less than ∼10 m across, red marks), resulting in a smoother image. Differences in shadowing are caused by different sunlit/secondary illumination incidence angles. Raw image credits to LROC/GSFC/ASU.

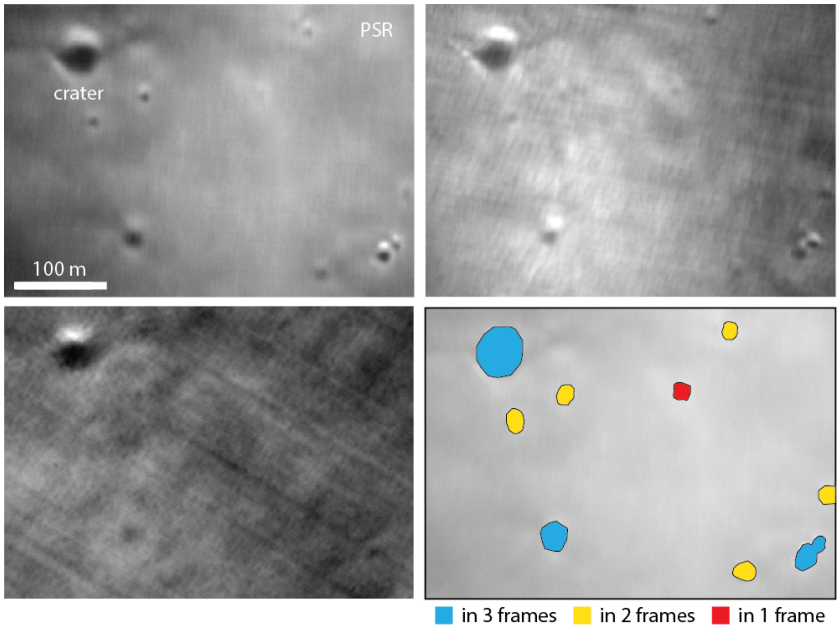


Figure 4.7: In-PSR qualitative verification of 3 overlapping map-projected HORUS frames with varying levels of signal (decreasing clockwise from top left to bottom left). Some features are present in all frames (blue), some in 2 (yellow), and some only in 1 (red).

data to the real distribution of PSR images, the real images may vary, and thus the generalisation ability of HORUS should be assessed. However, a key challenge in this setting is that there exist no real ground truth pairs of clean and noisy

images of PSRs to verify HORUS's performance. Instead, we use two alternative approaches which are described below.

Ground truth validation using temporary shadowed regions. One way to verify HORUS denoising on real images is to use temporary shadowed regions (TSRs). These are regions which transition between being sunlit and shadowed over the course of a lunar day, and thus we can use sunlit images of TSRs as ground truth when verifying HORUS denoising of their shadowed images. We compare raw sunlit images and HORUS denoised shadowed images of two TSRs in Figure 4.6. We find that for images with sufficient photon counts (e.g. Wapowski crater in Figure 4.6) HORUS is able to resolve the vast majority of topographic features, e.g., impact craters as small as ∼4 m across. For images with lower photon counts (e.g. Kocher crater in Figure 4.6) HORUS is able to resolve most large-scale topographic features, although higher frequency details are sometimes lost. Importantly, we do not observe any hallucinations (false positives) or other artefacts in the real HORUS images which could suggest poor generalisation of HORUS. In sharp contrast, the raw input image is affected by intense noise, making any meaningful observations difficult.

Validation using overlapping images. Another way to verify HORUS performance is to compare HORUS denoising across overlapping image frames. Here we do not have definitive ground truth as all images are shadowed, but we can analyse whether topographic features appear consistent throughout all HORUS frames. We compare three overlapping HORUS images taken over a PSR at the lunar south pole in Figure 4.7. As illustrated in Figure 4.7, we can trace topographic features through all three frames. However, the size of resolvable features increases with decreasing photon counts: The smallest feature we can identify in the highest photon count frame (top left) is ∼7 m across, while the smallest feature in the lowest photon count frame (bottom left) is ∼14 m across. This is consistent with the synthetic observations in Figure 4.5.

Both of the validation approaches above suggest that HORUS is able to reliably generalise to real images of PSRs. Furthermore, both approaches can be applied whenever a new PSR is processed using HORUS, as most PSRs are surrounded by TSRs and have multiple overlapping frames available.

4.5.2 Future work

There is much future research which could follow from this work. Firstly, HORUS allows us to see features such as boulders and craters within PSRs down to 3-5 m in size for the first time, and therefore enables many new and exciting applications in lunar science. For example, HORUS could be used to study the geomorphology and traversability of PSRs, as well as their potential for hosting volatiles such as water-ice. This would have significant impact in both lunar science and exploration. Furthermore, there exist over 200,000 LRO NAC images over the lunar poles, and applying HORUS to these could enable large-scale spatial, temporal and seasonal variations of PSRs to be studied.

Secondly, our confidence in HORUS denoising could be further improved by quantitatively assessing its performance on downstream tasks such as crater counting and elevation modelling. Being able to show improvement on these tasks would be a powerful indicator of its performance. Furthermore, given the ill-posed nature of the underlying inversion problem, adding uncertainty estimation to the outputs of HORUS (for example through the use of GANs or Bayesian neural networks) could help us assess our confidence in its denoising.

Finally, many further refinements of the HORUS algorithm are possible. For example, transfer learning could be used to fine-tune HORUS on a specific task, such as imaging transition zones between PSRs and sunlit regions (which are particularly relevant for future exploration missions), by carefully selecting its training images and scenes. Whilst we use standard convolutional architectures, more complex architectures may improve performance. It would also be interesting to study how transferable HORUS is to other CCD cameras, for example those carrying out low-light imaging of other celestial bodies, or even those used on Earth.

4.6 Summary

We have shown that PIML can carry out inversion using real data. HORUS was able to significantly enhance shadowed images of the lunar surface and reveal small-scale surface features within them for the first time, even though these images were dominated by complex real-world noise. This algorithm will likely have significant impact in the lunar science and exploration community by allowing us to visualise PSRs in unprecedented detail. However, the complexity of the dataset and its real-world noise meant that it was challenging to generate realistic training data and ensure HORUS generalised well to real images, and to validate its denoised images. Future work is needed to assess the uncertainty in HORUS' outputs and its applicability to other low-light denoising environments.

5

Physics-informed neural networks for wave simulation

5.1 Introduction

The previous two chapters focused on the scalability of PIML approaches for estimating underlying physical principles and quantities from real data. In this chapter we instead focus on the forward problem and consider the question, can PIML algorithms simulate complex physical phenomena? More specifically, we propose and assess the scalability of various approaches for simulating increasingly complex seismic wave phenomena.

Simulating seismic waves is essential for many real-world applications, from modelling earthquakes to carrying out seismic exploration. Yet, in many cases it is notoriously difficult. Depending on the problem, many complex phenomena can occur, including reflected, transmitted and grazing waves at interfaces in the medium, wavefront compression, expansion and healing through different velocity regions, multiple types of waves interfering simultaneously and a multi-scale range of amplitudes and frequencies [Igel, 2017]. Often, sophisticated numerical solvers with elaborate implementations are required to incorporate these phenomena. Furthermore they are usually very computationally expensive, requiring supercomputers to run.

Given its complexity, simulating seismic waves is an excellent task for investigating how well PIML-based simulation techniques scale to more complex physical phenomena. One good place to start is the modelling of a point source in a homogeneous medium, which has a simple analytical solution. Then, seismic waves in more Earth-realistic, strongly heterogeneous media could be modelled where more complex phenomena can occur, which is significantly harder to model.

In this chapter we present, evaluate and discuss three different deep learning and PIML approaches for simulating seismic waves in increasingly complex media. The first approach uses a deep neural network with a physics-informed WaveNet architecture [van den Oord et al., 2016] to simulate the seismic waves generated by a point source propagating in simple 2D horizontally layered acoustic media. The second approach uses a deep neural network with a conditional encoder-decoder architecture to simulate the same problem, but for more realistic media which contain geological faults and for a variable source location. The final approach uses a fully-connected neural network with a physics-informed loss function to simulate the entire seismic wavefield in both simple and strongly heterogeneous Earth-realistic 2D acoustic media.

We find that all the approaches tested are able to accurately simulate seismic waves within their respective media. Furthermore, once trained all of the networks are orders of magnitude faster than finite difference modelling. This suggests that they can provide useful alternatives to traditional numerical solvers when simulating real-world problems.

However, significant challenges are identified when scaling our methods to more complex and realistic tasks. Firstly, for the first two approaches, significantly longer training times, more training data and more free parameters in the neural networks are required in order to accurately simulate the seismic response as the complexity of the Earth model increases. Secondly, the accuracy of these approaches declines significantly when simulating Earth models outside of their training distribution, i.e. they do not provide a general simulation approach. Thirdly, whilst the third approach is able to model high amplitude phenomena such as direct arrivals, it

struggles to model lower amplitude phenomena such as secondary reflections. We discuss each of these challenges further.

This chapter is divided into five remaining parts. In Section 5.2 we provide relevant background on seismic simulation and machine learning in geophysics. Next, in Sections 5.3, 5.4 and 5.5 we present and evaluate each of the three approaches for simulating seismic waves described above. Finally in Section 5.6 we discuss the implications of all our approaches and the challenges encountered when scaling them to more complex and realistic tasks.

5.2 Background

5.2.1 Real-world seismic simulation

Seismic simulations are vital for many different geophysical applications. In seismic hazards analysis they are used to quantify the ground motion of potential earthquakes [Boore, 2003, Cui et al., 2010], in oil and gas prospecting they allow hydrocarbon reservoirs to be characterised and modelled [Chopra and Marfurt, 2007, Lumley, 2001], in global geophysics they allow us to obtain snapshots of the Earth's interior dynamics [Hosseini et al., 2019, Bozdağ et al., 2016] and in planetary science they play a central role in understanding the seismicity of Mars and other planets [Van Driel et al., 2019, Stähler et al., 2018].

Most real-world applications require the simulation of seismic waves in realistic, fully heterogeneous 3D Earth models where many complex physical phenomena can occur. In this setting, the most popular approaches are finite difference (FD) and spectral element methods (SEM) [Igel, 2017, Moczo et al., 2007, Komatitsch and Tromp, 2002a,b]. These approaches are able to capture a large range of physics, including the effects of undulating solid-fluid interfaces [Leng et al., 2019], intrinsic attenuation [van Driel and Nissen-Meyer, 2014a] and anisotropy [van Driel and Nissen-Meyer, 2014b]. Such methods are usually considered "exact" (i.e. they are capable of modelling the full range of physics) so long as their discretisation errors are kept small.

However, ensuring this becomes extremely computationally demanding as the complexity of the simulation increases. In particular, the number of discrete

elements required increases significantly as the required simulation frequency increases (typically $\propto \omega^4$ for 3D simulation). The Earth's interior has a multi-scale nature and thus requires high frequency simulations; for realistic applications billions of elements may be required which leads to high computational costs. For example, in global seismology one may be interested in modelling waves up to 1 Hz in frequency to resolve small-scale heterogeneities in the mantle and a single simulation of this type with conventional techniques can cost around 40 million CPU hours [Leng et al., 2019]. When carrying out seismic tomography after a seismic survey, seismic waves up to tens of Hertz in frequency may need to be modelled hundreds of thousands of times for each explosion in the survey, and such requirements can easily fill the largest supercomputers on Earth [Tromp et al., 2005].

Reducing the computational costs of 3D methods is therefore a major area of research and many approaches have been developed. One strategy is to employ hardware accelerators such as GPUs and multi-node parallelisation [Rietmann et al., 2012, Bohlen, 2002]. Another is to use algorithmic optimisations such as improved time schemes [Rietmann et al., 2015, Ma et al., 2014] and irregular meshing [Peter et al., 2011, Pelties et al., 2012]. However, whilst both strategies can offer modest speedups they typically do not drastically change the scaling of computational cost with frequency. Larger computational savings can be achieved by introducing fundamental approximations into the simulation, for example that the Earth is spherically symmetric [Friederich and Dalkolmo, 1995, Kawai et al., 2006], axisymmetric [Toyokuni and Takenaka, 2006, Nissen-Meyer et al., 2014], or has lateral smoothness [Leng et al., 2019]. With increasing frequency such methods can become several orders of magnitude faster than full 3D methods. For simple 2D and 1D Earth models, pseudo-analytical techniques such as ray tracing and amplitude-versus-offset modelling can be very efficient and effective [Aki and Richards, 1980, Vinje et al., 1993]. Thus, there appears to be a clear compromise between the level of approximation of the physical system and the computational cost of the simulation algorithm.

Machine learning may help to alleviate this issue, by offering different approaches for simulation. For example, we note that for many real-world applications we are

only interested in understanding the seismic response recorded at the surface of an Earth model (for example in seismic imaging or earthquake modelling), yet fully numerical methods still need to solve for the wavefield through the entire Earth model in order to obtain this response. In the first two approaches presented below we instead use machine learning to learn a direct mapping from the initial conditions of the simulation to the surface response without needing to compute the full wavefield, which leads to potentially large efficiency savings. At a higher level, machine learning opens up the possibility to learn from previous simulations, which may allow more powerful computational shortcuts to be taken when carrying out new simulations.

5.2.2 Related work

The use of machine learning and neural networks in geophysics is in general not new [Van Der Baan and Jutten, 2000]. For example, Murat and Rudman [1992] used neural networks to carry out automated first break picking, Dowla et al. [1990] used a neural network to discriminate between earthquakes and nuclear explosions and Poulton et al. [1992] used them for electromagnetic inversion of a conductive target. In seismic inversion, Röth and Tarantola [1994] used a neural network to estimate the velocity of 1D, layered, constant thickness velocity profiles from seismic amplitudes and Nath et al. [1999] used neural networks for cross-well travel-time tomography. However, these early approaches only used shallow network designs with small numbers of free parameters which limits the expressivity of neural networks and the complexity of problems they can learn about [Goodfellow et al., 2016].

Driven by more recent advances in machine learning, a resurgence is occurring [Bergen et al., 2019, Kong et al., 2019]. Early examples include Devilee et al. [1999], who used deep probabilistic neural networks to estimate crustal thicknesses from surface wave velocities and Valentine and Trampert [2012] who used a deep autoencoder to compress seismic waveforms. More recent examples include Perol et al. [2018] who presented an earthquake identification method using convolutional networks which is orders of magnitude faster than traditional techniques, Araya-Polo et al. [2018], Wu and Lin [2018], Yang and Ma [2019] who proposed various concepts

and deep learning techniques for carrying out seismic inversion and Sun and Demanet [2018] who showed deep learning could extrapolate low frequencies in recorded seismic data to improve the convergence of full waveform inversion (FWI) algorithms.

For seismic simulation, Zhu et al. [2017] presented a multi-scale convolutional network for predicting the evolution of the full seismic wavefield in heterogeneous media. Their method was able to approximate wavefield kinematics over multiple time steps, although it suffered from the accumulation of error over time and did not offer a reduction in computational time. Siahkoohi et al. [2019] showed that FD simulation could be accelerated by using a hybrid approach which combined convolutional neural networks with low fidelity FD time steps. Krischer and Fichtner [2017] used a generative adversarial network to simulate seismograms from radially symmetric and smooth Earth models. Yang et al. [2021b] showed that Fourier neural operators could provide accurate surrogate models for carrying out seismic simulation and inversion.

Recent work has also investigated the use of physics-informed neural networks (PINNs) for simulating seismic waves. Smith et al. [2021] presented a PINN to predict the travel times of waves in 3D heterogeneous media by solving the Eikonal equation. Karimpouli and Tahmasebi [2020] showed PINNs could solve the 1D time-dependent wave equation in a constant velocity medium and simultaneously estimate the velocity of the medium. Rasht-Behesht et al. [2021] took a similar approach, but extending to 2D heterogeneous acoustic media, showing that PINNs could invert for ellipsoidal and checkerboard velocity models given the seismic response from sources placed within these models. Xu et al. [2019a] similarly used PINNs to carry out 2D seismic inversion, whilst Shukla et al. [2020] used them for ultrasound inversion. Rao et al. [2021] presented a PINN for solving the elastic wave equation in 2D homogeneous media, whilst Song et al. [2021] used a PINN to simulate seismic waves in 2D and 3D aniostropic acoustic media using a scattered wavefield formulation in the frequency domain.

In contrast to the works above, in this work we present three distinct PIML approaches for carrying out seismic simulation, and discuss the advantages and disadvantages between them. Furthermore, whilst the works above mostly focus on

solving simplified toy problems, we place focus on assessing how well our approaches scale to more complex, real-world simulation tasks. For the PINN tested specifically, we present novel ways to train the network using curriculum learning and condition the PINN on the source location.

5.2.3 Acoustic wave equation

In this work we focus on simulating seismic waves in fully heterogeneous media. For example, one could consider simulating the seismic waves generated from a point source (e.g. an earthquake or explosion) within an Earth model. Only the compressional (acoustic) waves generated are considered and we only consider 2D media. The extension of our approaches to 3D simulation, as well as the inclusion of more complex wave physics (such as elastic waves), is discussed in Section 5.6.

In this case the system can be described by the acoustic wave equation, given by

$$\rho \nabla \cdot \left(\frac{1}{\rho} \nabla u \right) - \frac{1}{c^2} \frac{\partial^2 u}{\partial t^2} = -\rho \frac{\partial^2 f}{\partial t^2} \ , \tag{5.1}$$

where $u(x,t)$ describes the pressure of the seismic waves generated, $f(x,t)$ is a source term which describes the strength and duration of the source, $c(x) = \sqrt{\kappa/\rho}$ is the velocity of the medium, $\rho(x)$ is the density of the medium, $\kappa(x)$ is the adiabatic compression modulus, $u, f, c, \rho, \kappa, t \in \mathbb{R}^1$ and $x \in \mathbb{R}^d$ where d is the number of spatial dimensions ($d = 2$ in this case) [Long et al., 2013]. In general both the density and velocity of the medium can vary spatially. When the density of the medium is constant and the source term is negligible Equation 5.1 reduces to the canonical form of the wave equation, given by

$$\nabla^2 u - \frac{1}{c^2} \frac{\partial^2 u}{\partial t^2} = 0 \ . \tag{5.2}$$

5.3 Fast seismic simulation in 2D horizontally layered acoustic media using WaveNet

5.3.1 Overview

In this section we present and evaluate a deep neural network with a physics-informed WaveNet architecture [van den Oord et al., 2016] for simulating seismic waves in

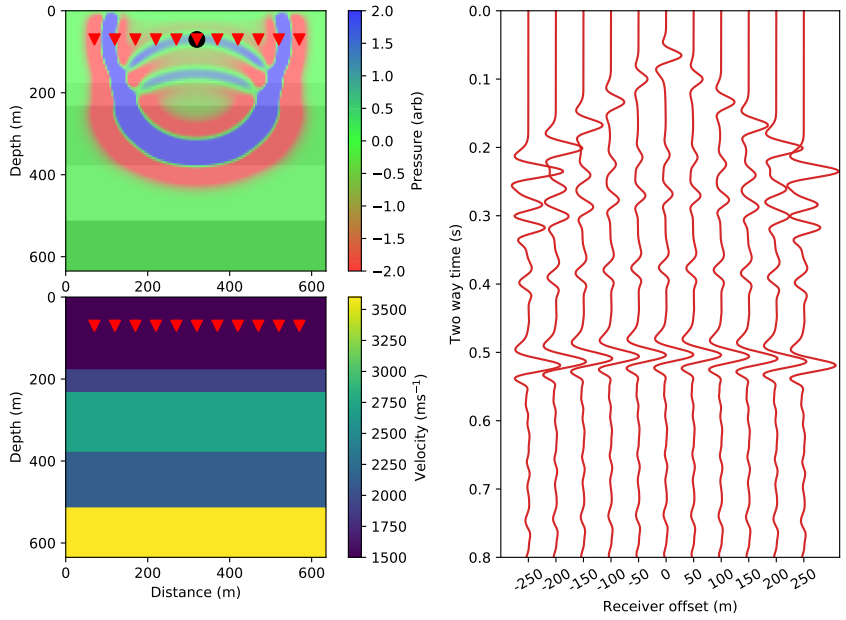


Figure 5.1: Ground truth FD simulation example. Left, top: A 20 Hz Ricker seismic source is emitted close to the surface and propagates through a 2D horizontally layered acoustic Earth model. The black circle shows the source location. 11 receivers are placed at the same depth as the source with a horizontal spacing of 50 m (red triangles). The full wavefield is overlain for a single snapshot in time. Note seismic reflections occur at each velocity interface. Left, bottom: The Earth velocity model. The Earth model has a constant density of 2200 kgm^{-2}. Right: The resulting ground truth pressure response recorded by each of the receivers, using FD modelling. A $t^{2.5}$ gain is applied to the receiver responses for display.

simple 2D horizontally layered acoustic Earth models. The goal of this section is to assess whether deep learning and PIML can be used to simulate seismic waves for a simplified task, before attempting to design approaches which scale to more realistic and complex tasks in Sections 5.4 and 5.5.

More specifically, we train the network to simulate the seismic response recorded at multiple receiver locations on the surface of an Earth model from a point source emitted at the surface, given the Earth's velocity model as an input. The network is trained using many example simulations from FD simulation. An example simulation we wish to learn is shown in Figure 5.1.

As mentioned above, many real-world seismic applications are concerned with sparse surface observations similar to this setup. A key difference of this approach compared to FD and SEM simulation is that, once trained, the network computes the

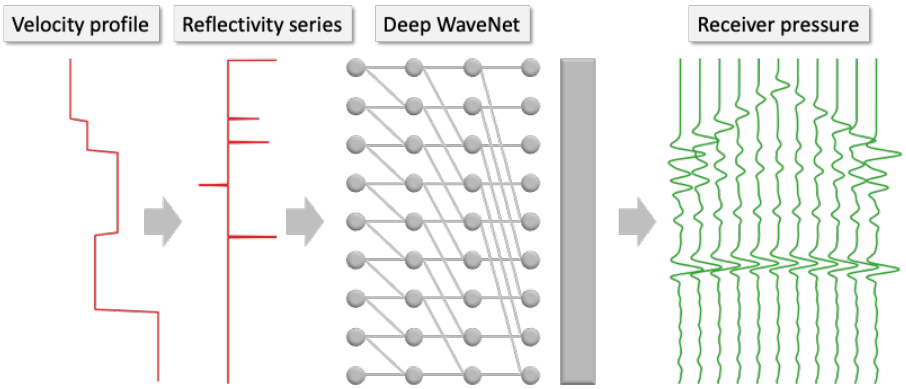


Figure 5.2: Our WaveNet simulation workflow. Given a 1D Earth velocity profile as input (left), our WaveNet deep neural network (middle) outputs a simulation of the pressure responses at the 11 receiver locations in Fig 5.1. The raw input 1D velocity profile sampled in depth is converted into its normal incidence reflectivity series sampled in time before being input into the network. The network is composed of 9 time-dilated causally-connected convolutional layers with a filter width of 2 and dilation rates which increase exponentially with layer depth. Each hidden layer of the network has same length as the input reflectivity series, 256 channels and a ReLU activation function. A final causally-connected convolutional layer with a filter width of 101 samples, 11 output channels and an identity activation is used to generate the output simulation.

seismic response at the surface in a single inference step without needing to iteratively model the seismic wavefield through time, which could offer a significant speed-up.

We will now discuss our simulation workflow and training methodology in more detail below.

5.3.2 Simulation workflow

Our simulation workflow is shown in Figure 5.2. The input to the workflow is a horizontally layered velocity profile and its output is a prediction of the pressure response recorded at each receiver location. The workflow consists of two distinct steps. First, a pre-processing step is carried out, where we convert each input velocity model sampled in depth into its corresponding normal incidence reflectivity series sampled in time (Figure 5.2, left). Second, a simulation step is carried out, where the reflectivity series is passed to the WaveNet network to simulate the pressure response recorded by each receiver sampled in time (Figure 5.2, right).

5.3.2.1 Reflectivity series

The reflectivity series is typically used in exploration seismology [Russell, 1988] and contains the values of the ratio of the amplitude of the reflected wave to the incident wave that occurs at each interface in a velocity model. For acoustic waves at normal incidence, these values are given by

$$R = \frac{\rho_2 v_2 - \rho_1 v_1}{\rho_2 v_2 + \rho_1 v_1} \ , \tag{5.3}$$

where ρ_1,v_1 and ρ_2,v_2 are the densities and P-wave velocities across the interface. In our workflow we represent the reflectivity series as a time series, where the time of each reflectivity value is defined as the time at which the primary reflection of the source from the corresponding velocity interface arrives at the zero-offset (normal incidence) receiver in Figure 5.1. These arrival times are computed by performing a standard 1D depth-to-time conversion of the input velocity model.

5.3.2.2 WaveNet

The WaveNet network was originally proposed by van den Oord et al. [2016] for text-to-speech synthesis and it is essentially a fully convolutional deep neural network with a few alterations made to better honour time series prediction tasks which are causal and occur over multiple time scales. Each convolutional layer is made causal; that is, the receptive field of each neuron only contains samples from the input whose sample times are before or the same as the neuron's sample time (see Figure 5.2, middle). Furthermore the width of each causal connection increases exponentially with the depth of each layer. This allows the field of view of the neurons to increase exponentially with layer depth, without needing a large number of layers.

5.3.2.3 Physics-informed architecture

We chose to convert the velocity model to its reflectivity series and use the causal WaveNet architecture to physically constrain our workflow. For horizontally layered velocity models and receivers horizontally offset from the source, the receiver pressure recordings are causally correlated to the normal incidence reflectively series of the

zero-offset receiver. Intuitively, a seismic reflection recorded after a short time has only travelled through a shallow part of the velocity model and the pressure responses are at most dependent on the past samples in this reflectivity series. We build this causality assumption into our workflow as a hard constraint by pre-processing the input velocity model into its corresponding reflectivity series and using the causal WaveNet architecture to simulate the receiver response.

5.3.2.4 Implementation details

The input to our simulation workflow is the 1D profile of a 2D horizontally layered velocity model, with a depth of 640 m and a step size of 5 m. We use Equation 5.3 and a standard 1D depth-to-time conversion to convert the velocity model into its normal incidence reflectivity series. The output reflectivity series has a length of 1 s and a sample rate of 2 ms. An example output reflectivity series is shown in Figure 5.2 (left).

The reflectivity series is passed to the WaveNet network, which contains 9 causally-connected convolutional layers (Figure 5.2, middle). Each convolutional layer has the same length as the input reflectivity series, 256 hidden channels, a receptive field width of 2 samples and a Rectified Linear Unit (ReLU) activation function [Nair and Hinton, 2010]. Similar to the original WaveNet design, we use exponentially increasing dilations at each layer to ensure that the first sample in the input reflectivity series is in the receptive field of the last sample of the output simulation. We add a final causally-connected convolutional layer with 11 output channels, a filter width of 101 samples and an identity activation to generate the output simulation, where each output channel corresponds to a receiver prediction. This results in the network having 1,333,515 free parameters in total.

5.3.3 Training data generation

To train the network, we generate 50,000 synthetic ground truth example simulations using the SEISMIC_CPML code, which performs 2^{nd}-order acoustic FD modelling [Komatitsch and Martin, 2007]. Each example simulation uses a randomly sampled 2D horizontally layered velocity model with a width and depth of 640 m and a

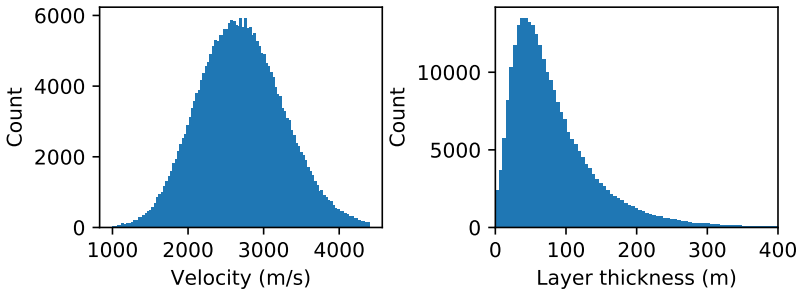


Figure 5.3: Distribution of layer velocity and layer thickness over all examples in the training set.

sample rate of 5 m in both directions. (Figure 5.1, bottom left). For all simulations we use a constant density model of 2200 kgm^{-2}.

In each simulation the layer velocities and layer thickness are randomly sampled from log-normal distributions. We also add a small velocity gradient randomly sampled from a normal distribution to each model such that the velocity values tend to increase with depth, to be more Earth-realistic. The distributions over layer velocities and layer thicknesses for the entire training set are shown in Figure 5.3.

We use a 20 Hz Ricker source emitted close to the surface and record the pressure response at 11 receiver locations placed symmetrically around the source, horizontally offset every 50 m (Figure 5.1, top left). We use a convolutional perfectly matched layer boundary condition such that waves which reach the edge of the model are absorbed with negligible reflection. We run each simulation for 1 s and use a 0.5 ms sample rate to maintain accurate FD fidelity. We downsample the resulting receiver pressure responses to 2 ms before using them for training.

We run 50,000 simulations and extract a training example from each simulation, where each training example consists of a 1D layered velocity profile and the recorded pressure response at each of the 11 receivers. We withhold 10,000 of these examples as a validation set to measure the generalisation performance of the network during training.

5.3.4 Training process

The network is trained using the Adam stochastic gradient descent algorithm [Kingma and Ba, 2015]. This algorithm computes the gradient of a loss function with respect

to the free parameters of the network over a randomly selected subset, or batch, of the training examples. This gradient is used to iteratively update the parameter values, with a step size controlled by a learning rate parameter. We propose a L2 loss function with time-varying gain function for this task, given by

$$L = \frac{1}{N}\|G(\hat{Y} - Y)\|_2^2 \,, \tag{5.4}$$

where $\hat{Y}$ is the simulated receiver pressure response from the network, Y is the ground truth receiver pressure response from FD modelling and N is the number of training examples in each batch. The gain function G has the form $G = t^g$ where t is the sample time and g is a hyperparameter which determines the strength of the gain. We add this to empirically account for the attenuation of the wavefield caused by spherical spreading, by increasing the weight of samples at later times. In this Section we use a fixed value of $g = 2.5$. We use a learning rate of $1\text{x}10^{-5}$, a batch size of 20 training examples and run training over 500,000 gradient descent steps.

5.3.5 Comparison to 2D ray tracing

We compare the WaveNet simulation to an efficient, quasi-analytical 2D ray-tracing algorithm which assumes horizontally layered media. We modify the 2D horizontally layered ray-tracing bisection algorithm from the CREWES seismic modelling library [Margrave and Lamoureux, 2018] to include Zoeppritz modelling of the reflection and transmission coefficients at each velocity interface [Aki and Richards, 1980] and 2D spherical spreading attenuation [Gutenberg, 1936, Newman, 1973] during ray tracing. The output of the algorithm is a primary reflectivity series for each receiver, which we convolve with the source signature used in FD modelling to obtain an estimate of the receiver responses.

5.3.6 Results

Whilst training the WaveNet the losses over the training and validation datasets converge to similar values, suggesting the network is generalising well to examples in the validation dataset. To assess the performance of the trained network, we

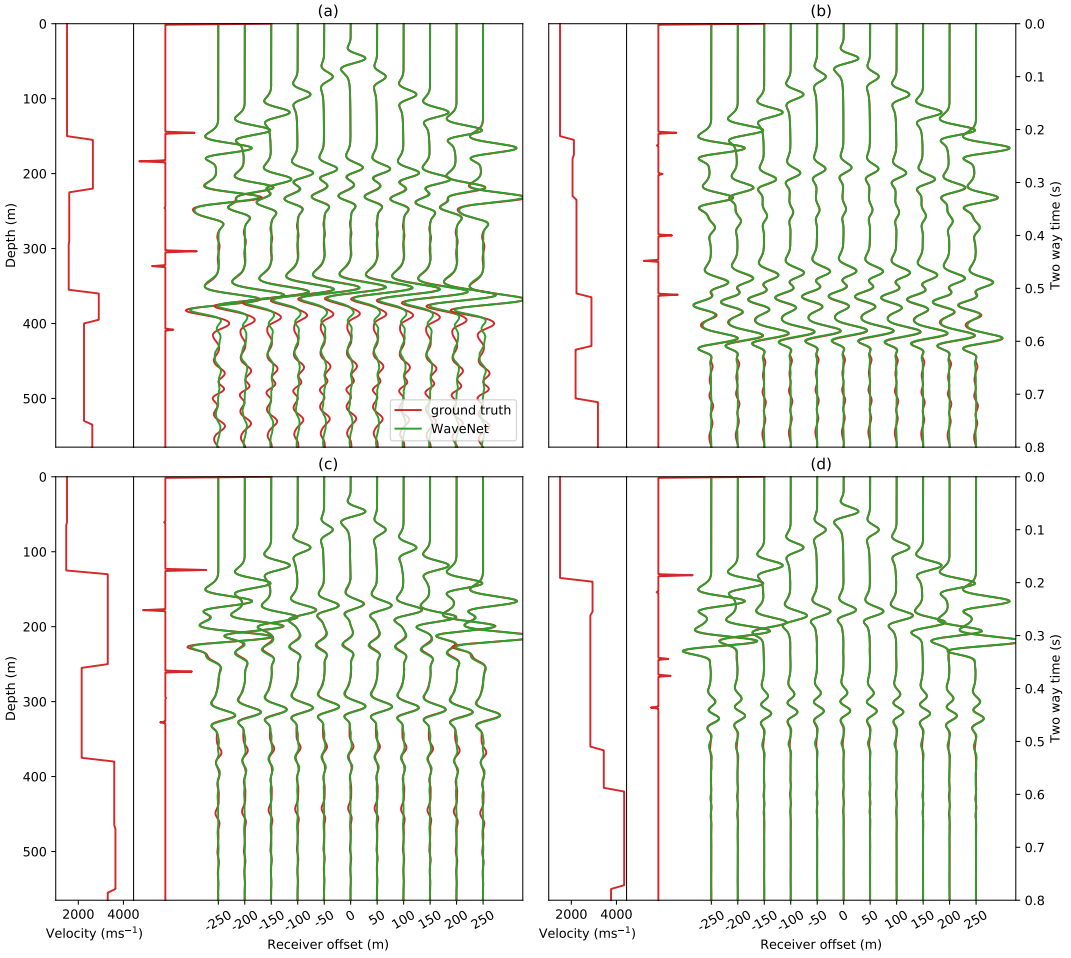


Figure 5.4: WaveNet simulations for 4 randomly selected examples in the test set. Red shows the input velocity model, its corresponding reflectivity series and the ground truth pressure response from FD simulation at the 11 receiver locations. Green shows the WaveNet simulation given the input reflectivity series for each example. A $t^{2.5}$ gain is applied to the receiver responses for display.

generate a random test set of 1000 unseen examples. The simulations for 4 randomly selected examples from this test set are compared to the ground truth FD modelling simulation in Figure 5.4. We also compare the WaveNet simulation to 2D ray tracing in Figure 5.5. For nearly all time samples the network is able to simulate the receiver pressure responses. The WaveNet is able to predict the Normal Moveout (NMO) of the primary layer reflections with receiver offset, the direct arrivals at the start of each receiver recording and the spherical spreading loss of the wavefield over time, though the network struggles to accurately simulate the multiple reverberations at the end of the receiver recordings.

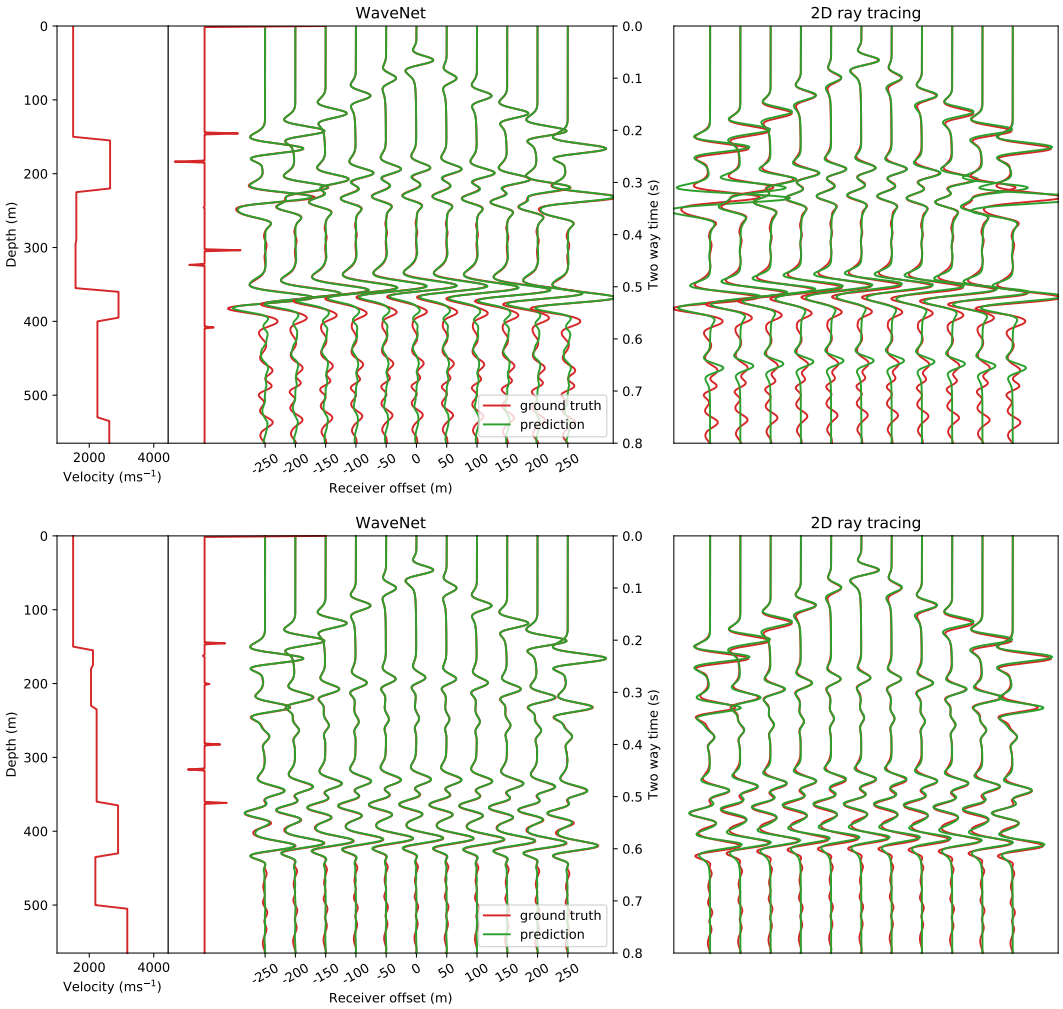


Figure 5.5: Comparison of WaveNet simulation to 2D ray tracing. We compare the WaveNet simulation to 2D ray tracing for 2 of the examples in Fig 5.4. Red shows the input velocity model, its corresponding reflectivity series and the ground truth pressure responses from FD simulation. Green shows the WaveNet simulation (left) and 2D ray tracing simulation (right). A $t^{2.5}$ gain is applied to the receiver responses for display.

We plot the histogram of the average absolute amplitude difference between the ground truth FD simulation and the simulation from the WaveNet and 2D ray tracing over the test set in Figure 5.6 (bottom right) and observe that the WaveNet simulation has a lower average amplitude difference than 2D ray tracing. Small differences in phase and amplitude at larger offsets are the main source of discrepancy between the 2D ray tracing and FD simulation, which can be seen in Figure 5.5, and are likely due to errors both in the ray tracing approximation and in using discretisation in the FD simulation. The WaveNet predictions are consistent and

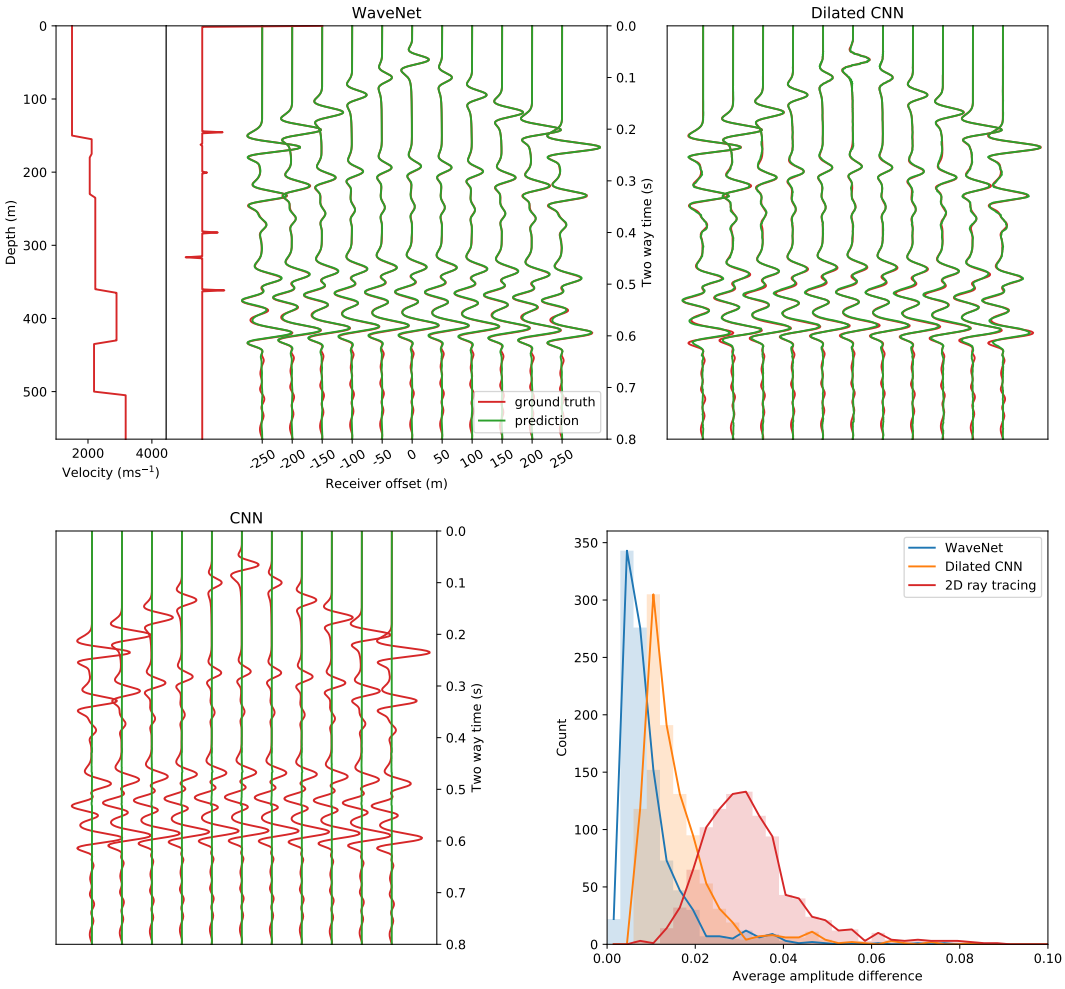


Figure 5.6: Comparison of different network architectures on simulation accuracy. Top left shows the WaveNet simulated pressure response for a randomly selected example in the test set (green) compared to ground truth FD simulation (red). Top right and bottom left show the simulated response when using two convolutional network designs with and without exponential dilations. Bottom right shows the histogram of the average absolute amplitude difference between the ground truth FD simulation and the simulations from the WaveNet, the dilated convolutional network and 2D ray tracing over the test set of 1000 examples. A $t^{2.5}$ gain is applied to the receiver responses for display.

stable across the test set, and their closer amplitude match to the FD simulation is perhaps to be expected because the network is trained to directly match the FD simulation, rather than the 2D ray tracing.

We compare the sensitivity of the network's accuracy to two different convolutional network designs in Figure 5.6. Their main differences to the WaveNet design is that both networks use standard rather than causal convolutional layers and the second network uses exponential dilations whilst the first does not. Both networks have 9

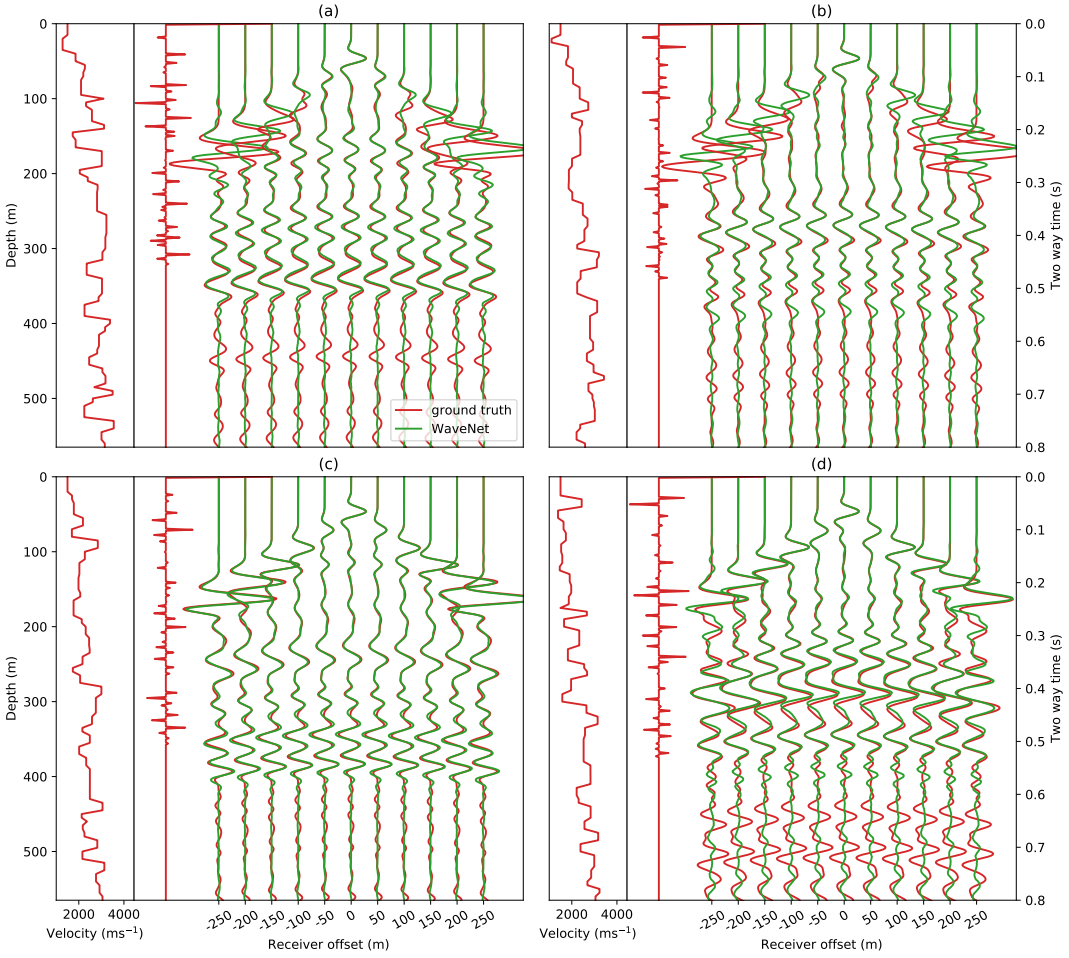


Figure 5.7: Generalisation ability of the WaveNet. The WaveNet simulations (green) for 4 velocity models with a much smaller average layer thicknesses than the training distribution are compared to ground truth FD simulation. Red shows the input velocity model, its corresponding reflectivity series and the ground truth pressure responses from FD simulation.

convolutional layers, each with 256 hidden channels, filter sizes of 3, ReLU activations for all hidden layers and an identity activation function for the output layer, with 1,387,531 free parameters in total. We observe that the convolutional network without dilations does not converge during training, whilst the dilated convolutional network has a higher average absolute amplitude difference over the test set from the ground truth FD simulation than the WaveNet network (Figure 5.6 (bottom right)).

The generalisation ability of the WaveNet outside of its training distribution is tested in Figure 5.7. We generate four velocity models with a much smaller average layer thickness than the training set and compare the WaveNet simulation to the

Method	Average CPU time (s)	Average GPU time (s)	Training time (days)
2D FD simulation	73 ± 1 (1x)	-	-
2D ray tracing	2.2 ± 0.1 (33x)	-	-
WaveNet (forward)	3.79 ± 0.03 (19x)	0.133 ± 0.001 (549x)	0.5
Conditional encoder-decoder	3.3 ± 0.1 (22x)	0.180 ± 0.003 (406x)	4

Table 5.1: Speed comparison of simulation methods. The time shown is the average time taken to generate 100 simulations on either a single core of a 2.2 GHz Intel Core i7 processor or a NVIDIA Tesla K80 GPU. The speed up factor compared to FD simulation is shown in brackets.

ground truth FD simulation. We find that the WaveNet is able to make an accurate prediction of the seismic response, but it struggles to simulate the multiple reflections and sometimes the interference between the direct arrival and primary reflections.

We compare the average time taken to generate 100 simulations to FD simulation and 2D ray tracing in Table 5.1. We find that on a single CPU core the WaveNet is 19 times faster than FD simulation, and using a GPU and the TensorFlow library [Abadi et al., 2016] it is 549 times faster. This speedup is likely to be higher than if the GPU was used for accelerating existing numerical methods [Rietmann et al., 2012]. In this case, the specialised 2D ray tracing algorithm offers a similar speed up to the WaveNet network. The network takes approximately 12 hours to train on one NVIDIA Tesla K80 GPU, although this training step is only required once and subsequent simulation steps are fast.

5.4 Fast seismic simulation in 2D faulted acoustic media using a conditional encoder-decoder

5.4.1 Overview

The WaveNet architecture we implemented above is limited in that it is only able to simulate seismic waves in simple horizontally layered Earth models. In this section we present and evaluate a second network which is significantly more general; it simulates seismic waves in 2D faulted acoustic media with arbitrary layers, fault properties and an arbitrary location of the seismic source on the surface of the media.

Similar to Section 5.3, the task is to simulate the seismic response recorded at multiple receiver locations on the surface of the Earth model from a point source

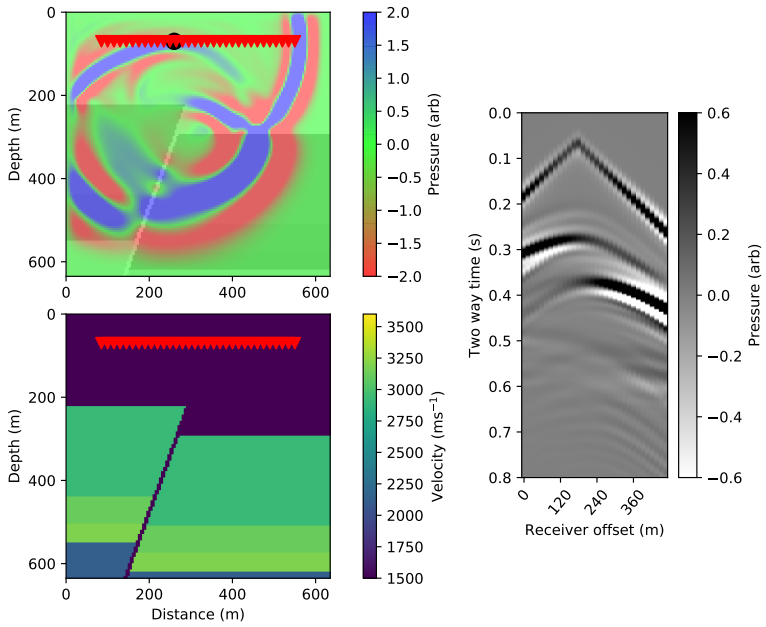


Figure 5.8: Ground truth FD simulation example, with a 2D faulted media. Left, top: The black circle shows the source location. 32 receivers are placed at the same depth as the source with a horizontal spacing of 15 m (red triangles). The full wavefield pressure is overlain for a single snapshot in time. Left, bottom: The Earth velocity model. Right: The resulting ground truth pressure response recorded by each receiver, using FD modelling. A $t^{2.5}$ gain is applied to the receiver responses for display.

emitted at the surface, given the Earth's velocity model as an input. In this case the source location can vary along the surface and is given as an additional input. The network is trained using many example simulations from FD simulation. An example simulation we wish to learn is shown in Figure 5.8.

This is a much more challenging task to learn for multiple reasons. Firstly, the velocity model varies along both dimensions and the resulting seismic wavefield has more complex kinematics than the wavefields in horizontally layered media. Secondly, we add the source location as an input and attempt to generalise across this initial condition too. Thirdly, we input the velocity model directly into the network without conversion to a reflectivity series beforehand; the network must learn to carry out its own depth to time conversion to simulate the receiver responses. We chose this approach rather than pre-processing the velocity model into a reflectively series beforehand because we note that for non-horizontally layered media the pressure responses are in general not causally correlated to the normal incidence reflectivity

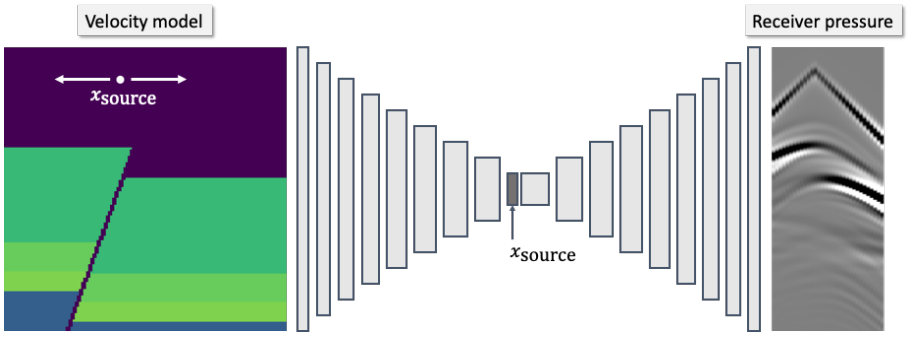


Figure 5.9: Our conditional encoder-decoder simulation workflow. Given a 2D velocity model and source location as input, a conditional encoder-decoder network outputs a simulation of the pressure responses at the receiver locations in Figure 5.8. The network is composed of 24 convolutional layers and concatenates the input source location with its latent vector.

series and our previous causality assumption does not hold.

We will now discuss our simulation workflow and training methodology in more detail below.

5.4.2 Simulation workflow

Our simulation workflow is shown in Figure 5.9. Instead of pre-processing the input velocity model to its associated reflectivity series, we input the velocity model directly into the network. The network is conditioned on the source position, which is allowed to vary along the surface of the Earth model. The output of the network is a simulation of the pressure responses recorded at 32 fixed receiver locations in the model shown in Figure 5.8.

5.4.2.1 Conditional encoder-decoder design

We use a conditional encoder-decoder network design, shown in Fig 5.9. This network compresses the velocity model into a set of latent features by slowly removing the spatial dimensions of the velocity model, before decompressing the latent features by adding a time dimension to infer the receiver responses. The idea is that this design helps the network learn the depth-to-time conversion required between the velocity model and the receiver responses. In addition, the input

source location is concatenated onto the latent vector in order to condition the network on this initial condition.

The network is composed of 10 convolutional layers which reduce the spatial dimensions of the input velocity model until it has a 1x1 shape with 1024 hidden channels. The input source surface position is concatenated onto this latent vector and 14 convolutional layers are used to expand the size of the latent vector until its output shape is the same as the target receiver gather. All hidden layers use ReLU activation functions and the final output layer uses an identity activation function. The resulting network has 18,382,296 free parameters. The full parameterisation of the network is shown in Table C.1.

5.4.3 Training process

We use the same training data generation process described by Section 5.3.3. When generating velocity models, we add a fault to the model. We randomly sample the length, normal or reverse direction, slip distance and orientation of the fault. Example velocity models drawn from this process are shown in Figure 5.10. We generate 100,000 example velocity models and for each model chose three random source locations along the top of the model. This generates a total of 300,000 synthetic ground truth example simulations to use for training the network. We withhold 60,000 of these examples to use as a validation set during training.

We train using the same training process and loss function described in Section 5.3.4, except that we employ a L1 norm instead of a L2 norm in the loss function (Equation 5.4). We use a learning rate of $1\text{x}10^{-4}$, a batch size of 100 examples and run training over 3,000,000 gradient descent steps. We use batch normalisation [Ioffe and Szegedy, 2015] after each convolutional layer to help regularise the network during training.

5.4.4 Results

During training the losses over the training and validation datasets converge to similar values and we test the performance of the trained network using a test set of 1000

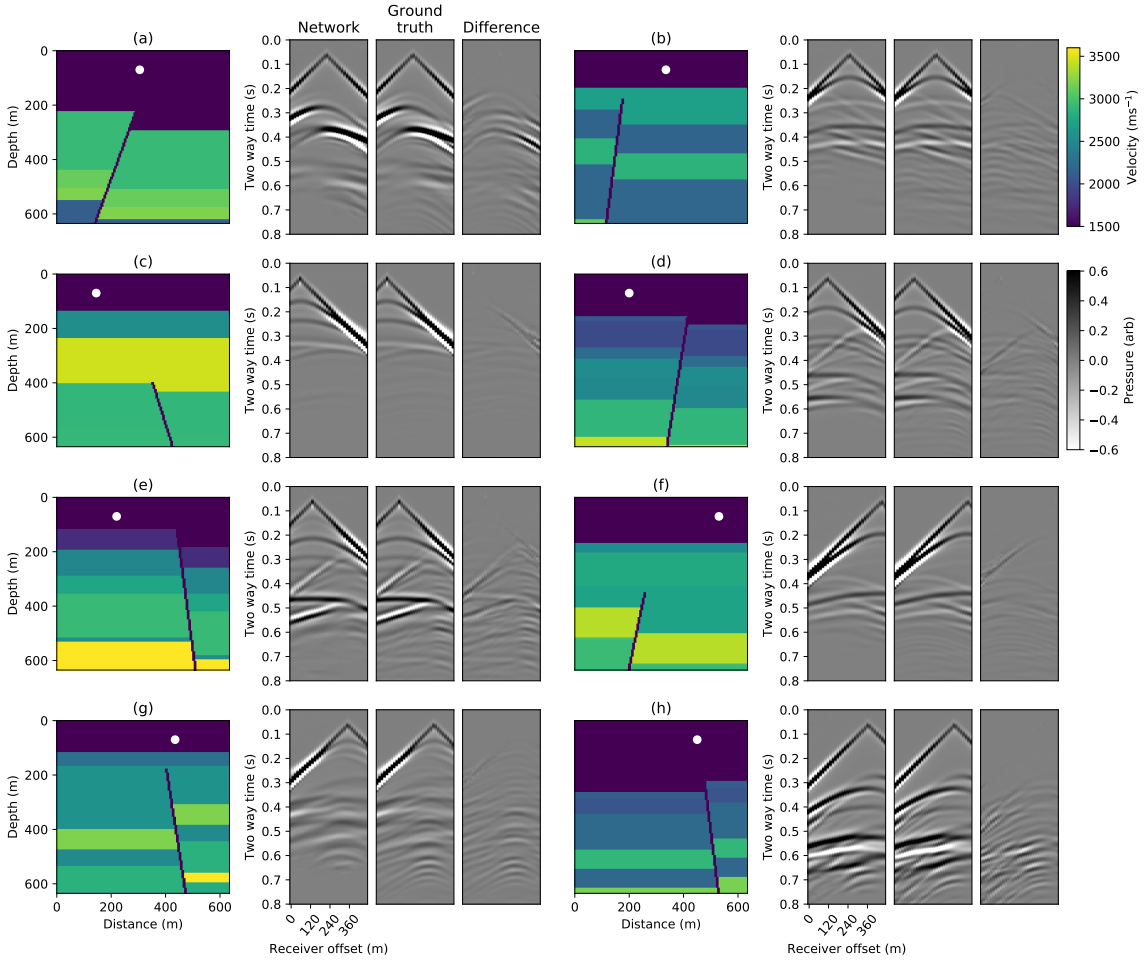


Figure 5.10: Conditional encoder-decoder simulations for 8 randomly selected examples in the test set. White circles show the input source location. The left simulation plots show the network predictions, the middle simulation plots show the ground truth FD simulations and the right simulation plots show the difference. A $t^{2.5}$ gain is applied for display.

unseen examples. The output simulations for 8 randomly selected velocity models and source positions from this set are shown in Figure 5.10. We observe that the network is able to simulate the kinematics of the primary reflections and in most cases is able to capture their relative amplitudes. We also plot the network simulation when varying the source location over 2 velocity models from the test set in Figure 5.11 and find that the network is able to generalise well over different source locations.

We test the accuracy of the simulation when using different network designs and training hyperparameters, shown in Figure 5.13. We compare example simulations from the test set when using our baseline conditional encoder-decoder network, when halving the number of hidden channels for all layers, when using a L2 loss function

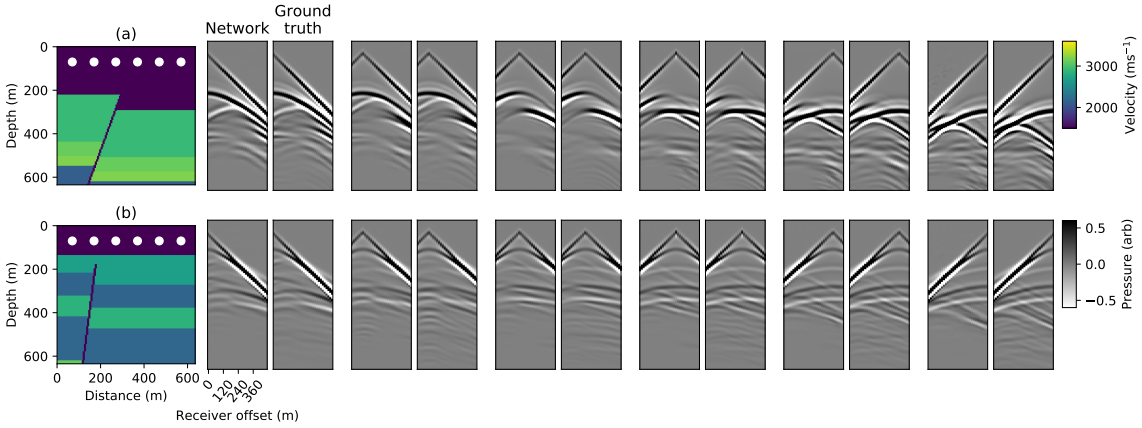


Figure 5.11: Conditional encoder-decoder simulation accuracy when varying the source location. The network simulation is shown for 6 different source locations whilst keeping the velocity model fixed. The source positions are regularly spaced across the surface of the velocity model (white circles). Example simulations for 2 different velocity models in the test set are shown, where each row corresponds to a different velocity model. The pairs of simulation plots in each row from left to right correspond to the network prediction (left in the pair) and the ground truth FD simulation (right in the pair), when varying the source location from left to right in the velocity model. A $t^{2.5}$ gain is applied for display.

during training, when using gain exponents of $g = 0$ and $g = 5$ in the loss function and when removing 2 layers from the encoder and 8 layers from the decoder. We plot the histogram of the average absolute amplitude difference between the ground truth FD simulation and the network simulation over the test set for all of the cases above, and observe that in all cases the simulations are less accurate than our baseline approach. Without the gain in the loss function, the network only learns to simulate the direct arrival and the first few reflections in the receiver responses. With a gain exponent of $g = 5$, the network simulation is unstable and it fails to simulate the first 0.2 seconds of the receiver responses. When using the network with less layers the simulations have edge artefacts, whilst the network with half the number of hidden channels is closest to the baseline accuracy. In testing we find that training a network with the same number of layers but without using a bottleneck design to reduce the velocity model to a 1x1x1024 latent vector does not converge.

We compare the accuracy of the conditional encoder-decoder to the WaveNet network in Figure C.1. We plot the simulation from both networks for an example model in the horizontally layered velocity model test set and the histogram of the average absolute amplitude difference between the ground truth FD simulation

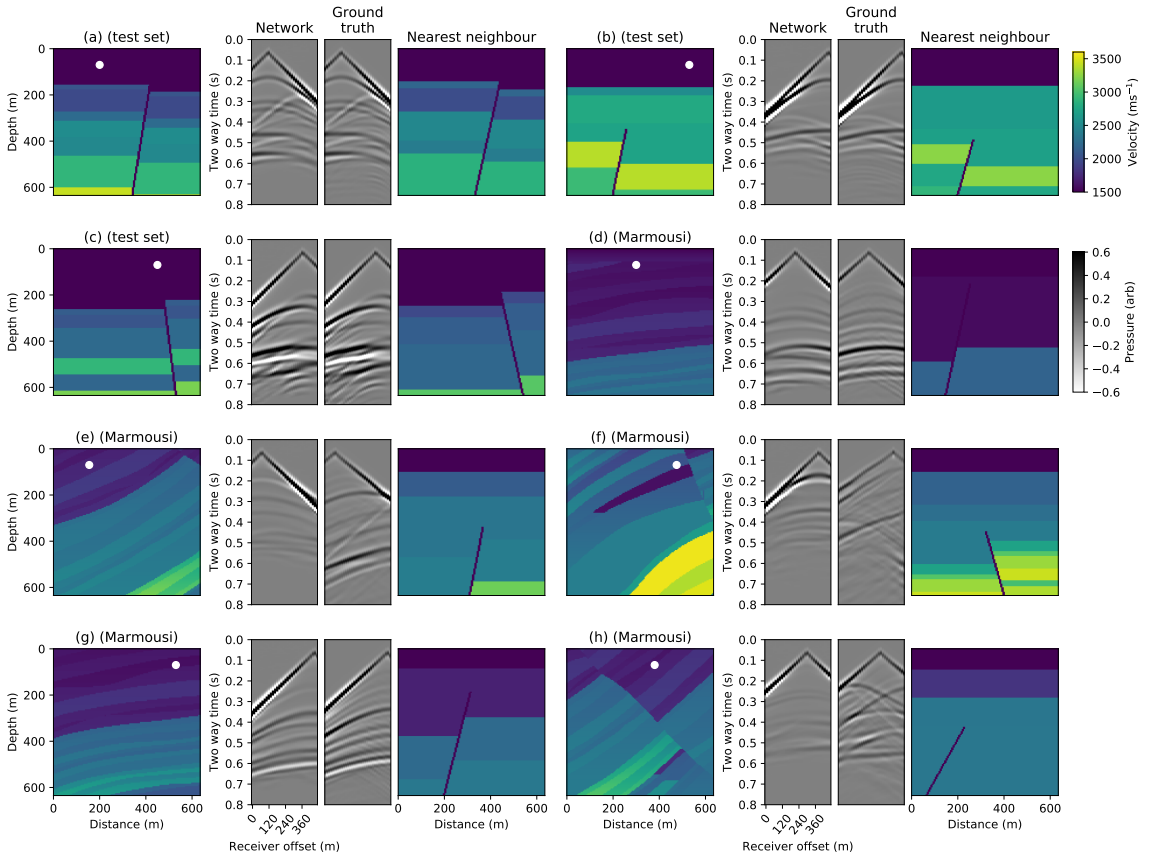


Figure 5.12: Generalisation ability of the conditional encoder-decoder. The conditional encoder-decoder simulations for 5 velocity models taken from different regions of the Marmousi P-wave velocity model are shown (examples (d)-(h)). For each example, left shows the input velocity model and source location, the middle simulations plots show the network prediction (left) and the ground truth FD simulation (right) and right shows the nearest neighbour in the training set to the input velocity model. Simulations from 3 of the test velocity models in Figure 5.10 are also shown with their nearest neighbours (examples (a)-(c)). A $t^{2.5}$ gain is applied for display.

and the WaveNet and conditional encoder-decoder simulations over this set. Both networks are able to accurately simulate the receiver responses and the WaveNet simulation is slightly more accurate than the conditional encoder-decoder, though of course the latter is more general.

We test the generalisation ability of the conditional encoder-decoder outside of its training distribution by inputting randomly selected 640×640 m boxes from the publicly available 2D Marmousi P-wave velocity model [Martin et al., 2006] into the network. These velocity models contain much more complex faulting at multiple scales, higher dips and more layer variability than our training dataset. The resulting

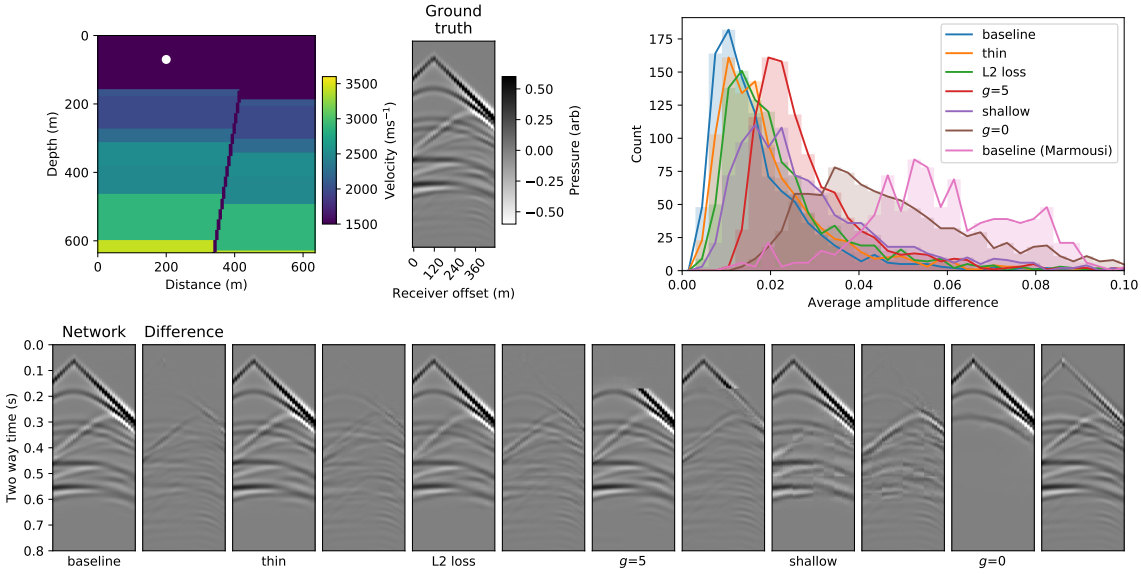


Figure 5.13: Comparison of different conditional encoder-decoder network designs and training hyperparameters on simulation accuracy. Top left shows a randomly selected velocity model and source location from the test set and its corresponding ground truth FD simulation. Bottom compares simulations and their difference to the ground truth when using our proposed conditional encoder-decoder (baseline), when halving the number of hidden channels for all layers (thin), when using a L2 loss function during training (L2 loss), when using gain exponents of $g = 0$ and $g = 5$ in the loss function and when removing 2 layers from the encoder and 8 layers from the decoder (shallow). Top right shows the histogram of the average absolute amplitude difference between the ground truth FD simulation and the simulation from the different cases over the test set. The histogram of the baseline network over the Marmousi test dataset is also shown. A $t^{2.5}$ gain is applied for display.

network simulations are shown in Figure 5.12. We calculate the nearest neighbour to the input velocity model in the set of training velocity models, defined as the training model with the lowest L1 difference summed over all velocity values from the input velocity model, and show this alongside each example.

We find that the network is not able to accurately simulate the full seismic response from velocity models which have large dips and/or complex faulting (examples (e), (f) and (h)) that are absent in the training set. This observation is similar to most studies which analyse the generalisability of deep neural networks outside their training set (e.g. Zhang and Lin [2018] and Earp and Curtis [2020]). However, encouragingly, the network is able to mimic the response from velocity models with small dips ((d) and (g)), even though the nearest training-set neighbour contains a fault whereas the Marmousi layers are continuous.

We compare the average time taken to generate 100 simulations using the conditional encoder-decoder network to FD simulation in Table 5.1. We find that on a single CPU core the network is 22 times faster than FD simulation and when using a GPU and the PyTorch library [Paszke et al., 2019] it is 406 times faster. This is comparable to the speed up obtained with the WaveNet. It is likely that 2D ray tracing will not offer the same speed up as observed in Section 5.3.6, because computing ray paths through these models is likely to be more demanding. The network takes approximately 4 days to train on one NVIDIA Titan V GPU. This is 8 times longer than training the WaveNet network, although we made little effort to optimise its training time. We find that when using only 50,000 training examples the validation loss increases and the network overfits to the training dataset.

5.5 Full wavefield simulation using physics-informed neural networks

5.5.1 Overview

Whilst the conditional encoder-decoder presented above provides a more general simulation approach than the WaveNet network from Section 5.3, both approaches are still unable to simulate seismic waves in realistic, arbitrary, fully heterogeneous media. In this section we take a different approach, and assess whether a PINN [Raissi et al., 2019, Lagaris et al., 1998] is able to simulate seismic waves in both simple and fully heterogeneous Earth-realistic 2D acoustic media.

More specifically, the task is to simulate the seismic response from a point source emitted from within an Earth model, given the source location and the Earth's velocity model as input. There are multiple differences between the simulation task considered here and the previous two approaches. Firstly, rather than learning the partial solution at a set of fixed receiver locations, we train the PINN to simulate the full wavefield solution through the entire Earth model. Secondly, we simulate the seismic response from a point source located within the Earth model, rather than on the surface.

The input to the PINN is a space and time coordinate concatenated with the source location and its output is the estimated wavefield solution at that coordinate.

The wave equation and input velocity model are used directly in the loss function when training the network. Similar to the conditional encoder-decoder presented above, we add the source location as an additional input to the network and attempt to generalise across this initial condition.

There are multiple advantages of using a PINN for this task compared to the previous two approaches. Firstly, by using the wave equation directly in the loss function we require much less training data from FD simulation compared to the approaches above. In particular, only boundary training data from the first few timesteps of FD simulation are required and the network is able to generalise to much later timesteps in an unsupervised fashion. Secondly, the full wavefield solution is learned rather than the response at a set of fixed receiver locations, allowing a more flexible range of applications. Finally, the PINN allows us to simulate seismic waves in fully heterogeneous, Earth-realistic media. However, whilst the PINN presented here can generalise across different source locations without needing to be retrained, a limitation is that it must be retrained for each new velocity model and we discuss this further in Section 5.6.

We will now discuss our simulation workflow and training methodology in more detail below.

5.5.2 Simulation workflow

Our simulation workflow is shown in Figure 5.14. Given a fixed velocity model and a variable source location, a PINN is used to simulate the full wavefield resulting from a point source emitted from within the Earth model. In contrast to the previous two approaches which provide the wavefield solution at a discrete set of points, the PINN defines a continuous functional approximation of the solution. More precisely, the input to the PINN is an arbitrary point in space and time, (x, t), concatenated with the source location, s, and its output, $NN(x, t, s; \theta)$, is an approximation of the wavefield pressure at this point, $NN(x, t, s; \theta) \approx u(x, t, s)$, where θ denote the trainable parameters of the network. As above we focus on 2D acoustic simulation, i.e. the underlying dynamics are governed by Equation 5.1 with $x, s \in \mathbb{R}^2$ and $t, u \in \mathbb{R}^1$.

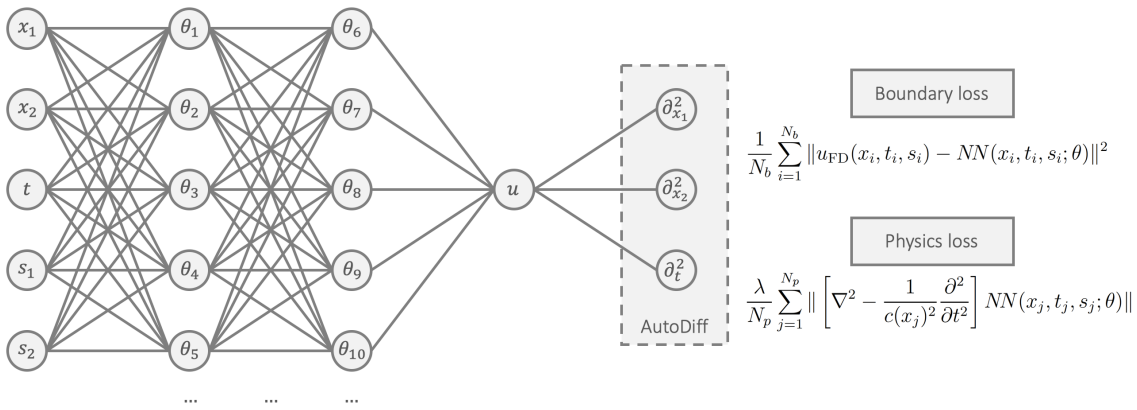


Figure 5.14: Physics-informed neural network used to solve the wave equation. The input to the network is a single point in time and space (x, t) and its output is an approximation of the wavefield solution at this location. We extend the original PINN approach proposed by Raissi et al. [2019] by further conditioning the network on the initial source location s. We use a fully connected network architecture with 10 layers, 1024 hidden channels, softplus activation functions before all hidden layers and a linear activation function for the final layer.

5.5.3 Training process

The PINN is trained by using a combination of data from FD simulation and the wave equation as a direct constraint in its loss function. Specifically, we use the following loss function to train the PINN

$$\mathcal{L}(\theta) = \frac{1}{N_b}\sum_{i=1}^{N_b} \|u_{\text{FD}}(x_i,t_i,s_i) - NN(x_i,t_i,s_i;\theta)\|^2 + \frac{\lambda}{N_p}\sum_{j=1}^{N_p} \| \left[\nabla^2 - \frac{1}{c(x_j)^2}\frac{\partial^2}{\partial t^2}\right] NN(x_j,t_j,s_j;\theta)\| \ , \quad (5.5)$$

where $c(x)$ is the fixed velocity model, $u_{\text{FD}}(x_i, t_i, s_i)$ are a set of example solution points from finite difference modelling and $\lambda \in \mathbb{R}^+$ is a hyperparameter added to control the relative strength of the two loss terms. We denote the first term the "boundary loss" and the second the "physics loss".

There are a number of important points to note about this loss function. Firstly, the boundary loss allows us to define the initial conditions of the simulation. We do this by asserting that the PINN solution matches the solution from FD simulation for a small number of initial timesteps. Given the fixed velocity model and many different example source locations, a FD simulation is run using a Ricker point source for each

example source location over a small number of initial time steps. This results in a set of (discretised) example wavefield solution points $u_{\mathrm{FD}}(x_i, t_i, s_i)$ which the PINN is trained to match in the boundary loss. We use multiple timesteps from FD simulation because the wave equation is a second order differential equation and we therefore need to define both the initial wavefield $u(x, t = 0, s)$ and its temporal gradient $u_t(x, t = 0, s)$ for the solution learned by the PINN to be unique. For simplicity, we only use timesteps from FD simulations after the source term in Equation 5.1 is negligible and redefine $t = 0$ to be time at which the source term is negligible.

Secondly, the physics loss attempts to ensure that the solution learned by the PINN at later times obeys the underlying acoustic wave equation (Equation 5.1). For simplicity, we use a constant density model, and as we are only considering simulation after the source term is negligible above Equation 5.2 can be used in the loss function. The fixed velocity model is included within this loss term as an implicit parameter.

When evaluating the physics loss random batches of points and source locations from the entire domain are selected at each training step, defined as $x_j \in [0, X_{1,\max}] \times [0, X_{2,\max}],\ t_j \in [0, T_{\max}],\ s_j \in [S_{1,\min}, S_{1,\max}] \times [S_{2,\min}, S_{2,\max}]$. When evaluating the boundary loss, random batches of example wavefield solution points are selected at each training step where $x_i \in [0, X_{1,\max}] \times [0, X_{2,\max}],\ t_i \in [0, T_1],\ s_i \in [S_{1,\min}, S_{1,\max}] \times [S_{2,\min}, S_{2,\max}]$. Importantly, the physics loss is evaluated at times much later than the example wavefield solution points (i.e. $T_1 \ll T_{\max}$), such that the PINN learns the solution beyond the boundary training data. After training we use later time steps from FD simulation to benchmark the performance of the PINN at later times. In testing we find that a L1 norm on the physics loss shows better convergence than a L2 norm for this task (discussed further in Section 5.5.5.3).

Finally, curriculum learning is used to help the network converge to a solution. More specifically, we start by training the network to reconstruct the boundary data only by using the boundary loss, and then "switch on" the physics loss halfway through training. This allows the network to learn the initial wavefield before learning to solve the wave equation through time. Furthermore a linearly growing time horizon with training step is used when randomly sampling points in the physics

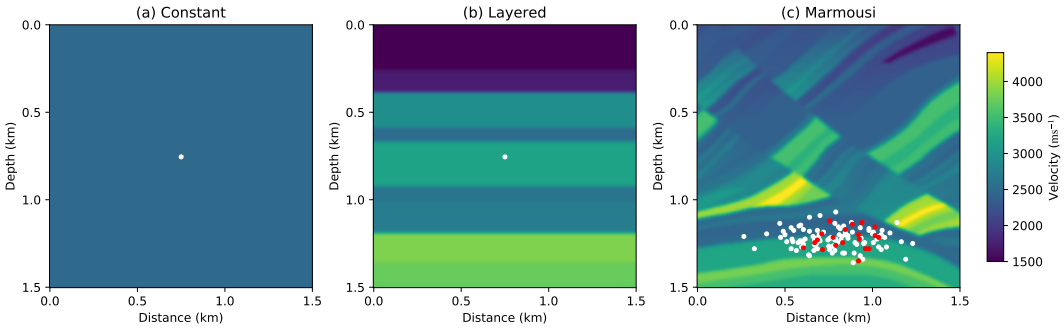


Figure 5.15: Velocity models used to test our approach. We use a PINN to solve the wave equation in a variety of different media, starting from a simple homogeneous velocity model (left), to a layered velocity model (middle) and finally a region of the Earth-realistic Marmousi model [Martin et al., 2006] (right). The velocity model is an implicit parameter in the training scheme and the network must be retrained to solve the wave equation for each case. White points show the locations of the sources used to train the network in each case. For the Marmousi model we train the network to generalise over different source locations and red points show the source locations used to test the network.

loss, analogous to the way a numerical solver iteratively solves the wave equation through time. Both schemes are intended to allow the network to learn incrementally rather than all at once and in testing they are found to improve convergence.

5.5.4 Overview of experiments

We use three different case studies to test our approach. The case studies use increasingly complex velocity models, shown in Figure 5.15, to test the limits of our approach. First, a PINN is trained to simulate the seismic response from a fixed point source located in the center of a homogeneous velocity model. Second, a PINN is trained for the same task, except that a horizontally layered velocity model is used. Finally, a PINN is trained to simulate the seismic response from a point source with a varying location within a region of the Earth-realistic Marmousi P-wave velocity model [Martin et al., 2006]. For the first two case studies we do not attempt to generalise across the source location (i.e. a single source location is used when generating FD training data and sampling points in the loss function), whilst for the last case study we do attempt to generalise across this initial condition. 2D Gaussian smoothing with a standard deviation of 2 grid points is applied to each velocity model before it is used to alleviate discontinuity issues (discussed further in Section 5.5.5).

5.5.4.1 Implementation details

For all tests a fully connected neural network is used with 10 layers, 1024 hidden channels per layer, softplus activation functions after all hidden layers and a linear activation function for the final layer. We choose a softplus activation over ReLU such that the network has continuous second order derivatives, which is required by the underlying wave equation. We write our own code to analytically compute the gradients of the PINN with respect to its inputs required when evaluating the physics loss, although the autodifferentiation features of our deep learning framework (PyTorch [Paszke et al., 2019]) could equivalently be used.

The network is trained using the Adam stochastic gradient descent algorithm with a learning rate of $1\text{x}10^{-5}$ [Kingma and Ba, 2015]. For each update step a random batch of discretised points are sampled from the example wavefield solution points to compute the boundary loss and a random batch of continuous points and source locations over the full input space up to the current time horizon are sampled to compute the physics loss. A batch size of $N_b = N_p = 500$ samples is used for both.

5.5.4.2 Boundary data generation

We use the SEISMIC_CPML FD modelling code [Komatitsch and Martin, 2007] to generate the initial wavefield training data for each case study. All simulations use a 300×300 grid with a spacing of 5 m in each direction (i.e. $X_{1,\text{max}} = X_{2,\text{max}} = 1500$ m), a 0.002 s time interval, a constant density model of 2200 kgm^{-2} and a 20 Hz Ricker source time function.

For the first two velocity models a single simulation with one source location is used. The point source is placed in the centre of the velocity model and the first 10 time steps (i.e. $T_1 = 0.02$ s or 0.14 source cycles) after the source term becomes negligible are taken as training data. For the Marmousi velocity model 100 simulations are used each with a random source location drawn from a 2D Gaussian distribution located towards the bottom of the model (shown in Figure 5.15), and 20 time steps (i.e. $T_1 = 0.04$ s or 0.28 source cycles) after the source term becomes

negligible are taken. For all cases the points used for computing the physics loss are sampled up to a maximum time of $T_{\max} = 0.4$ s.

5.5.5 Results

5.5.5.1 Homogeneous velocity model, single source

Our first case study tests the ability of the PINN in the simplest possible case: a homogeneous velocity model. In this media the initial wavefield propagates outwards with a fixed velocity and its amplitude reduces due to spherical divergence. We train the network and find that both the boundary loss and physics loss converge, and predictions of the wavefield after training are shown in Figure 5.16. The network agrees with the FD solution very accurately, capturing both its kinematics and its amplitude attenuation. We also train the same network with only the boundary loss, and show its wavefield predictions in Figure 5.16. In contrast to this network, the physics-informed network is able to accurately predict the wavefield at times much later than the initial wavefield, showing much better generalisation capability outside of the boundary training data. This is because the PINN's physics loss is sampled at much later timesteps than the labelled FD data the uninformed network is trained on. Despite its poor accuracy at later timesteps, the uniformed network is still able to capture the notion of the wavefront propagating outwards outside of the training data. This case study confirms the validity of the approach and gives us confidence to move onto the more complex case studies below.

5.5.5.2 Layered velocity model, single source

Our second case study uses the horizontally layered velocity model, which induces much more complex wavefield dynamics; reflected and transmitted waves form at each velocity interface and the wavefield is compressed and expanded when travelling through different regions of the velocity model. The boundary and physics loss converge and the predictions of the PINN after training are shown in Figure 5.17. The network is able to accurately capture the full range of dynamics: in particular it is able to simulate transmitted and reflected waves which are not present in the

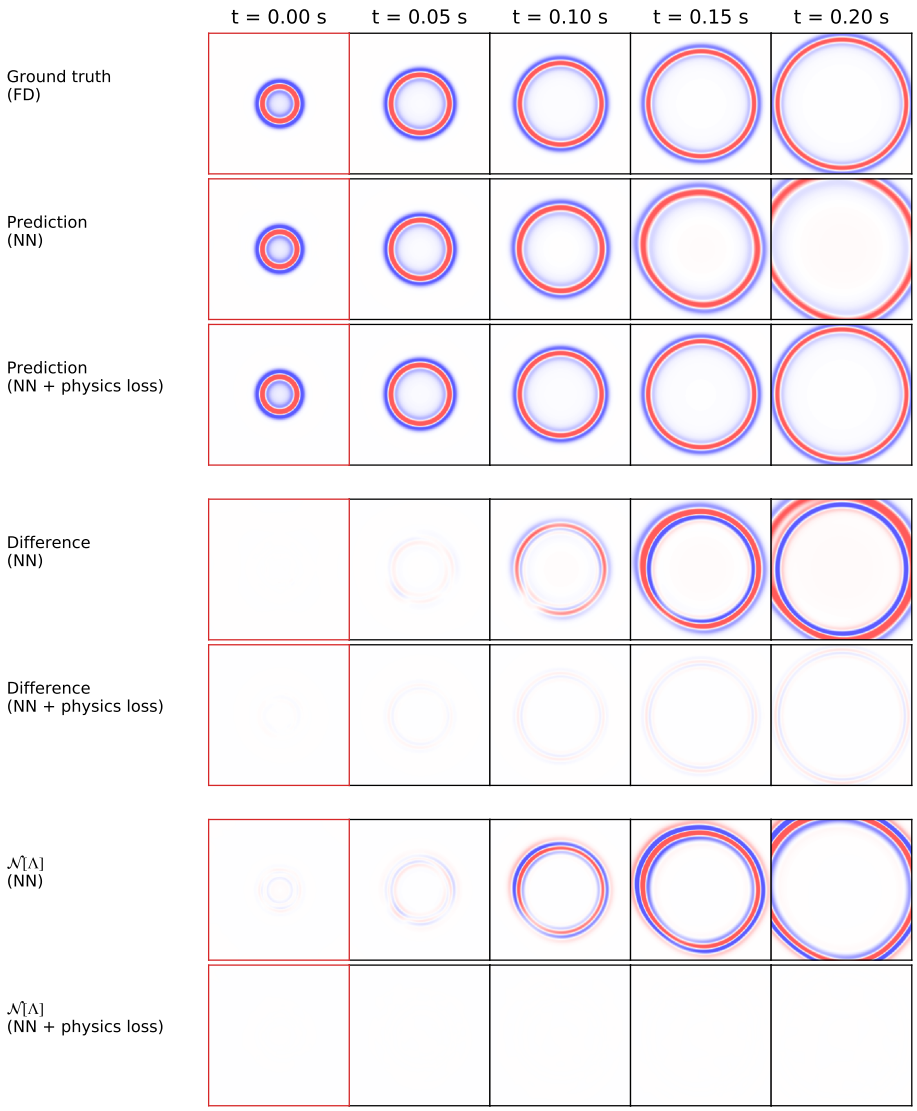


Figure 5.16: Comparison of the PINN wavefield prediction to ground truth FD simulation, using a homogeneous velocity model. The PINN ("NN + physics loss") is also compared to the same network trained using only the boundary loss ("NN"). Top three rows show the FD, PINN and uninformed network wavefield solutions through time. Middle two rows show the difference of the two network predictions to FD simulation. Bottom two rows show the residual of the underlying wave equation (from physics loss in Equation 5.5) for both networks, which is close to zero for an accurate solution to the wave equation. Plots bordered in black indicate wavefield times which are outside of the boundary training data. The boundary training data only covers the time range 0.00-0.02 s and the PINN is able to generalise to much later times (beyond 0.20 s, 10× the range of the boundary data). Colour bar ranges are the same between each type of plot, for fair comparison. The colour bar is shown in Figure 5.19.

initial wavefield, in contrast to the same network just trained using the boundary loss. This suggests that it is able to capture the specific physics phenomena occurring

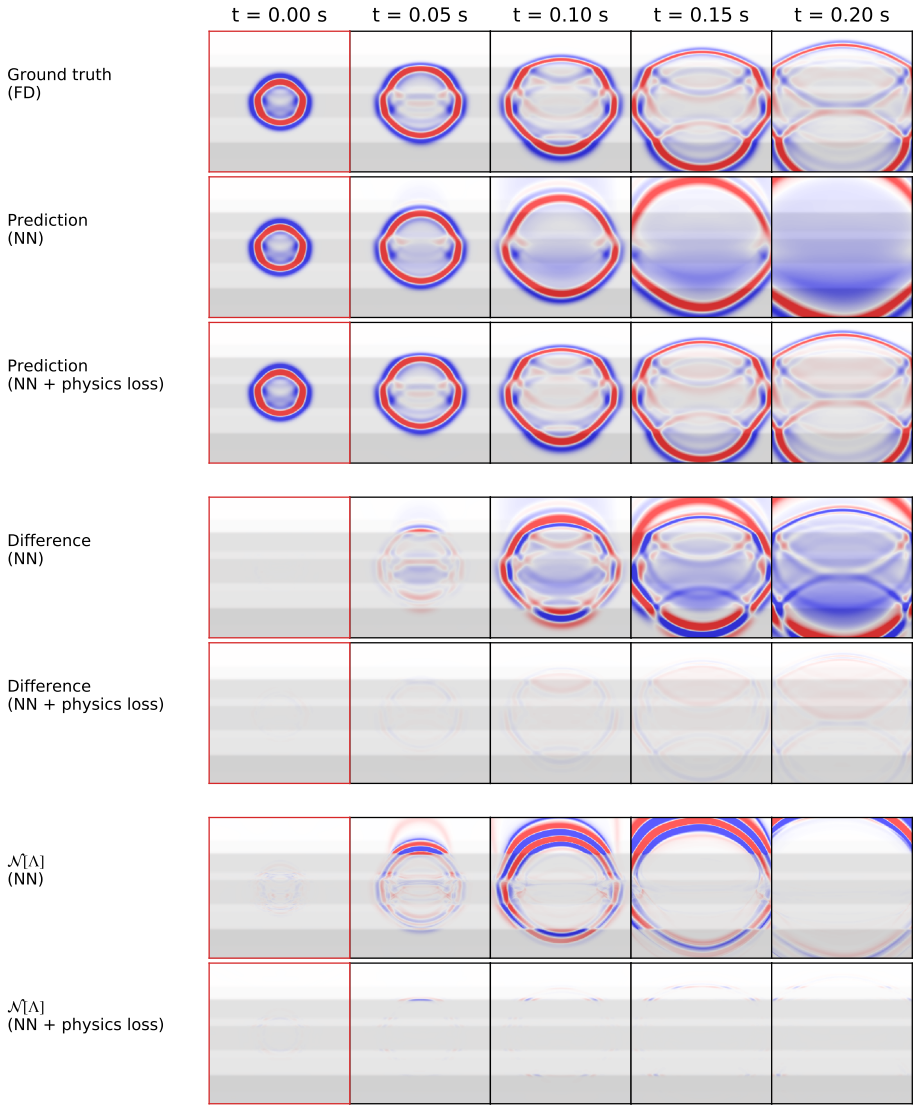


Figure 5.17: Comparison of the PINN wavefield prediction to ground truth FD simulation, using a layered velocity model. The same layout as Figure 5.16 is used; top three rows show the FD, PINN and uninformed network wavefield solutions through time; middle two rows show the difference of the two network predictions to FD simulation; bottom two rows show the residual of the underlying wave equation (from physics loss in Equation 5.5) for both networks, which is close to zero for an accurate solution to the wave equation; plots bordered in black indicate wavefield times which are outside of the boundary training data. In this case the plots are overlain on the velocity model, shown in grey. Similar to Figure 5.16 the boundary training data only covers the time range 0.00-0.02 s and the PINN is able to generalise to much later times (beyond 0.20 s, 10× the range of the boundary data).

at the interfaces of the media. It is also able to accurately capture the compression, expansion and spherical spreading attenuation of the wavefield.

For this case we plot the boundary and physics loss during training in Figure 5.18.

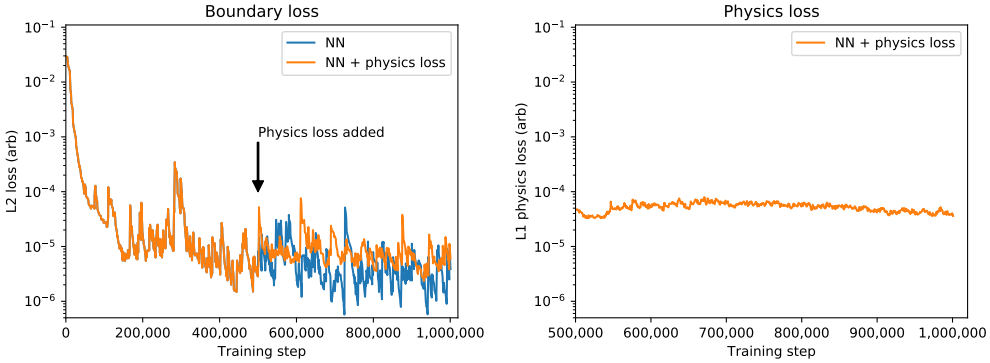


Figure 5.18: Training loss for the layered velocity model against training step. Left shows the boundary loss and right shows the physics loss in Equation 5.5. The physics loss is added halfway through training and includes an expanding time horizon to help the network converge. The PINN ("NN + physics loss") is compared to a network trained only using the boundary loss ("NN").

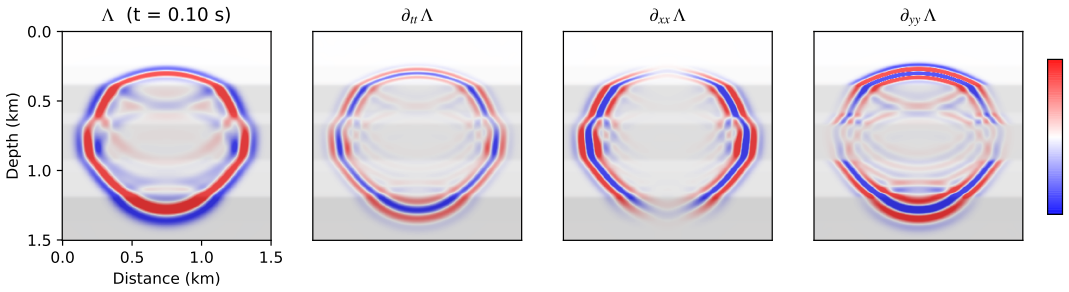


Figure 5.19: Second order derivatives of the PINN trained using the layered velocity model. The derivatives are shown at a single snapshot in time and the plots are overlain on the velocity model. From left to right: network wavefield prediction, $\Lambda = NN(x, t = 0.10)$, its second order time, second order x and second order y derivatives. Sharp changes occur in the second order y derivative at velocity interfaces. Arbitrary scale used for colour bar.

The boundary loss slightly increases when the physics loss is added and then remains stable. The physics loss is relatively constant throughout training, even as later times are added in the expanding time horizon, suggesting on average it is kept at the same low level as the initial wavefield. However, looking in more detail at Figure 5.17 we observe that the loss is higher along the interfaces in the velocity model. This is perhaps due to the discontinuities in the velocity model which translate into discontinuities in the second order gradients of the wavefield along the y-direction, which the network may struggle to represent. We plot the second order gradients of the network in Figure 5.19 and find that the network has learnt sharp, although not fully discontinuous, contrasts at the interfaces.

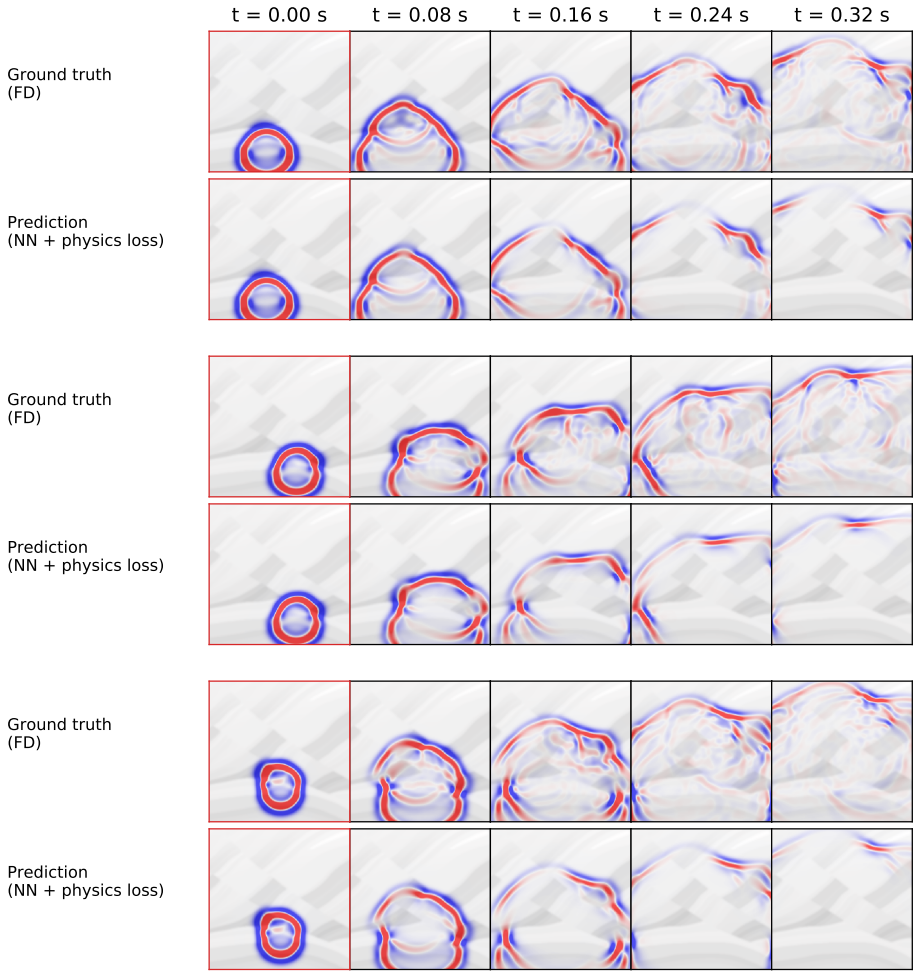


Figure 5.20: Comparison of the PINN wavefield prediction to ground truth FD simulation, using the Marmousi velocity model. Each pair of rows shows the PINN prediction and FD simulation for a randomly selected source location in the test set. The plots are overlain on the velocity model, shown in grey. For this case the boundary training data only covers the time range 0.00-0.04 s and the PINN is able to generalise to much later times (beyond 0.32 s, 8× the range of the boundary data).

5.5.5.3 Marmousi model, multiple sources

Our final case study uses the Marmousi velocity model. This is a complex, Earth-realistic model which includes faults, velocity inversions and a large range of dips (angles of interfaces). For this case study we further increase the complexity of the problem by conditioning the network on the source location, using the source locations shown in Figure 5.15 to train the network. To test the network we generate a separate test set of 20 input source locations (also shown in Figure 5.15) which are not used during training. We find that the boundary and physics loss converge,

albeit at a slower rate than for the homogeneous and layered velocity models, and the predictions of the PINN for three of the source locations in this test set are compared to FD simulation in Figure 5.20.

In this case the network is able to accurately simulate the kinematics of the initial wavefront throughout the media to times much later than the initial wavefield. The network is also able to generalise across the source position, accurately modelling the initial wavefield and subsequent dynamics without needing to be retrained. However, the network struggles to learn the secondary reflected waves of the initial wavefront as it propagates through the model. The reason for this is unclear, and one explanation could be that because of their small relative amplitudes these reflections do not make a large enough contribution to the loss function in Equation 5.5 to be modelled accurately. In testing we found that a L1 norm on the physics loss improved the accuracy of the network compared to a L2 loss, perhaps by increasing the contribution of lower wavefield amplitudes, and it is possible that further amplitude balancing is required. Another explanation is that there are also inaccuracies in the FD simulation. We discuss this issue in more detail in Section 5.6.

For this case study the PINN takes approximately 1 day to train using a single NVIDIA Titan V GPU, although we make little effort to optimise its training time. The majority of this time is spent computing the second order derivatives of the network, which are approximately 10× as slow to calculate as the forward pass of the network. After training, the network is very efficient when computing the wavefield for arbitrary source positions at arbitrary points in time and space (of the order of 0.001 s), whilst 2D FD modelling of the full media is much slower for each source simulation (of the order of 1 s, or 1000× slower).

5.6 Discussion

In this chapter we have presented and evaluated three different approaches for simulating seismic waves in increasingly complex media. Our WaveNet is able to simulate the seismic response recorded at multiple receiver locations on the surface

of an Earth model from a point source emitted at the surface, given any horizontally-layered velocity model as input. Our conditional encoder-decoder is able to carry out the same task, but for any faulted media and any source location on the surface of the media as input. Finally, our PINN is able to simulate the full seismic wavefield from a point source emitted from within simple and fully heterogeneous media, given a fixed velocity model and any source location as input. These approaches show that deep learning and PIML can be used to carry out seismic simulation in increasingly complex media. Furthermore, once trained, these surrogate models are orders of magnitude more efficient than FD simulation.

These results are encouraging and suggest that deep learning and PIML can be valuable for seismic simulation. However, this work also highlights a number of challenges for scaling our approaches to more complex and realistic simulation tasks which are discussed below.

5.6.1 Extension to (visco)elastic simulation

An important ability for realistic geophysical applications is to be able to simulate seismic waves in media with more complex rheologies, such as elastic, viscoelastic and anisotropic media. Such models more realistically describe seismic wave propagation in the Earth and result in more complex phenomena such as compressional and shear wave propagation, attenuation, dispersion and shear wave splitting [Yilmaz, 2001].

From an algorithmic point of view, it is simple to extend our WaveNet and conditional encoder-decoder networks to simulate such media. For example, for elastic wave simulation, shear wave velocity and density models could be added as additional input channels and the number of output channels could be increased so that multi-component particle velocity vectors are output. The same training scheme could be used, with training data generated using elastic FD simulation instead of acoustic simulation and a loss function which compares vector fields instead of scalar fields. Similarly, our PINN could be extended by changing its loss function to use more complex forms of the wave equation and increasing the dimensionality of its

output so that it models vector fields rather than scalar fields. However, further research is required to understand if these extensions are feasible.

One challenge for the WaveNet and conditional encoder-decoder networks is likely to be the increased cost of generating training data. The cost of traditional numerical simulation using more complex forms of the wave equation (e.g. elastic simulation) far exceeds the cost of acoustic simulation by orders of magnitude. However, once trained, we postulate that our approaches will be able to carry out these simulations with similar run times as our current approaches, because the algorithmic changes proposed above are minor. While this is speculative at this point, it is intriguing to investigate, because the current difference in computational cost using traditional approaches has prevented the geophysical community from fully realising the benefits of these simulations.

5.6.2 Extension to 3D simulation

Another essential extension is to move from 2D to 3D simulation. Most real-world applications require 3D simulation to accurately model the Earth. Similar to above, this is at least algorithmically simple to achieve for our approaches. For the WaveNet and conditional encoder-decoder networks, 3D simulation could be achieved by adding an extra dimension to their input, hidden and output layers. For the PINN, one would just need to increase the size of its input layer by one. For all approaches, once trained, we would expect a similar order of magnitude acceleration of simulation time compared to traditional numerical modelling, because the networks would still estimate the seismic response at a given location without needing to iteratively model the full seismic wavefield through time.

However, multiple challenges are likely to arise in this setting. Firstly, for the WaveNet and conditional encoder-decoder networks, increasing their dimensionality would significantly increase the size and number of free parameters in the network and therefore likely make their optimisation harder. Finding a more efficient representation, such as meshes or oct-trees [Ahmed et al., 2018] to reduce the dimensionality of the problem may be critical in this aspect.

Secondly, and similar to above, a major challenge is likely to be the increased computational cost of generating training data, which for instance is significantly higher in 3D. For the WaveNet and conditional encoder-decoder networks, whilst we only use the subset of the wavefield at each receiver location to train our networks, finding a way to use the entire wavefield from FD simulation to train these network may help reduce the number of training simulations required. The cost of generating training data may be less important for our PINN, as only the first few timesteps from FD simulation are required. However, the PINN does not escape this challenge entirely; it is likely that many more training points when evaluating its physics loss are needed to ensure that the PINN fully samples its 3D input domain, which is likely to increase training times significantly.

Although large, we note that generating training data and training our networks is an amortised cost because the networks only needs to be trained once. Thus high training costs could become permissible in geophysical applications such as seismic inversion where thousands of similar simulations are required.

A related extension is to assess whether our WaveNet and conditional encoder-decoder networks are able to simulate the seismic response at vertically-spaced receiver locations at different depths rather than horizontally-spaced receiver locations at the surface. Such a model may be useful for simulating e.g. the seismic activity around a hydrocarbon well. Only the training data of these models needs to be changed and we expect them to perform similarly, as vertical seismic responses typically have a similar character (in terms of number of events, frequency and amplitude range) to surface responses.

5.6.3 Generalisation to more varied and complex Earth models

Perhaps the largest challenge is overcoming the lack of generalisation of our approaches. More specifically, whilst our WaveNet and conditional encoder-decoder networks are effective at simulating seismic waves in media similar to their training distribution, when tested using media outside of this distribution their performance

degrades significantly. Furthermore, to simulate more complex velocity models, the conditional encoder-decoder requires more free parameters, more time to train and more training examples than the WaveNet network. Similarly, our PINN requires longer training times when simulating seismic waves in the Earth-realistic Marmousi media compared to when simulating seismic waves in homogeneous media.

The poor generalisation of deep neural networks outside of their training distribution is a well known and common challenge in general [Goodfellow et al., 2016]. A naive approach would be to increase the size and variety of the training data to improve the generality of our networks, however this would quickly become computationally intractable when trying to sample all possible Earth models and initial conditions. This trade-off between computational intractability and network generalisation appears analogous to the trade-off between computational effort and the level of approximation using traditional simulation noted in Section 5.2. We note that for many practical applications it may be acceptable to use a training distribution with a limited range; for example, in many seismic inversion applications such tomography, FWI, and seismic hazard assessment, thousands of forward simulations of similar Earth models are carried out. We find that the nearest neighbour test is a useful way to understand if an input velocity model is close to the training distribution and therefore if the network's output simulation is likely to be accurate. Probabilistic approaches, such as Bayesian deep learning [Gal, 2016], could be investigated for their ability to provide more quantitative uncertainty estimates on the network's output simulation.

To overcome this challenge it is likely that stronger physical priors are required to regularise our workflows better. Using causality in the WaveNet generates more accurate simulations than when using a standard convolutional network; this suggests that adding this constraint helps the network simulate the seismic response, although it is an open question how best to represent causality when simulating more arbitrary Earth models. We also find that a bottleneck design helps the conditional encoder-decoder to converge; our hypothesis is that this encourages a depth-to-time conversion by slowly reducing the spatial dimensions of the velocity model before expanding

them into time. More advanced network designs, for example using attention-like mechanisms [Vaswani et al., 2017] to help the network focus on relevant parts of the velocity model, rather than using convolutional layers with full fields of view, or using Long Short-Term Memory (LSTM) cells to help the network model multiple reverberations could be tested. For the PINN, using the neural network as part of a solution ansatz where the initial and boundary conditions of the solution are already defined may make it easier to train [Lagaris et al., 1998] and recent work on learning operator mappings with PINNs may help them learn families of solutions [Wang et al., 2021d].

Finally, another important challenge is to investigate whether the simulation costs of our approaches scale more favourably with increasing frequency, ω, compared to traditional numerical methods which typically scale with ω^4; in this study we only consider simulation at a fixed frequency range and it would be intriguing to study this further.

5.6.4 PINN-specific challenges

A few challenges specifically relating to our PINN are identified. Firstly, the PINN struggles to accurately model the reflected and transmitted waves at the interfaces in the media, where there are discrete changes in the second order derivatives of the wavefield. We note that modelling these discontinuities is also a challenge for numerical methods, where oversampling is typically required to ensure the solution remains accurate [Igel, 2017]. We somewhat avoid the problem by smoothing the velocity model before use, although errors are still observed along the interfaces. The PINN prediction when using the layered velocity model without smoothing is shown in Figure C.3; the accuracy of this solution is significantly worse than in Figure 5.17. Another likely related issue is that the Marmousi case study struggles to model reflected waves with low relative amplitudes. Future work could investigate these observations, for example by forcing the network to be discontinuous at interfaces (e.g. by using separate networks for each velocity section), by changing the activation function [Sitzmann et al., 2020], by using specific physics constraints

that arise at interfaces in the loss function (such as wavefield continuity [Aki and Richards, 1980]), or by using a more intelligent sampling scheme, such as adaptive co-location [Anitescu et al., 2019], instead of random sampling to sample interfaces more often in the loss function.

Another extension of this work is to condition the PINN on more initial conditions, such as the velocity model, as well as the source location. This would allow it to become more general and potentially be useful for a wider range of applications. However, combining a discretised velocity model (similar to those input to our WaveNet and conditional encoder-decoder networks) with the functional representation of the wavefield solution provided by the PINN is likely to be challenging. One approach may be to use an "encoder" network to reduce the velocity model into a set of salient features before feeding it to the fully connected network to help the network converge, although more research is required to assess this approach, and in particular how it scales to more complex velocity models over larger input domains.

5.7 Summary

We have shown that PIML can simulate complex seismic phenomena, although given the potentially large training costs and the challenges of generality, it may be that the PIML techniques tested are most advantageous to practical simulation tasks where many similar simulations are required, such as seismic inversion or statistical seismic hazard analysis, and least useful for problems where a very small number of simulations per model family are used. In seismology, however, many current and future challenges fall into the former category, which renders these initial results promising. Further research is required to understand how best to scale these approaches to more complex forms of the wave equation and 3D simulation, as well as how to design approaches which generalise to unseen Earth models outside of their training distribution. Finally we note that we only tested three particular approaches and many others could be designed which could prove more effective.

6

Scalable physics-informed neural networks

6.1 Introduction

Up until now we have used domain-specific case studies to investigate the scalability of various PIML techniques, where each technique was selected for and highly tuned to the problem at hand. In this chapter we instead study the scalability of a general-purpose PIML technique. In particular, we consider whether PIML algorithms can carry out large-scale simulations, and investigate the scalability of physics-informed neural networks (PINNs) [Raissi et al., 2019, Lagaris et al., 1998] when solving differential equations with large domains.

PINNs are a general-purpose approach for solving problems relating to differential equations and have recently become popular across many different domains [Cuomo et al., 2022]. As discussed in Chapter 2, PINNs have several advantages compared to traditional numerical methods, for example they provide mesh-free solutions to differential equations and they are able to carry out forward and inverse modelling within the same optimisation problem.

Whilst popular, much of the current research on PINNs only considers problems confined to small domains with simple, low frequency solutions. This is useful for proving the concept of the work, as such problems are often well-studied and are easy to implement, but less research has focused on how well PINNs scale

to problems with larger domains and more complex, higher frequency and multi-scale solutions. Being able to solve such problems is crucial for many real-world applications. Importantly, when solving differential equations, traditional numerical methods often become computationally intractable as higher frequencies (or similarly, longer distance/ time scales) are introduced into the solution. Thus, it is important to assess how well PINNs scale in this setting.

In this chapter we investigate how well PINNs scale to problems with larger domains. First, we consider a motivating problem where we use a PINN to solve a simple differential equation and assess how its performance changes as higher frequencies are introduced into the solution. Next, we propose an extension to PINNs designed to help them scale to large-scale problems, called finite basis physics-informed neural networks (FBPINNs). FBPINNs are a general framework for solving problems relating to differential equations and extend PINNs by combining them with domain decomposition, individual subdomain normalisation and flexible training schedules. The performance of FBPINNs and PINNs is compared across a range of small and larger, multi-scale simulation tasks.

Across all the problems studied the PINNs tested perform well for problems with small domains and low frequency solutions, but they struggle when solving problems with larger domains and higher frequency solutions. More specifically, as the problem size increases they take significantly longer to converge, require many more free parameters in their neural networks and have worse accuracy. Multiple related factors contribute to this issue, including the increasing complexity of the underlying PINN optimisation problem as the problem size grows and the spectral bias of neural networks.

In contrast to the PINNs tested, the proposed FBPINNs are able to alleviate some of these scaling challenges. They are able to accurately and efficiently solve both the smaller and larger scale problems studied, including those with multi-scale solutions, and thus potentially allow a way to scale PINNs to large, real-world problems.

Whilst the FBPINNs outperform the PINNs in this setting, we find some remaining challenges when scaling FBPINNs to large domains. In particular, the

accuracy of the FBPINN solution can depend on the choice of domain decomposition, and it is likely that the performance of FBPINNs decreases for high-dimensional problems. Furthermore, despite their improved efficiency compared to PINNs the computational costs of FBPINNs remain high compared to traditional numerical methods. These challenges are discussed further. Overall, our work shows that it is highly non-trivial to scale PINNs to large domains.

Finally, in addition to this work we have released an open-source library which allows users to solve general differential equations over large domains with any number of dimensions using FBPINNs, available here: `https://github.com/benmoseley/FBPINNs`.

6.2 Background

6.2.1 On scaling PINNs to real-world problems

As described in Chapter 2, PINNs have recently emerged as a popular and powerful technique for solving problems related to differential equations [Lagaris et al., 1998, Raissi et al., 2019, Karniadakis et al., 2021]. Since their conception, PINNs have been utilised in a wide range of applications and many extensions have been proposed.

Whilst popular and effective, PINNs suffer from some significant limitations which limit their application on real-world problems. One is that, in comparison to classical approaches, the theoretical convergence properties of PINNs are poorly understood. This can make it challenging to know how accurate a PINN solution is. The PINN loss function can be highly non-convex and result in a stiff optimisation problem, yet it is unclear which classes of problems this affects [Wang et al., 2021c, 2022]. Existing work has made initial steps towards understanding the theoretical properties of PINNs [Wang et al., 2022, Shin et al., 2020, Mishra and Molinaro, 2022] but this sub-field is still in its early stages.

Another limitation is the poor computational efficiency of PINNs. A standard PINN must be retrained for each solution, which is expensive and typically means traditional numerical methods strongly outperform PINNs, at least for forward modelling tasks. Works which condition PINNs on their initial conditions so that they

learn a family of solutions (i.e. surrogate models, rather than single solutions), such as physics-informed DeepONets [Lu et al., 2021, Wang et al., 2021b] and discretised versions of PINNs which use finite difference filters to construct differential equation residuals in their loss function [Zhu et al., 2019, Geneva and Zabaras, 2020, Gao et al., 2021] are promising and could potentially help alleviate this issue.

Finally, and of central importance in this work, is the challenge of scaling PINNs to larger problem domains. As the complexity of the solution increases, the size of the neural network (or number of free parameters) inevitably needs to increase such that the network is expressive enough to represent the solution. This in turn results in a harder PINN optimisation problem, both in terms of the number of free parameters and the increased number of training points required to sufficiently sample the solution over the larger domain.

Related to this challenge is the spectral bias of neural networks. This is the well-studied observation that neural networks tend to learn higher frequencies much more slowly than lower frequencies [Xu et al., 2019b, Rahaman et al., 2019, Basri et al., 2019, Cao et al., 2019], with various convergence rates being proved. Because a problem's input variables are typically normalised over the domain before being input to a PINN, low frequency features in the solution effectively become high frequency features as the domain size increases, which can severely hinder the convergence of PINNs [Wang et al., 2021e].

6.2.2 Related work

The limitations described above have meant that the vast majority of PINN applications to date have only solved problems in small domains with a limited frequency range. Recent works have started to investigate these issues. Wang et al. [2021e] showed that spectral bias affects PINNs, and proposed the use of Fourier input features to help alleviate this issue, which transform the input variables over multiple scales using trigonometric functions as a pre-processing step. Liu et al. [2020] proposed multi-scale deep neural networks, which used radial scaling of the input variables in the frequency domain to achieve uniform convergence across multiple

scales when solving problems with a PINN loss. However, for both these methods, the scale of their input features must be chosen to match the frequency range of interest.

For reducing the complexity of the PINN optimisation problem, an increasingly popular approach is to use domain decomposition [Heinlein et al., 2021], taking inspiration from existing classical methods such as finite element modelling. Instead of solving one large PINN optimisation problem, the idea is to use a "divide and conquer" approach and train many smaller problems in parallel. This can reduce training times and could potentially reduce the difficulty of the global optimisation problem too.

For example, Jagtap and Karniadakis [2020] proposed extended physics-informed neural networks (XPINNs), which divide the problem domain into many subdomains, and use separate neural networks in each subdomain to learn the solution. A key consideration in any domain decomposition approach is to ensure that individual subdomain solutions are communicated and match across the subdomain interfaces, and Jagtap and Karniadakis [2020] rely upon additional interfaces terms in their PINN loss function to do so. In follow up work Shukla et al. [2021] showed their approach can be parallelised to multiple GPUs, decreasing training times. However, a key downside of their approach is that it contains discontinuities in the PINN solution across subdomain interfaces, as the interface conditions are only weakly constrained in the loss function.

Similar domain decomposition strategies were proposed by Dwivedi et al. [2021] and Dong and Li [2021], but who instead used extreme learning machines (ELMs) [Huang et al., 2006] as the neural networks in each subdomain, which can be rapidly trained. However, whilst Dong and Li [2021] claimed computational efficiency comparable to finite element modelling, ELMs restrict the capacity of neural networks as only the weights in the last layer can be updated. In other related approaches, Li et al. [2020c] replaced subdomain solvers of the classical Schwarz domain decomposition approach with PINNs, resulting in a similar strategy to Jagtap and Karniadakis [2020]. Stiller et al. [2020] proposed GatedPINNs, which use an auxiliary neural network to learn the domain decomposition itself, and Kharazmi

et al. [2021] use domain decomposition to train variational PINNs, although only when defining the space of test functions.

In this work, we propose a new domain decomposition approach for solving large, multi-scale problems relating to differential equations called finite basis physics-informed neural networks (FBPINNs). In contrast to the existing domain decomposition approaches, we ensure that interface continuity is strictly enforced across the subdomain boundaries by mathematical construction of our PINN solution ansatz, which removes the need for additional interface terms in our PINN loss function. This is achieved by using overlapping subdomains, multiplying each subdomain network with a smooth window function and defining the solution ansatz to be a summation over all subdomain networks. Furthermore, we explicitly consider the effects of spectral bias by using separate input variable normalisation within each subdomain, which restricts the effective solution frequency each subdomain network sees. We further propose flexible training schedules which can aid the convergence of FBPINNs and propose a parallel training algorithm which allows FBPINNs to be scaled computationally.

6.2.3 PINN definition and example

The mathematical definition of a PINN was described in Section 2.3.2.4 and we use the same definition and nomenclature in this chapter. To aid understanding, in this section we give a concrete example of a PINN and describe more of the considerations required when training PINNs.

As a concrete example, one could consider using a PINN, $NN(x, t; \theta)$, to solve the nonhomogeneous time-dependent wave equation with Dirichlet and Neumann boundary conditions at $t = 0$, given by

$$\begin{aligned} \left[\nabla^2 - \frac{1}{c^2}\frac{\partial^2}{\partial t^2}\right] u(x,t) &= f(x,t) \ , \\ u(x,0) &= g_1(x) \ , \\ \frac{\partial u}{\partial t}(x,0) &= g_2(x) \ , \end{aligned} \tag{6.1}$$

where c is the wave speed (note, we have separated the time dimension from the spatial dimensions of the input vector for readability). In this case the PINN loss function (given by Equation 2.3) becomes

$$\mathcal{L}(\theta) = \frac{1}{N_p}\sum_{i}^{N_p} || \left[\nabla^2 - \frac{1}{c^2}\frac{\partial^2}{\partial t^2}\right] NN(x_i, t_i; \theta) - f(x_i, t_i) ||^2 + \frac{1}{N_{b1}}\sum_{j}^{N_{b1}} || NN(x_{1j}, 0; \theta) - g_1(x_{1j}) ||^2 + \frac{1}{N_{b2}}\sum_{j}^{N_{b2}} || \frac{\partial}{\partial t} NN(x_{2j}, 0; \theta) - g_2(x_{2j}) ||^2 . \tag{6.2}$$

There are a number of important points to note regarding the training of PINNs. The first is that a sufficient number of training points $\{x_i\}$ and $\{x_{kj}\}$ should be selected such that the network is able to learn a consistent solution across the entire domain. Secondly, when evaluating the loss function the gradients of the neural network with respect to its inputs are required; these are typically analytically available and easily obtainable in modern deep learning packages through the use of autodifferentiation [Paszke et al., 2019]. Thirdly, whilst the known values of the solution (and/or its derivatives) are required to evaluate the boundary loss, evaluating the physics loss only requires samples of the input vector, i.e. from a ML perspective this term can be viewed as an unsupervised regulariser. Thus, PINNs require very little training data in this sense.

Finally, it is important to note that the PINN represents a continuous functional approximation of the solution, and not just a discrete representation on a mesh.

6.2.4 Weakly vs strongly constrained PINNs

In this section we describe an alternative strategy for asserting boundary conditions when training PINNs. We will use this strategy for all the problems studied in this chapter, although we note that the proposed FBPINN framework is not limited to this approach and can use the standard approach too.

A downside of the standard PINN approach described by Equation 2.3 is that the boundary conditions are only weakly enforced in the loss function, meaning that the learned solution may be inconsistent. Furthermore recent work has shown theoretically and empirically that the PINN optimisation problem can be stiff to solve,

to the point where it may not converge at all, due to the two terms in the loss function competing with each other [Wang et al., 2021c,e, Sun et al., 2020]. An alternative approach, as originally proposed by Lagaris et al. [1998], is to strictly enforce the boundary conditions by using the neural network as part of a solution ansatz. For example, for the wave equation problem described by Equation 6.1, instead of defining a neural network to directly approximate the solution $NN(x,t;\theta) \approx u(x,t)$, one could instead use the ansatz

$$\hat{u}(x,t;\theta) = g_1(x) + t\, g_2(x) + t^2\, NN(x,t;\theta) \ , \tag{6.3}$$

to define the approximate solution in the PINN optimisation problem. It is straightforward to verify that this ansatz automatically satisfies the boundary conditions in Equation 6.1. Because the boundary conditions are automatically satisfied, only the physics loss, $\mathcal{L}_p(\theta)$, needs to be included in Equation 2.3, which turns the optimisation problem from a constrained one into a simpler unconstrained one. This formulation has since been extended to irregular domains [Lagaris et al., 2000, Berg and Nyström, 2018], and general schemes for constructing suitable ansatze have been proposed [Leake and Mortari, 2020].

6.3 A motivating example

In this section we investigate a motivating problem where we use a PINN to solve a simple differential equation and assess how its performance changes as higher frequencies are introduced into the solution. This problem will serve to motivate the FBPINN framework introduced in Section 6.4. Specifically, we consider the following problem

$$\begin{aligned} \frac{du}{dx} &= \cos(\omega x) \ , \\ u(0) &= 0 \ , \end{aligned} \tag{6.4}$$

where $x, u, \omega \in \mathbb{R}^1$, which has the exact solution

$$u(x) = \frac{1}{\omega}\sin(\omega x) \ . \tag{6.5}$$

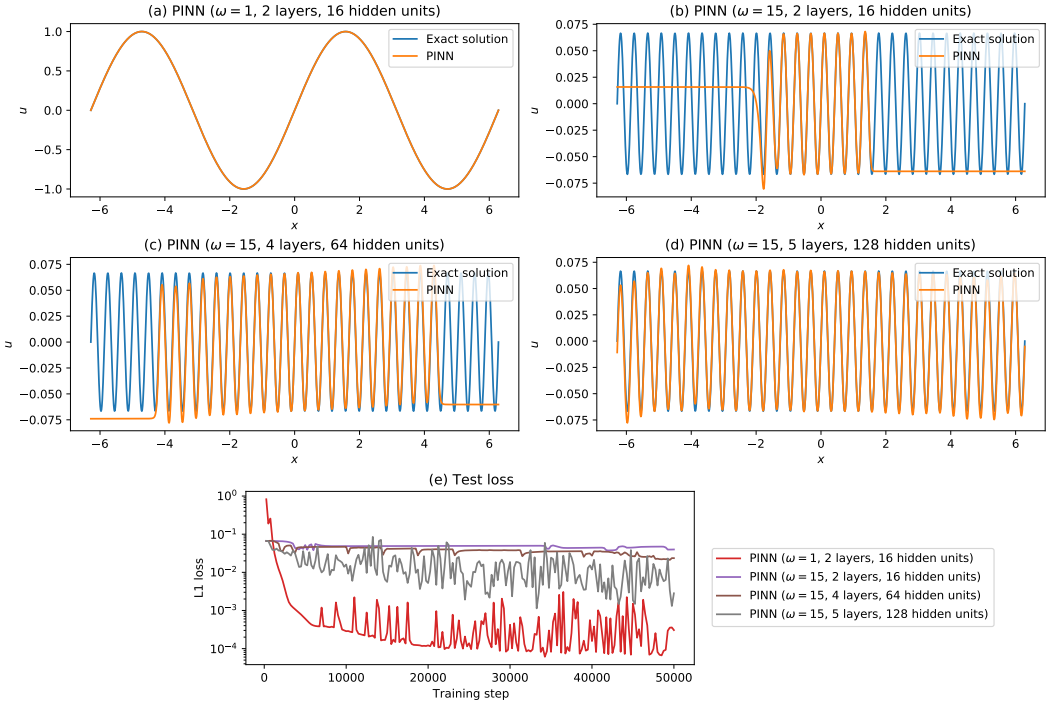


Figure 6.1: A motivating problem: using PINNs to solve $\frac{du}{dx} = \cos(\omega x)$. When $\omega = 1$ (i.e. ω is low), a PINN with 2 hidden layers and 16 hidden units is able to rapidly converge to the solution; (a) shows the PINN solution compared to the exact solution and (e) shows the L1 error between the PINN solution and the exact solution against training step. When $\omega = 15$ (i.e. ω is high), the same PINN struggles to converge, as shown in (b). Whilst increasing the size (number of free parameters) of the PINN improves its accuracy, as shown in (c) and (d), it converges much more slowly and with much lower accuracy than the low frequency case, as shown in (e).

We will use the PINN solution ansatz

$$\hat{u}(x;\theta) = \tanh(\omega x) NN(x;\theta) \ , \tag{6.6}$$

to solve this problem. We choose to use the tanh function in the ansatz because away from the boundary condition its value tends to ± 1, such that the neural network does not need to learn to dramatically compensate for this function away from the boundary. We also approximately match the width of the tanh function to the wavelength of the exact solution, again such that the compensation the network needs to learn in the vicinity of the boundary is of similar frequency to the solution. In our subsequent experiments we find both of these strategies help PINNs converge over large domains and different solution frequencies.

The PINN is trained using the unconstrained loss function

$$\mathcal{L}(\theta) = \mathcal{L}_p(\theta) = \frac{1}{N_p} \sum_{i}^{N_p} || \frac{d}{dx} \hat{u}(x_i; \theta) - \cos(\omega x_i) ||^2 \ , \tag{6.7}$$

as derived using Equation 2.5.

6.3.1 Low frequency case ($\omega = 1$)

First, the above PINN is trained for the $\omega = 1$ case. We use a fully connected network with 2 hidden layers, 16 hidden units per layer and tanh activation functions, using 200 training points ($\{x_i\}$) regularly spaced over the domain $x \in [-2\pi, 2\pi]$. The input variable x is normalised to $[-1, 1]$ over the domain before being input to the network, and the output of the network is unnormalised by multiplying it by $\frac{1}{\omega}$ before being used in the ansatz defined by Equation 6.6. The PINN is trained using gradient descent with the Adam optimiser [Kingma and Ba, 2015] and a learning rate of 0.001. All code is implemented using the `PyTorch` library, using its autodifferentiation features [Paszke et al., 2019].

Figure 6.1 (a) and (e) shows the PINN solution after training and its L1 error compared to the exact solution (evaluated using 1000 regularly spaced points in the domain) as a function of training step. We find that the PINN is able to quickly and accurately converge to the exact solution.

6.3.2 High frequency case ($\omega = 15$)

Next, the same experiment is run, but with two changes; we increase the frequency of the solution to $\omega = 15$ and the number of regularly spaced training points to $200 \times 15 = 3000$. Figure 6.1 (b) and (e) shows the resulting PINN solution and L1 convergence curve (using 5000 test points). We find for this case the PINN is unable to accurately learn the solution, only being able to capture the first few cycles of the solution away from the boundary condition. We further retrain the PINN using larger network sizes of 4 layers and 64 hidden units, and 5 layers and 128 hidden units, shown in Figure 6.1 (c) and (d), and find that only the PINN with 5 layers and 128 hidden units is able to fully model all of the cycles, although

its final relative L1 error is much worse than the $\omega = 1$ case and its convergence curve is much slower and more unstable. We note that the $\omega = 1$ PINN with 2 layers and 16 hidden units uses 321 free parameters whilst the $\omega = 15$ PINN with 5 layers and 128 hidden units uses 66,433 free parameters.

6.3.3 Remarks

Whilst the PINN performs well for low frequencies, it struggles to scale to higher frequencies; the higher frequency PINN requires many more free parameters, converges much more slowly and has worse accuracy. Multiple significant and related factors contribute to this issue. One is that as the complexity of the solution increases, the network requires more free parameters to accurately represent it, making the optimisation problem harder. Another is that as the frequency increases, more training sample points are required to sufficiently sample the domain, again making the optimisation problem harder. Third is the spectral bias of neural networks, which is the observation that neural networks tend to learn higher frequencies much more slowly than low frequencies, a property which has been rigorously studied. Compounding these factors is the fact that, as the size of the network, number of training points, and convergence time grows, the computational resources required increases significantly.

It is important to note that for this problem, scaling to higher frequencies is equivalent to scaling to larger domain sizes. Because we have normalised the input variable to $[-1, 1]$ within the domain before inputting it to the network, keeping $\omega = 1$ but expanding the domain size by 15 times and re-normalising presents the same optimisation problem to the neural network as changing ω to 15. Indeed, increasing the domain size and scaling to higher frequencies are related problems, and the above case study highlights a general observation of PINNs: as the domain size increases, the PINN optimisation typically becomes much harder and takes much longer to converge.

Another important note is that classical methods also typically scale poorly to large domain sizes/ higher frequencies. For example, when solving the problem above the number of mesh points required by standard finite difference modelling would

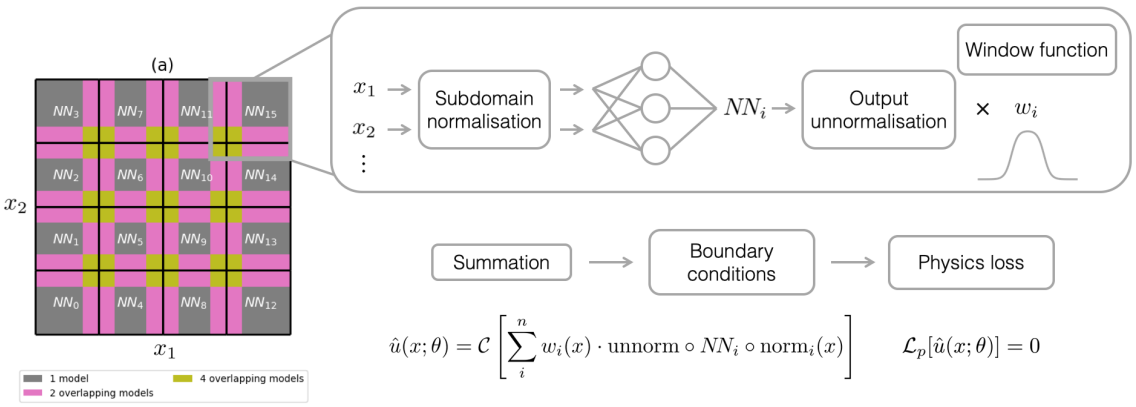


Figure 6.2: FBPINN workflow. FBPINNs use domain decomposition and separate subdomain normalisation to address the issues related to scaling PINNs to large domains. First, the problem domain is divided into overlapping subdomains; an example 2D hyperrectangular subdivision is shown in (a). Next, separate neural networks are placed within each subdomain. Each network is locally confined to its subdomain by multiplying it with a differentiable window function, such that within the center of the subdomain, the network learns the full solution to the differential equation, whilst in the overlapping regions, the solution is defined as the sum over all overlapping networks. For each network, the input variables $x = (x_1, x_2, ...)$ are normalised between [-1,1] over the subdomain, and the output of the network is unnormalised using a common unnormalisation. Finally, an optional constraining operator $\mathcal{C}$ can be applied which appropriately constrains the ansatz such that it automatically satisfies the boundary conditions. FBPINNs are trained using a very similar loss function to standard PINNs which does not require the use of additional interface terms.

scale $\propto \omega^d$. However, FD modelling would not suffer from the additional PINN-related problems described above of many more free parameters in the optimisation problem and slower convergence due to spectral bias.

In the next section we will present FBPINNs, which, as we shall see in Section 6.5, are able to solve the case study above much more accurately and efficiently than the PINNs studied.

6.4 Finite basis physics-informed neural networks (FBPINNs)

6.4.1 Workflow overview

FBPINNs are a general domain decomposition approach for solving large, multi-scale problems relating to differential equations. The main goal of FBPINNs is to address the scaling issues of PINNs described above, which is achieved by using a

combination of domain decomposition, subdomain normalisation and flexible training schedules. In this subsection we give an overview of FBPINNs before giving a detailed mathematical description in Section 6.4.2.

The FBPINN workflow is shown in Figure 6.2. In FBPINNs, the problem domain is divided into overlapping subdomains. A neural network is placed within each subdomain such that within the center of the subdomain, the network learns the full solution, whilst in the overlapping regions, the solution is defined as the sum over all overlapping networks. Before being summed each network is multiplied by a smooth, differentiable window function which locally confines it to its subdomain. In addition to this the input variables of each network are separately normalised over each subdomain, and we can define flexible training schedules which allow us to restrict which subdomain networks are updated at each gradient descent training step (described in more detail in Section 6.4.3).

By dividing the domain into many subdomains the single, large PINN optimisation problem is turned into many smaller subdomain optimisation problems. By using separate subdomain normalisation (as well as a domain decomposition appropriate for the complexity of the solution) we ensure that the effective solution frequency each subdomain optimisation problem sees is low. The use of flexible training schedules allows us to focus on solving smaller parts of the domain sequentially instead of all of them at once. Our hypothesis is that all three of these strategies alleviate the scaling issues described above, leading to an easier global optimisation problem.

With any domain decomposition technique it is important to ensure that the individual neural network solutions are communicated and match across the subdomain interfaces. In FBPINNs, during training the neural networks share their solutions in the overlap regions between subdomains, and by mathematically constructing our global solution as the sum of these solutions we automatically ensure the solution is continuous across subdomains. This approach allows FBPINNs to use a similar loss function to PINNs, without requiring the use of additional interface loss terms, in contrast to other domain decomposition approaches (e.g. Jagtap and Karniadakis [2020]).

Finally, FBPINNs can be trained in a highly parallel fashion, decreasing their training times, and we present an algorithm for doing so in Section 6.4.4. FBPINNs are inspired by classical finite element methods (FEMs), where the solution of the differential equation is expressed as the sum of a finite set of basis functions with compact support, although we note that FBPINNs use the strong form of the governing equation as opposed to the weak form in FEMs.

6.4.2 Mathematical description

We will now give a detailed mathematical description of FBPINNs. In FBPINNs, the problem domain $\Omega \subset \mathbb{R}^d$ is subdivided into n overlapping subdomains, $\Omega_i \subset \Omega$. FBPINNs can use any type of subdivision, regular or irregular, with any overlap width, as long as the subdomains overlap. An example subdivision, a regular hyperrectangular division, is shown in Figure 6.2 (a). For simplicity, we use this type of division for the rest of this work.

Given a subdomain definition, the following FBPINN solution ansatz is used to define the approximate solution to the problem defined by Equation 2.2

$$\hat{u}(x;\theta) = \mathcal{C}\left[\ \overline{NN}(x;\theta)\ \right] \ , \tag{6.8}$$

where

$$\overline{NN}(x;\theta) = \sum_i^n w_i(x) \cdot \text{unnorm} \circ NN_i \circ \text{norm}_i(x) \ , \tag{6.9}$$

$NN_i(x;\theta_i)$ is a separate neural network placed in each subdomain Ω_i, $w_i(x)$ is a smooth, differentiable window function that locally confines each network to its subdomain, $\mathcal{C}$ is a constraining operator which adds appropriate "hard" constraints to the ansatz such that it satisfies the boundary conditions (following the same procedure described in Section 6.2.4), norm_i denotes separate normalisation of the input vector x in each subdomain, unnorm denotes a common unnormalisation applied to each neural network output, and $\theta = \{\theta_i\}$.

The purpose of the window function is to locally confine each neural network solution, NN_i, to its subdomain. Any differentiable function can be used, as long as it is (negligibly close to) zero outside of the subdomain and greater than zero

within it. For the hyperrectangular subdomains studied in this paper, we use the following window function;

$$w_i(x) = \prod_j^d \phi((x^j - a_i^j)/\sigma_i^j)\phi((b_i^j - x^j)/\sigma_i^j) \ , \tag{6.10}$$

where j denotes each dimension of the input vector, a_i^j and b_i^j denote the midpoint of the left and right overlapping regions in each dimension (where $a_i^j < b_i^j$, i.e. the black lines in Figure 6.2 (a)), $\phi(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function, and σ_i^j is a set of parameters defined such that the window function is (negligibly close to) zero outside of the overlap region. The individual normalisation norm_i in each subdomain is applied by normalising the input vector between $[-1, 1]$ in each dimension over the subdomain before it is input to the network, whilst the common output unnormalisation unnorm is chosen such that each neural network output stays within the range $[-1, 1]$, and depends on the solution itself. Any neural network can be used to define $NN_i(x; \theta_i)$; in this paper for simplicity we only consider fully connected neural networks.

Given the ansatz defined by Equation 6.8, FBPINNs are trained using the unconstrained loss function

$$\mathcal{L}(\theta) = \mathcal{L}_p(\theta) = \frac{1}{N_p} \sum_i^{N_p} || \, \mathcal{D}[\hat{u}(x_i; \theta); \lambda] - f(x_i) \, ||^2 \ , \tag{6.11}$$

using a set of training points $\{x_i\}$ sampled over the full domain Ω. This loss function is the same form as used when training the strongly-constrained PINNs described in Section 6.2.4, and doesn't require the use of additional interface terms by construction of our ansatz. Alternatively, the constraining operator $\mathcal{C}$ in the FBPINN ansatz can be removed and the corresponding "weak" loss function (cf Equation 2.3) used to train FBPINNs, in a similar fashion to PINNs. Note that whilst using subdomain normalisation means that each subdomain network effectively has its own relative coordinate system, the subdomain networks are trained using gradients with respect to the same global input coordinates by simply backpropagating through this normalisation step when evaluating the physics loss.

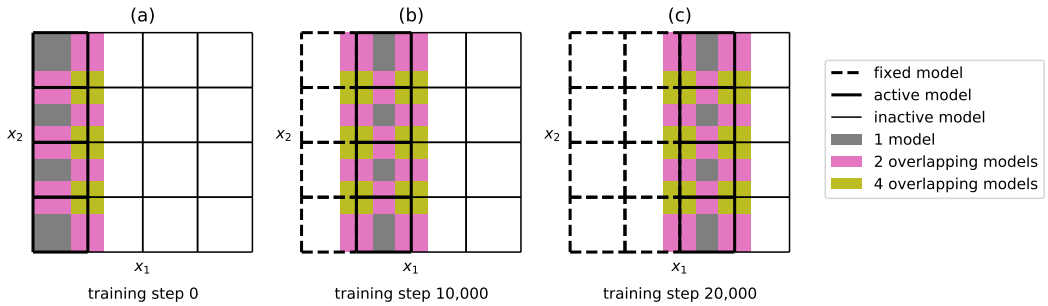


Figure 6.3: Flexible training schedules for FBPINNs. We can design flexible training schedules which can help to improve the convergence of FBPINNs. These schedules define which subdomain networks are updated during each training step. Within these schedules we define "active" models, which are the networks which are currently being updated, "fixed" models, which are networks which have already been trained and have their free parameters fixed, and "inactive" models which are as-of-yet untrained networks. The plots above show one particular training schedule designed to learn the solution "outwards" from the boundary condition, which in this case is assumed to be along the left edge of the domain. Note during each training step only training points from the active subdomains are required, shown by the coloured regions in the plot.

6.4.3 Flexible training schedules

Alongside the issues with scaling PINNs to large domains described in Section 6.3, another issue is the difficulty of ensuring that the PINN solution learnt far away from the boundary is consistent with the boundary conditions. More precisely, it is conceivable that, early-on in training, the PINN could fixate on a particular solution which is inconsistent with the boundary condition at a location in the domain far away from the boundary, because the network has not yet learnt the consistent solution closer to the boundary to constrain it. With two different particular solutions being learned, the optimisation problem could end up in a local minima, resulting in a harder optimisation problem. Indeed, we find evidence for this effect in our numerical experiments (in particular, Sections 6.5.2.4 and 6.5.5). Thus, for some problems it may make sense to sequentially learn the solution "outwards" from the boundary, in a similar fashion to, for example, time-marching schemes employed in classical methods.

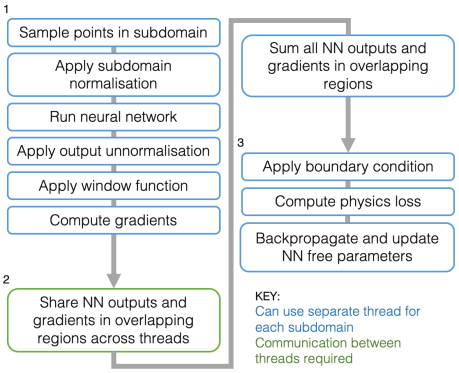
FBPINNs allow for such functionality, which is easy to implement because of their domain decomposition. At any point during training, we can restrict which subdomain networks are updated, and can therefore design flexible training schedules to suit a particular boundary problem. An example training schedule is shown in

```
# 1) Get NN output and gradients in each subdomain
# Can use separate thread for each subdomain
for im in active_or_fixed_neighbour_models:
    x[im] = sample_subdomain_points(im)
    x_norm = (x[im] - x_mu[im]) / x_sd[im]
    u[im] = NN[im](x_norm)
    u[im] = u[im]*u_mu + u_sd
    u[im] = windows[im] * u[im]
    g[im] = problem.get_gradients(u[im], x[im])

# 2) Sum NN outputs in overlapping regions
# Requires communication between threads
for im in active_models:
    for iseg in overlapping_regions[im]:
        for im2 in overlapping_models[iseg]:
            if im2 != im:
                u[im][iseg] += u[im2][iseg].detach()
                g[im][iseg] += g[im2][iseg].detach()

# 3) Apply hard boundary conditions, compute loss,
# backpropagate and update free parameters
# Can use separate thread for each subdomain
for im in active_models:
    u[im], g[im] = problem.boundary(u[im], g[im])
    loss = problem.physics_loss(x[im], u[im], g[im])
    loss.backward()
    optimizers[im].step()
```

(a) `PyTorch` psuedocode for a single FBPINN training step.



(b) Visual schematic of a FBPINN training step inside each subdomain.

Figure 6.4: Parallel algorithm for training FBPINNs. FBPINNs are trained using gradient descent, and the psuedocode for each training step is shown in (a). The effect of each training step on each subdomain is shown in (b). The algorithm can be implemented using entirely independent threads for each subdomain, except during step 2) where they must share their subdomain network outputs within overlapping regions with the threads of their neighbouring subdomains.

Figure 6.3. Within a training schedule we define "active" models, which are the networks which are currently being updated, "fixed" models, which are networks which have already been trained and have their free parameters fixed, and "inactive" models which are as-of-yet untrained networks. During each training step, only active models and their fixed neighbours contribute to the summation in the FBPINN ansatz, and only training points within the active subdomains are sampled.

6.4.4 Parallel implementation

The FBPINN optimisation problem defined by Equation 6.11 is solved by using gradient descent, similar to PINNs. This can be naively implemented within a single, global optimisation loop, but in practice, we can train FBPINNs in a highly parallel

and more data-efficient way by taking advantage of the domain decomposition. In this section we describe a parallel implementation of FBPINNs, for which its pseudocode is shown in Figure 6.4.

There are two key considerations when parallelising FBPINNs. First is that, outside of each neural network's subdomain its output is always zero after the window function has been applied, which means that training points outside of the subdomain will provide zero gradients when updating its free parameters. Thus, only training points within each subdomain are required to train each network, which allows training to be much more data-efficient. Second is that multiple parts of each training step can be implemented in parallel; a separate thread for each network can be used when calculating their outputs and gradients with respect to the input vector, and once the network outputs in overlapping regions have been summed, a separate thread for each network can be used to backpropagate the loss function and update their free parameters. This allows the training time to be dramatically reduced.

We now describe the parallel training algorithm in detail. We use a standard gradient descent training algorithm, consisting of a number of identical gradient descent steps implemented inside a `for` loop. The pseudocode for each training step is shown in Figure 6.4 (a), and a schematic of how each training step affects each subdomain is shown in Figure 6.4 (b). Each training step consists of three distinct steps. First, for each subdomain, training points are sampled throughout the subdomain, normalised and input into the subdomain network. The output of the network is unnormalised, multiplied by the window function, and its appropriate gradients (depending on the specific problem) with respect to the input variables are computed using autodifferentiation. Second, for training points which intersect the overlapping regions between subdomains, the network outputs and gradients are shared across subdomains and summed. Third, for each subdomain, the constraining operator is applied to the summed solution and its gradients, the loss function is computed using these quantities, and the free parameters of the network are updated using backpropagation. We note that because each network has an independent set of free parameters, the computational graph can be discarded (or "detached") for

all outputs shared in its overlapping regions except for those from the current network when backpropagating, allowing the backpropagation operation to be implemented in parallel. Another note is that we must ensure that the same training data points are used in the overlapping regions for each network such that their solutions can be summed.

6.5 Results

6.5.1 Overview of experiments

In this section we carry out a number of experiments to test the accuracy and efficiency of FBPINNs. We are interested in how FBPINNs scale to larger problem sizes, and our experiments range from smaller to larger problems, both in terms of their domain size and dimensionality. Problems with multi-scale solutions are included.

First, in Section 6.5.2, FBPINNs are tested on the motivating 1D example problem presented in Section 6.3 (that of learning the solution to $\frac{du}{dx} = \cos \omega x$). Harder versions of this problem are introduced, including one with a multi-scale solution and another using second order derivatives in the underlying differential equation. In summary we find that FBPINNs are able to accurately and efficiently learn the solutions to these problems when ω is high, significantly outperforming PINNs. In Section 6.5.3, we extend the motivating 1D problem to 2D, learning the solution to the equation $\frac{\partial u}{\partial x_1} + \frac{\partial u}{\partial x_2} = \cos(\omega x_1) + \cos(\omega x_2)$ when ω is high. Again, the FBPINN significantly outperforms the PINN tested. In Section 6.5.4, we test FBPINNs on a standard PINN benchmark problem, which is the (1+1)D viscous time-dependent Burgers equation. In this case the FBPINN matches the accuracy of the PINN tested, whilst being significantly more data-efficient. Finally in Section 6.5.5 we learn the solution to the (2+1)D time-dependent wave equation for a high-frequency point source propagating through a medium with a non-uniform wave speed, which is the most challenging problem studied here. Whilst the PINN tested exhibits unstable convergence, the FBPINN using a time-marching training schedule is able to robustly converge to the solution.

Some of the FBPINN settings and hyperparameters are fixed across all experiments. Specifically, the same optimiser (Adam [Kingma and Ba, 2015]), learning rate (0.001), network type (fully connected), network activation function (tanh), type of subdomain division (hyperrectangular), and window function (as defined in Equation 6.10) are used across all experiments. The relevant corresponding aspects are also fixed and the same across all of the PINN benchmarks used. Other settings and hyperparameters, such as the number of training steps, number of training points, number of hidden layers, number of hidden units, number of subdomains, overlapping width of each subdomain, output unnormalisation and training schedule vary depending on the problem, and for some cases studies we show ablations of them. For clearer comparison, we always use the same training point sampling scheme and density for both PINNs and FBPINNs, the same unnormalisation of their network outputs and the same constraining operator when forming their ansatze. The PINN input variable is always normalised between $[-1, 1]$ in each dimension across the problem domain before being input to the network. The PINNs are implemented within the same coding framework as the FBPINNs, which uses the `PyTorch` library [Paszke et al., 2019]. All 1D problems are trained using a single CPU core, whilst all other problems are trained using a single NVIDIA Titan V GPU. For this work we only use a single thread when training FBPINNs, although this thread does exactly implement the parallel training algorithm described in Section 6.4.4. Evaluating the multi-threaded performance of our parallel algorithm will be the subject of future work.

6.5.2 1D sinusoidal experiments

First, we test FBPINNs using the motivating 1D problem described in Section 6.3 (Equation 6.4). The following FBPINN ansatz is used

$$\hat{u}(x;\theta) = \tanh(\omega x)\overline{NN}(x;\theta) \ , \tag{6.12}$$

which uses the same constraining operator as the PINN ansatz defined in Equation 6.6.

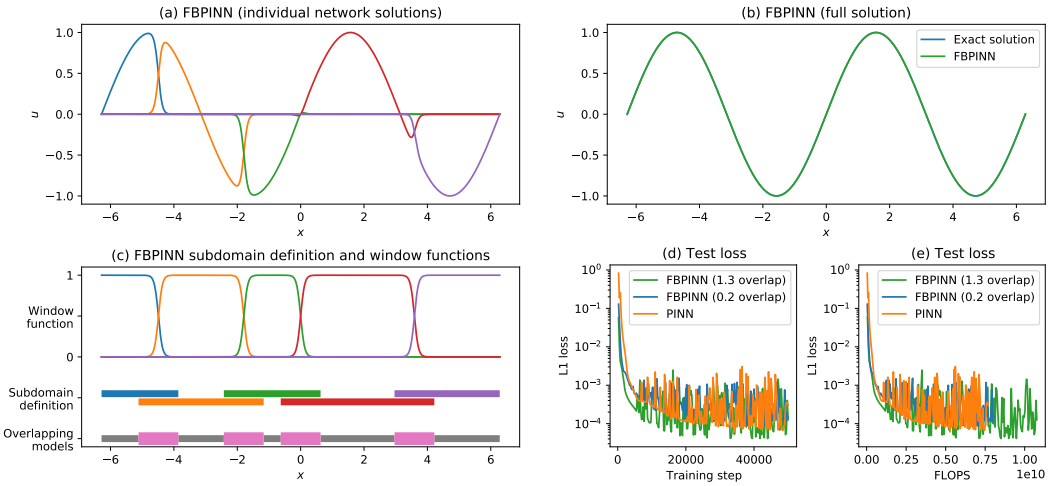


Figure 6.5: Performance of FBPINNs on the motivating problem $\frac{du}{dx} = \cos(\omega x)$ when $\omega = 1$. For this case, we find the FBPINN described in Section 6.5.2.1 has similar performance to the $\omega = 1$ PINN in Section 6.3.1 (shown in Figure 6.1 (a)). The individual FBPINN subdomain solutions after training are shown in (a). The full FBPINN solution compared to the exact solution is shown in (b). The FBPINN subdomain definition, overlapping regions in the domain (thick pink lines), and window function for each subdomain are shown in (c). The L1 error between the FBPINN solution and the exact solution against training step is shown in (d) and (e). Also shown in (d) and (e) are the convergence curves for a FBPINN trained with a smaller subdomain overlap width and the $\omega = 1$ PINN from Section 6.3.1.

6.5.2.1 Low frequency case ($\omega = 1$)

For the low frequency case, the domain $x \in [-2\pi, 2\pi]$ is divided into $n = 5$ overlapping subdomains shown in Figure 6.5 (c). Each subdomain is defined such that all of their overlapping regions have a width of 1.3 and their associated window functions (as defined in Equation 6.10) are also shown in Figure 6.5 (c). Each subdomain network has 2 hidden layers and 16 hidden units per layer. Similar to the PINN in Section 6.3.1, the output of each subdomain network is unnormalised by multiplying it by $\frac{1}{\omega}$ before summation, and the FBPINN is trained using 50,000 training steps and 200 training points regularly spaced over the domain. An "all-active" training schedule is used, where all of the subdomain networks are active every training step. The FBPINN has 1,605 free parameters in total.

Figure 6.5 (a) shows the individual network solutions after training and (b) shows the full FBPINN solution. For ease of interpretation, each individual network plot in (a) shows the output of each subdomain network just before summation but

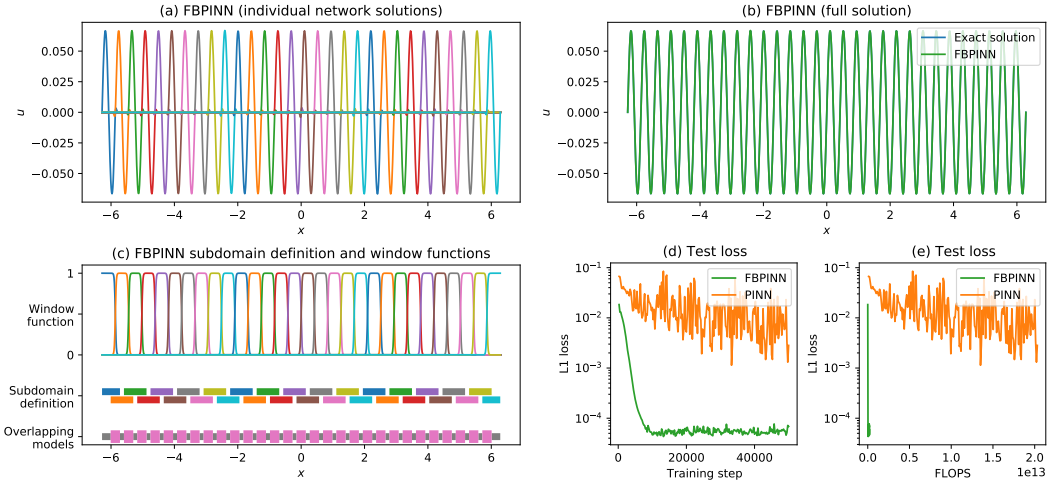


Figure 6.6: Performance of FBPINNs on the motivating problem $\frac{du}{dx} = \cos(\omega x)$ when $\omega = 15$. The FBPINN described in Section 6.5.2.2 is compared to the best-performing $\omega = 15$ PINN in Section 6.3.2 (namely the PINN with 5 layers and 128 hidden units, shown in Figure 6.1 (d)). For this case, we find the FBPINN significantly outperforms the PINN, converging to the solution with much higher accuracy and much less training steps. This plot has the same layout as Figure 6.5.

with the constraining operator $(\tanh(\omega x)\cdot)$ applied. Figure 6.5 (d) compares the L1 convergence curve of the FBPINN to the L1 convergence curve of the low-frequency PINN (with 2 layers and 16 hidden units) studied in Section 6.3.1. Figure 6.5 (e) shows the same curve against the cumulative number of training floating point operations (FLOPs) required during forward inference of the subdomain networks, which is a measure of data-efficiency[1]. For this case study, we find that the FBPINN is able to solve the problem as accurately and with a similar data-efficiency to the PINN.

For this case study we also test the sensitivity of the FBPINN to different subdomain overlap widths. Figure 6.5 (d) and (e) show the convergence curve for the same FBPINN but with its subdomains defined such that all overlapping regions have a width of 0.2. In this case the FBPINN has similar performance.

6.5.2.2 High frequency case ($\omega = 15$)

Next we test the performance of the FBPINN when $\omega = 15$ which is a much harder problem, as discussed in Section 6.3. For this case we divide the domain into $n = 30$

[1]Note, this measure only counts FLOPs spent during the forward inference of the networks, and does not count FLOPs spent during gradient computation, backpropagation or any other part of the training algorithm. See D.1 for the exact formula used.

equally spaced subdomains with overlapping widths of 0.3, as shown in Figure 6.6 (c). The subdomain network size is kept the same as the case above at 2 layers and 16 hidden units per layer, and the same "all-active" training schedule is used. The FBPINN has 9,630 free parameters in total. Similar to the high-frequency PINNs tested in Section 6.3.2, the number of regularly spaced training points is increased to $200 \times 15 = 3000$. We compare the FBPINN to the best performing high-frequency PINN from Section 6.3.2, namely the PINN with 5 layers and 128 hidden units.

Figure 6.6 shows the same plots as Figure 6.5 for this case. We find that, in stark contrast to the PINN, the FBPINN is able to converge to the solution with very high accuracy in very few training steps. Furthermore training the FBPINN requires multiple orders of magnitude less forward inference FLOPs than the PINN. This is because it uses much smaller network sizes in each of its subdomains, dramatically reducing the amount of computation required.

6.5.2.3 Multi-scale case

We extend the difficulty of this problem by including multi-scale frequency components in its solution. Specifically, we consider the modified problem

$$\begin{aligned} \frac{du}{dx} &= \omega_1 \cos(\omega_1 x) + \omega_2 \cos(\omega_2 x) \ , \\ u(0) &= 0 \ , \end{aligned} \tag{6.13}$$

which has the exact solution

$$u(x) = \sin(\omega_1 x) + \sin(\omega_2 x) \ . \tag{6.14}$$

Here we chose $\omega_1 = 1$ and $\omega_2 = 15$, i.e. the solution contains both a high and low frequency component.

The same FBPINN and PINN from Section 6.5.2.2 are retrained for this problem, except that their loss functions are modified to use the differential equation above, unnormalisation is applied to both by multiplying their network outputs by 2, and ω_2 is used in their ansatz constraining operator $(\tanh(\omega_2 x)\cdot)$. For this case we also train PINNs with smaller network sizes, namely 2 layers and 16 hidden units, and 4 layers and 64 hidden units.

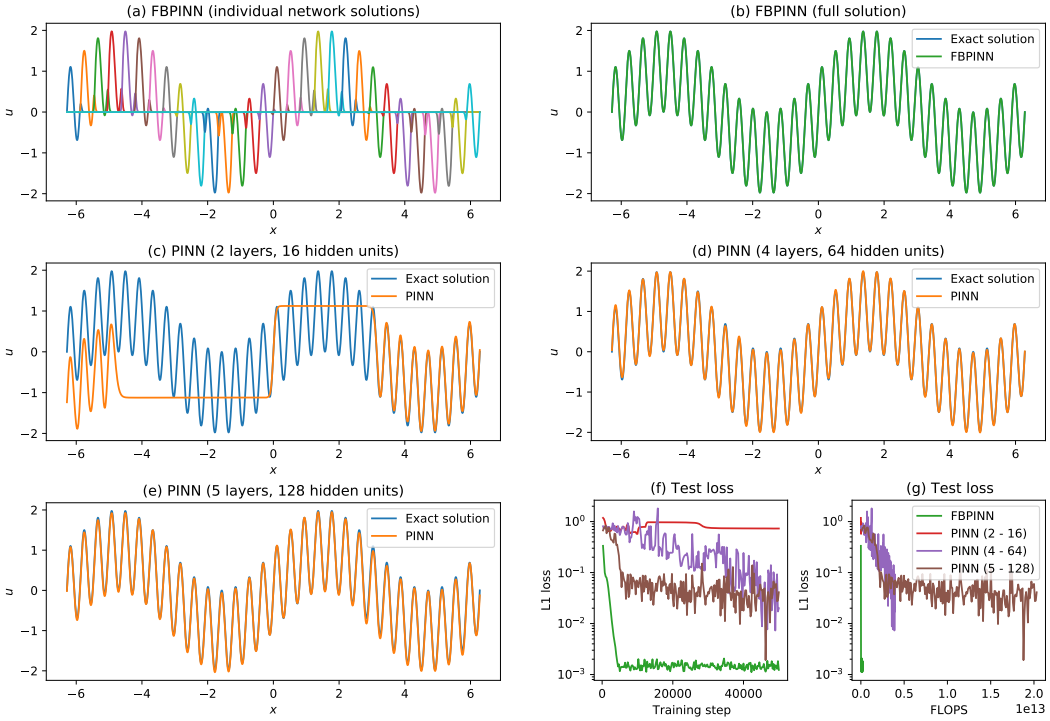


Figure 6.7: Performance of FBPINNs on the multi-scale problem $\frac{du}{dx} = \omega_1 \cos(\omega_1 x) + \omega_2 \cos(\omega_2 x)$ where $\omega_1 = 1$ and $\omega_2 = 15$. The FBPINN described in Section 6.5.2.3 is compared to three different PINNs which have 2 layers and 16 hidden units, 4 layers and 64 hidden units, and 5 layers and 128 hidden units. Similar to Figure 6.6, the FBPINN significantly outperforms the PINNs tested. The individual FBPINN subdomain solutions after training are shown in (a). The full FBPINN solution is shown in (b). The three PINN solutions are shown in (c)-(e). The L1 errors of the FBPINN and PINN solutions compared to the exact solution are shown in (f) and (g).

The FBPINN individual network solutions, the FBPINN full solution and the three PINN solutions are shown in Figure 6.7 (a), (b), (c), (d) and (e) respectively, and their L1 convergence curves are compared in (f) and (g). Similar results are observed to Section 6.5.2.2: in stark contrast to the PINNs, the FBPINN is able to converge to the solution with a much higher accuracy in a much smaller number of training steps. Whilst the PINNs with 4 and 5 layers are able to model all of the cycles in the solution, their accuracy is nearly two orders of magnitude worse than the FBPINN, and their convergence curve is much more unstable.

6.5.2.4 Second order derivative case

We also extend the difficulty of this problem by changing the underlying equation from a first order differential equation to a second order equation. Namely, we

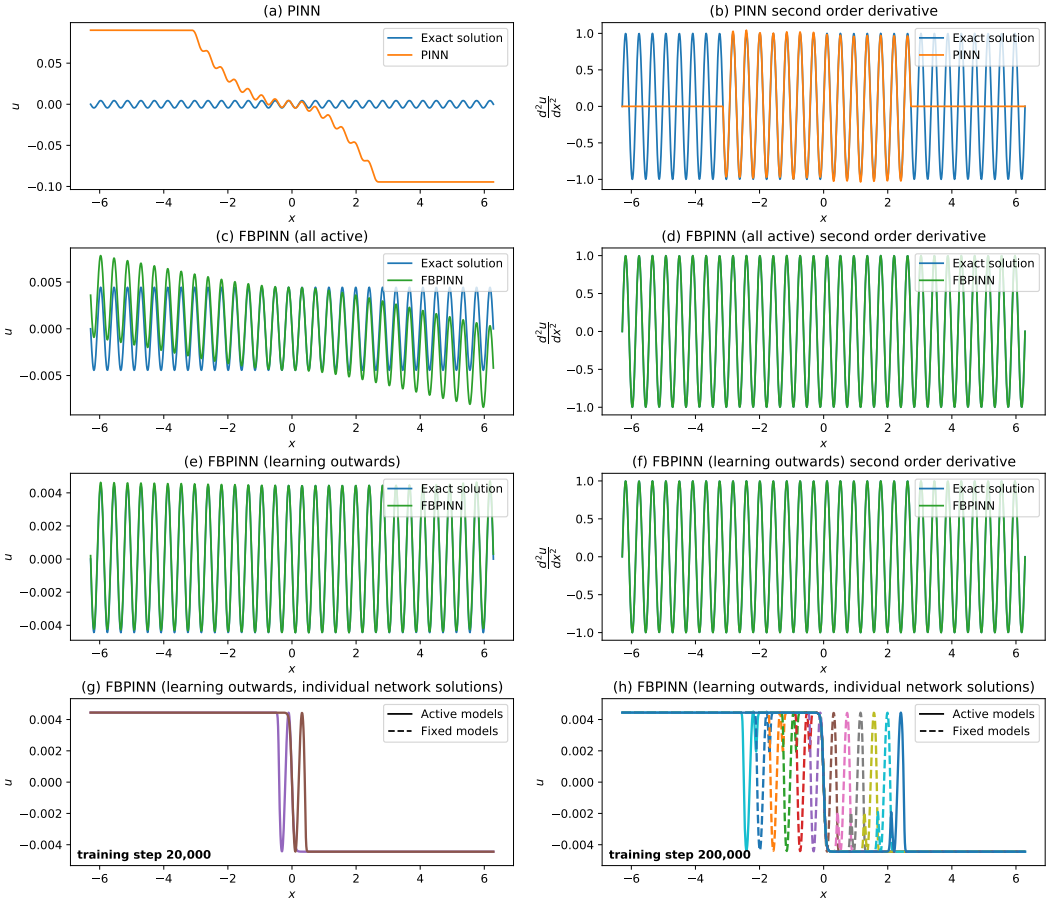


Figure 6.8: Performance of FBPINNs on the problem $\frac{d^2u}{dx^2} = \sin(\omega x)$ with $\omega = 15$. Two FBPINNs with different training schedules are tested for this problem. The first has an "all-active" training schedule, where all models are active all of the time, and its resulting solution and second order derivative are shown in (c) and (d). The second uses a "learning outwards" training schedule which slowly expands the active model outwards from the boundary condition at $x = 0$ as training progresses (as depicted in (g) and (h)), and its solution and second order derivative are shown in (e) and (f). Both FBPINNs use the subdomain definition shown in Figure 6.6 (c), and are compared to a PINN with 5 layers and 128 hidden units, shown in (a) and (b).

consider the related problem

$$\begin{aligned} \frac{d^2u}{dx^2} &= \sin(\omega x) \ , \\ u(0) &= 0 \ , \\ \frac{du}{dx}(0) &= -\frac{1}{\omega} \ , \end{aligned} \tag{6.15}$$

which has the exact solution

$$u(x) = -\frac{1}{\omega^2}\sin(\omega x) \ . \tag{6.16}$$

We use the FBPINN ansatz

$$\hat{u}(x;\theta) = -\frac{1}{\omega^2}\tanh(\omega x) + \tanh^2(\omega x)\overline{NN}(x;\theta) \ , \tag{6.17}$$

such that the boundary conditions are satisfied, and the same construction for the PINN ansatz. Alike Section 6.5.2.2 we consider the high frequency case $\omega = 15$.

The same FBPINN and PINN from Section 6.5.2.2 are retrained for this problem, except for the changes to the problem definition above, and that the outputs of the FBPINN and PINN networks are unnormalised by multiplying them by $\frac{1}{\omega^2}$. We also train both networks for twice as long (100,000 steps).

The resulting FBPINN and PINN solutions are shown in Figure 6.8 (c) and (a), along with their second order derivatives in (d) and (b). We find that both methods struggle to accurately model the solution, although the FBPINN is able to capture all of its cycles. Whilst both models learn the solution accurately in the vicinity of the boundary condition, they learn it poorly outside of it. One explanation is that both models are suffering from integration errors; a small error in the second order derivative (which is being penalised in the loss function) can lead to large errors in the solution. Indeed, both models learn much more accurate second order derivatives, as seen in Figure 6.8. Another explanation is that away from the boundary the FBPINN and PINN are fixating on a different (and incorrect) particular solution of the underlying equation. In particular, away from the boundary the FBPINN solution appears to be superimposed with a linear function of the input variable, which is a feasible solution under this differential equation (but is not consistent with the boundary conditions).

To improve the FBPINN solution further, we retrain the FBPINN using a training schedule that allows the solution to be "learned outwards" from the boundary condition. The schedule starts with only the two models in the center of the domain being active, and then slowly expands the active model outwards in the positive and negative directions, fixing the previously active models behind them, as shown in Figure 6.8 (g) and (h). In this case 500,000 training steps are used (equating to 33,333 training steps per active model). The resulting solution and its second

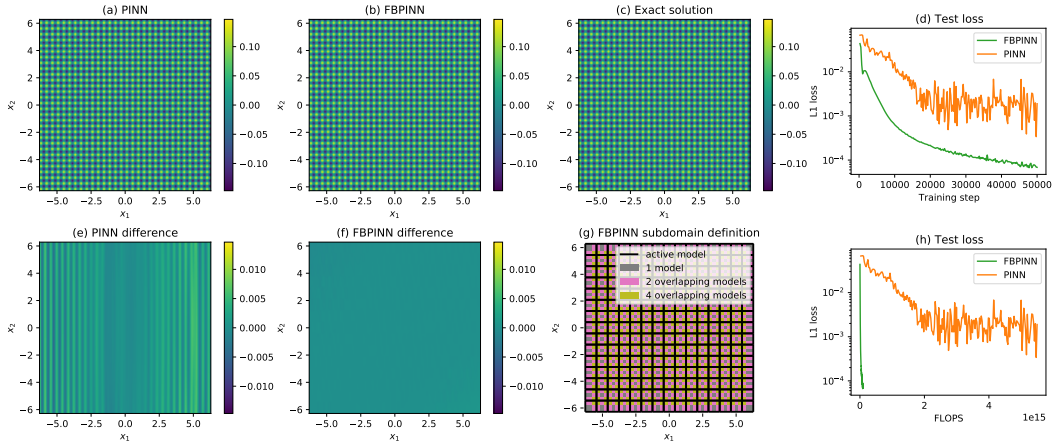


Figure 6.9: Performance of FBPINNs on the problem $\frac{\partial u}{\partial x_1} + \frac{\partial u}{\partial x_2} = \cos(\omega x_1) + \cos(\omega x_2)$ with $\omega = 15$. The FBPINN described in Section 6.5.3 is compared to a PINN with 5 layers and 128 hidden units. Similar to the 1D sinusoidal problems above, we find that the FBPINN significantly outperforms the PINN tested. The FBPINN subdomain definition is shown in (g). The exact solution is shown in (c). The FBPINN solution and its difference to the exact solution are shown in (b) and (f), and a similar set of plots for the PINN are shown in (a) and (e). The L1 errors of the FBPINN and PINN solutions compared to the exact solution are shown in (d) and (h).

order derivative are shown in Figure 6.8 (e) and (f). We find that this FBPINN performs best, accurately modelling the solution many cycles away from the boundary condition, although small errors remain at the edges of the domain.

6.5.3 2D sinusoidal experiments

In this section we study the extension of the motivating problem above from 1D to 2D. Specifically, we consider the problem

$$\begin{aligned} \frac{\partial u}{\partial x_1} + \frac{\partial u}{\partial x_2} &= \cos(\omega x_1) + \cos(\omega x_2) \ , \\ u(0, x_2) &= \frac{1}{\omega} \sin(\omega x_2) \ , \end{aligned} \tag{6.18}$$

where $x = (x_1, x_2) \in \mathbb{R}^2, u \in \mathbb{R}^1$, with a problem domain $x_1 \in [-2\pi, 2\pi], x_2 \in [-2\pi, 2\pi]$. This problem has the exact solution

$$u(x_1, x_2) = \frac{1}{\omega} \sin(\omega x_1) + \frac{1}{\omega} \sin(\omega x_2) \ . \tag{6.19}$$

We use the FBPINN ansatz

$$\hat{u}(x_1, x_2; \theta) = \frac{1}{\omega} \sin(\omega x_2) + \tanh(\omega x_1) \overline{NN}(x_1, x_2; \theta) \ , \tag{6.20}$$

such that the boundary conditions are satisfied, using the same constraining operator for the PINN ansatz. Similar to Section 6.5.2.2 we consider the high frequency case $\omega = 15$. Note that the solution along the second dimension x_2 is already provided in the ansatz, and so the FBPINN and PINN only need to learn to correct the ansatz along the first dimension (although x_2 is still input to the networks and so they could still learn an incorrect solution along the second dimension).

For the FBPINN we divide the 2D domain into $n = 15 \times 15 = 225$ equally spaced subdomains with overlapping widths of 0.6, as shown in Figure 6.9 (g). Each subdomain network has 2 layers and 16 hidden layers, and we use the "all-active" training schedule defined above. For the PINN a network with 5 layers and 128 hidden units is chosen. The FBPINN has 75,825 free parameters whilst the PINN has 66,561 free parameters. The FBPINN and PINN are both unnormalised by multiplying their network outputs by $\frac{1}{\omega}$, and both are trained using $900 \times 900 = 810{,}000$ training points regularly spaced throughout the domain and 50,000 training steps. 1000×1000 regularly sampled test points throughout the domain are used when computing their L1 error compared to the exact solution.

Figure 6.9 (c) shows the exact solution, (b) and (a) show the FBPINN and PINN solutions, and (f) and (e) show their difference to the exact solution. The FBPINN and PINN convergence curves are compared in Figure 6.9 (d) and (h). Similar observations to Section 6.5.2.2 can be made, namely; the FBPINN is able to converge to the solution with much higher accuracy and much less training steps than the PINN; whilst the PINN is able to model all of the cycles of the solution, its accuracy is over one order of magnitude worse than the FBPINN; because a much smaller subdomain network size is used in the FBPINN, training the FBPINN requires multiple orders of magnitude less forward inference FLOPs than the PINN.

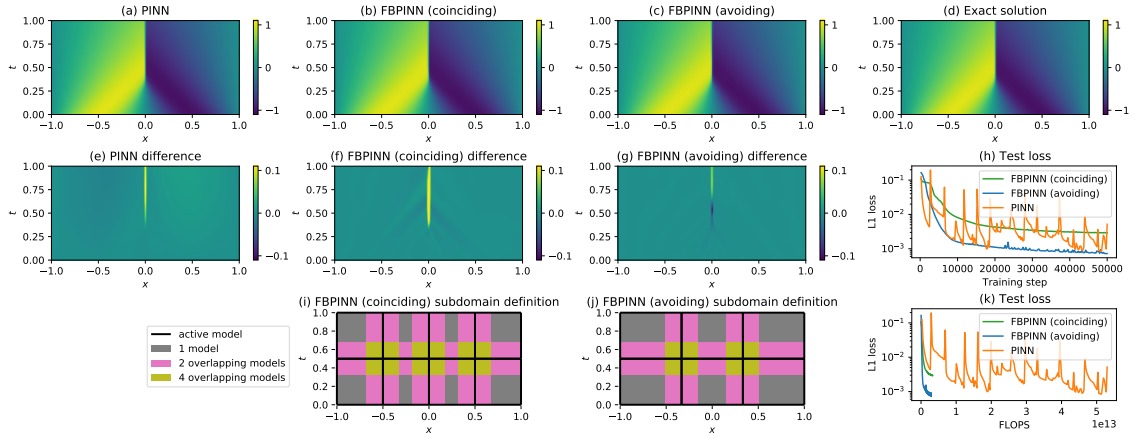


Figure 6.10: Performance of FBPINNs on the (1+1)D viscous time-dependent Burgers equation. The exact solution is shown in (d). Two FBPINNs with different subdomain definitions are tested. The first uses a subdomain definition where the subdomain interfaces coincide with the discontinuity in the exact solution, shown in (i). The second uses a definition where the interfaces avoid the discontinuity, shown in (j). Both FBPINNs are compared to a PINN with 4 layers and 64 hidden units. The coinciding FBPINN solution and its difference to the exact solution are shown in (b) and (f). Similar sets of plots for the avoiding FBPINN and PINN are shown in (c) and (g), and (a) and (e) respectively. The L1 errors of the FBPINN and PINN solutions compared to the exact solution are shown in (h) and (k). We find that the coinciding FBPINN has slightly worse accuracy than the PINN, whilst the avoiding FBPINN has slightly better accuracy.

6.5.4 (1+1)D Burgers equation

In this section the FBPINN is tested using a standard PINN benchmark problem, which is the (1+1)D viscous time-dependent Burgers equation, given by

$$\begin{aligned} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} &= \nu \frac{\partial^2 u}{\partial x^2} \ , \\ u(x,0) &= -\sin(\pi x) \ , \\ u(-1,t) &= 0 \ , \\ u(+1,t) &= 0 \ , \end{aligned} \tag{6.21}$$

where $x, t, u, \nu \in \mathbb{R}^1$, with a problem domain $x \in [-1, 1], t \in [0, 1]$. Interestingly, for small values of the viscosity parameter, ν, the solution develops a discontinuity at $x = 0$ as time increases. We use $\nu = 0.01/\pi$ such that this is the case. The exact solution is analytically available by use of the Hopf-Cole transform (see Basdevant et al. [1986] for details, which are omitted here for brevity).

To solve the problem, we use the FBPINN ansatz

$$\hat{u}(x,t;\theta) = -\sin(\pi x) + \tanh(x+1)\tanh(x-1)\tanh(t)\overline{NN}(x,t;\theta)\ , \tag{6.22}$$

such that the boundary conditions are satisfied, using the same constraining operator for the PINN ansatz.

For the FBPINN we divide the 2D domain into $n = 4 \times 2 = 8$ equally spaced subdomains with overlapping widths of 0.4, as shown in Figure 6.10 (i). We purposefully coincide the subdomain interfaces with the discontinuity in the solution at $x = 0$, to test how the domain decomposition affects the solution accuracy across the discontinuity. Each subdomain network has 2 layers and 16 hidden layers, and we use the "all-active" training schedule. For the PINN a network with 4 layers and 64 hidden units is used, as in testing we found smaller networks performed worse. The FBPINN has 2,696 free parameters whilst the PINN has 12,737 free parameters. The FBPINN and PINN are both unnormalised by multiplying their network outputs by 1, and both are trained using $200 \times 200 = 40{,}000$ training points regularly spaced throughout the domain and 50,000 training steps. 400×400 regularly sampled test points throughout the domain are used when computing their L1 error compared to the exact solution.

Figure 6.10 (d), (b) and (a) show the exact, FBPINN and PINN solutions respectively, and (f) and (e) show the difference of the FBPINN and PINN solutions to the exact solution. Figure 6.10 (h) and (k) show the L1 convergence curves of the FBPINN and PINN. We observe that the FBPINN solution is slightly less accurate across the discontinuity than the PINN, although its overall convergence is more stable. Whilst the FBPINN is able to model the discontinuity, it appears that the window function and summation of networks does make it harder for the FBPINN to model the solution in this region. To avoid this issue, we retrain the FBPINN using 6 overlapping subdomains with interfaces which do not coincide with the discontinuity, as shown in Figure 6.10 (j). The resulting solution, difference and convergence curves are shown in (c), (g), (h) and (k), and we find that this FBPINN is able to more accurately model the discontinuity with a slightly higher overall accuracy than the

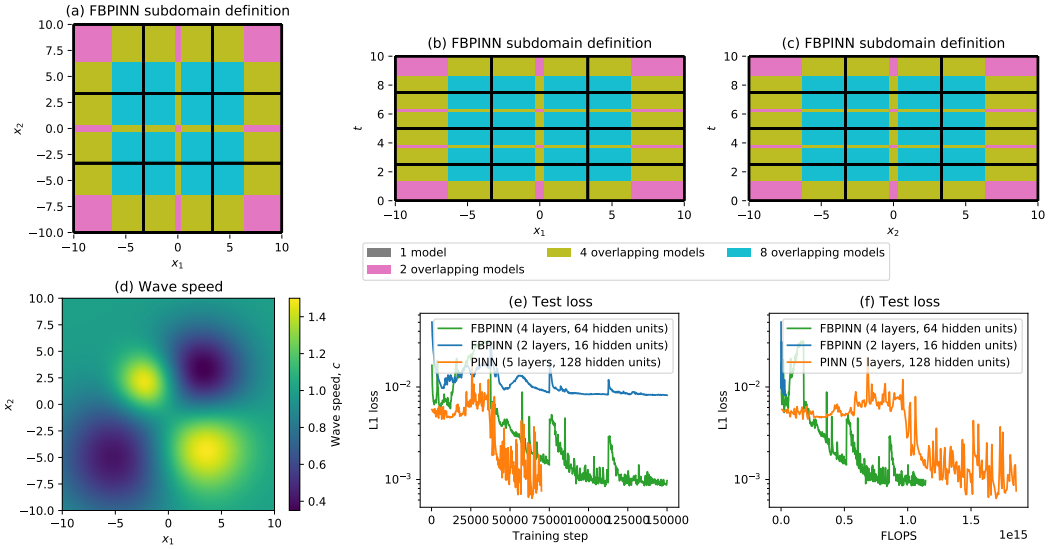


Figure 6.11: Setup and convergence curves for the (2+1)D time-dependent wave equation problem. The FBPINN subdomain definition is shown in (a), (b), and (c), where the plots show orthogonal cross sections through the middle of the domain. The spatially-varying wave speed is shown in (d). The L1 errors of the FBPINN and PINN solutions compared to the solution from finite difference modelling are shown in (e) and (f).

PINN. Furthermore, as observed above, because a much smaller subdomain network size is used in the FBPINNs, training both FBPINNs requires multiple orders of magnitude less forward inference FLOPs than the PINN.

6.5.5 (2+1)D wave equation

Finally, we test the FBPINN using the (2+1)D time-dependent wave equation, modelling the dynamics of an initially stationary point source propagating through a medium with a non-uniform wave speed. Specifically, we solve the following problem

$$\begin{aligned} \left[\nabla^2 - \frac{1}{c(x)^2}\frac{\partial^2}{\partial t^2}\right] u(x,t) &= 0 \ , \\ u(x,0) &= e^{-\frac{1}{2}(||x-\mu||/\sigma)^2} \ , \\ \frac{\partial u}{\partial t}(x,0) &= 0 \ , \end{aligned} \tag{6.23}$$

where $x, \mu \in \mathbb{R}^2$, $t, u, \sigma \in \mathbb{R}^1$, $c(x) \in \mathbb{R}^1$ is the spatially-varying wave speed, and μ and σ control the starting location and central frequency of a Gaussian point source. The wave speed $c(x)$ is defined as a simple mixture of 2D Gaussian distributions and is shown in Figure 6.11 (d). For this case we use the problem domain $x_1 \in$

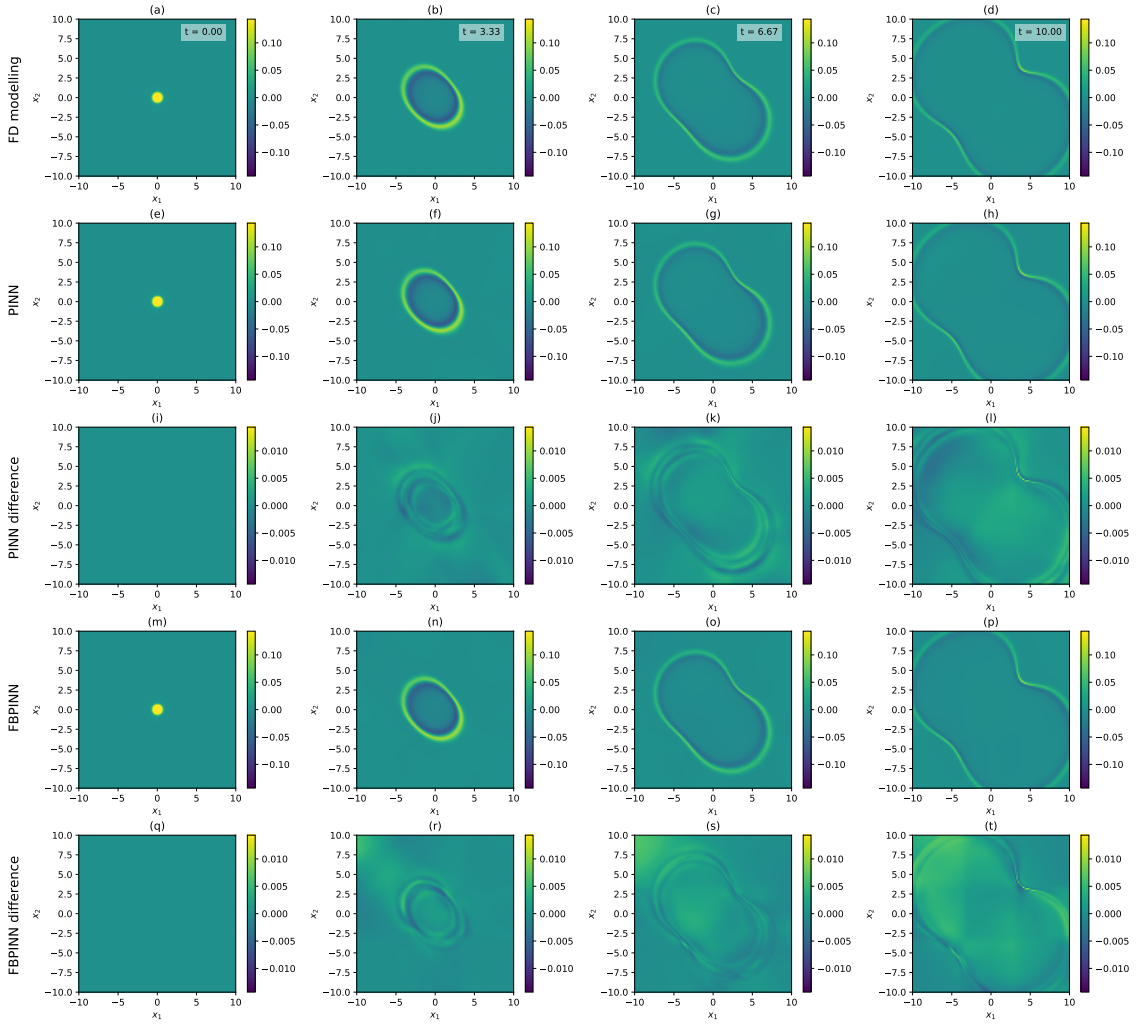


Figure 6.12: Performance of FBPINNs on the (2+1)D time-dependent wave equation problem. The FBPINN described in Section 6.5.5 is compared to a PINN with 5 layers and 128 hidden units. We find that the FBPINN and PINN have similar accuracy, although the FBPINN converges more robustly to the solution (see Figure 6.13). The solution at 4 time-steps spanning the domain from finite difference modelling is shown in (a)-(d). The PINN solution at these time-steps is shown in (e)-(h), and the FBPINN solution at these time-steps is shown in (m)-(p). The difference of the PINN solution to the solution from finite difference modelling is shown in (i)-(l), and similar difference plots for the FBPINN are shown in (q)-(t).

$[-10, 10], x_2 \in [-10, 10], t \in [0, 10]$ and set $\mu = (0, 0)$ and $\sigma = 0.3$. Modelling the wave equation can be challenging in general because its solutions are typically broadband, oscillatory and dispersive in nature, and can include reflections, refractions, wavefront compression and expansion through different velocity regions and a large range of amplitudes [Igel, 2017]. For this case, the solution (or "wavefield") expands outwards from the point source, compressing and expanding as it moves through regions

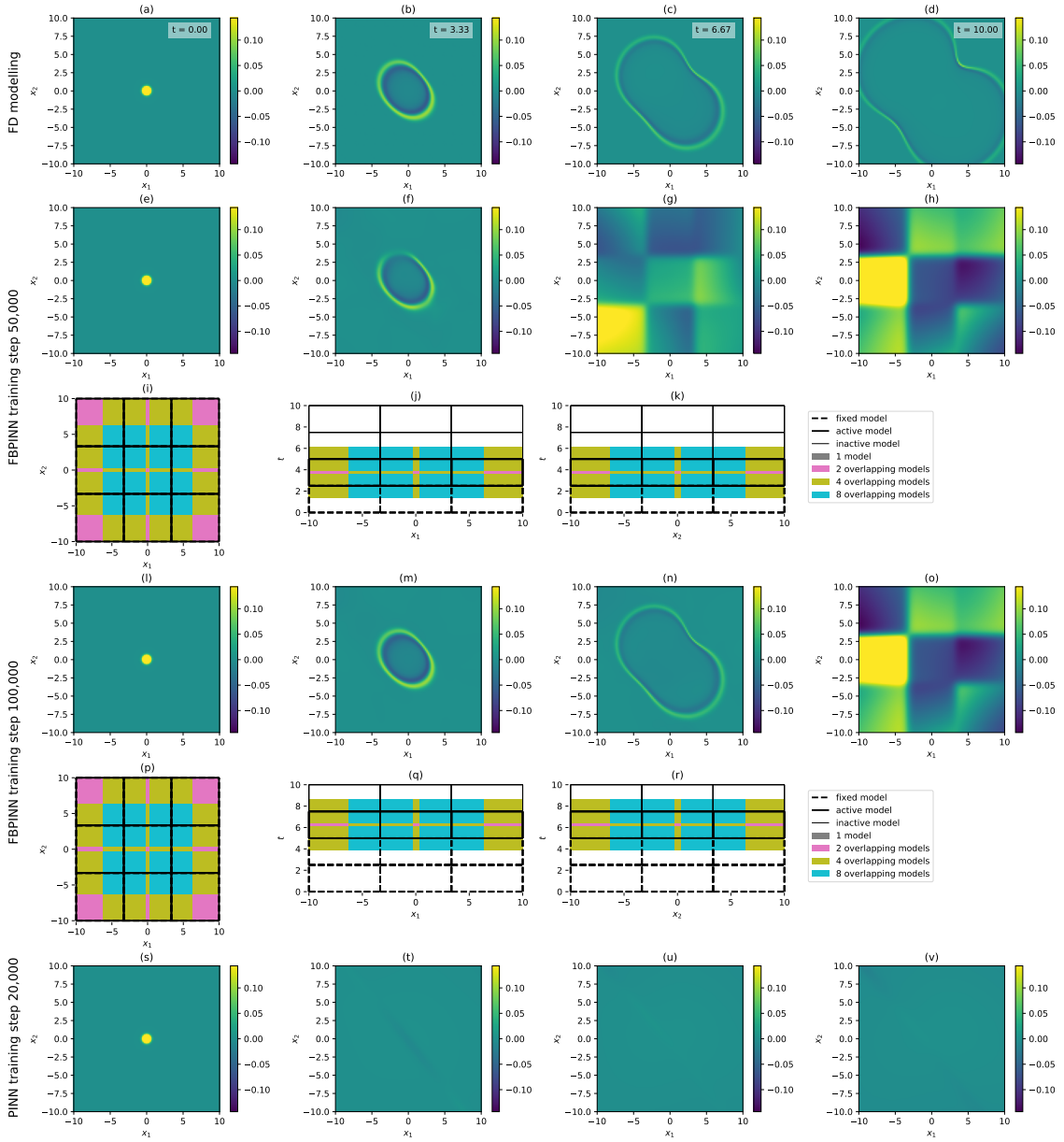


Figure 6.13: FBPINN and PINN solutions during training for the (2+1)D time-dependent wave equation problem. The solution from finite difference modelling is shown in (a)-(d) (repeated from Figure 6.12). The FBPINN solution at training step 50,000 and 100,000 as well as the PINN solution at training step 20,000 are plotted in (e)-(h), (l)-(o) and (s)-(v) respectively using the same 4 time-steps. Whilst the FBPINN robustly learns the solution outwards from $t = 0$ as its time-marching training schedule progresses, even after 20,000 training steps the PINN solution is still close to zero everywhere apart from the boundary condition. Orthogonal cross sections of the active, fixed and inactive FBPINN subdomains through the center of the domain during the time-marching training schedule are shown for each FBPINN training step in (i)-(k) and (p)-(r).

with different wave speeds. We compare our results to the solution from finite difference (FD) modelling (see D.2 for our detailed implementation), which is shown

in Figure 6.12 (a)-(d) for 4 time-steps spanning the domain.

To solve the problem, we use the FBPINN ansatz

$$\hat{u}(x,t;\theta) = \phi(5(2-t/t_1))\, e^{-\frac{1}{2}(||x-\mu||/\sigma)^2} + \tanh^2(t/t_1)\overline{NN}(x,t;\theta)\ , \tag{6.24}$$

where $\phi(\cdot)$ is the sigmoid function and $t_1 = \sigma/c(\mu)$. This ansatz is designed such that at $t = 0$ the sigmoid function is (negligibly close to) 1, allowing the ansatz to match the boundary conditions. At times $t \gg 2\,t_1$, the sigmoid function is (negligibly close to) 0, removing the point source term from the ansatz. This is done so that the FBPINN does not need to learn to correct for this part of the ansatz once the point source has expanded away from its starting location. The same constraining operator is used for the PINN ansatz.

For the FBPINN we divide the 3D domain into $n = 3 \times 3 \times 4 = 36$ subdomains with overlapping widths of 6 in the spatial dimensions and 2 in the time dimension, as shown in Figure 6.11 (a)-(c). Each subdomain network has 4 layers and 64 hidden layers. For this problem we use a "time-marching" training schedule, where initially only subdomains in the first time-step are active, before slowly expanding the active time-step outwards and fixing the earlier time-step models. For the PINN a network with 5 layers and 128 hidden units is chosen. The FBPINN has 460,836 free parameters whilst the PINN has 66,689 free parameters. The FBPINN and PINN are both unnormalised by multiplying their network outputs by 1, and both are trained using $58 \times 58 \times 58 = 195{,}112$ training points randomly sampled throughout the domain. The PINN is trained for 75,000 training steps, whilst the FBPINN is trained for 150,000 steps (equating to 37,500 training steps per active model). For this case we chose random sampling over regular sampling because the training point density is relatively low compared to the frequency of the solution and in testing this allowed for better convergence of both the FBPINN and PINN. $100 \times 100 \times 10$ regularly sampled test points throughout the domain are used when computing their L1 error compared to the finite difference solution.

Figure 6.12 (e)-(h) and (m)-(p) show the resulting PINN and FBPINN solutions respectively over 4 time-steps spanning the problem domain. Figure 6.12 (i)-(l)

and (q)-(t) show their difference compared to the finite difference solution, and Figure 6.11 (e) and (f) show their L1 convergence curves. We find that the FBPINN and PINN solutions have a similar accuracy, although the FBPINN takes roughly half as many forward inference FLOPs to train as the PINN, because its subdomain network size is smaller. We also test a FBPINN using a smaller subdomain network size (2 layers and 16 hidden units, the same as all previous examples) but this does not converge, as shown in the convergence plots in Figure 6.11 (e) and (f). Whilst the PINN appears suitable for this problem, we plot the PINN and FBPINN solutions midway through training in Figure 6.13. We find that whilst the FBPINN robustly learns the solution outwards from $t = 0$ as its time-marching training schedule progresses, even after 20,000 training steps the PINN solution is still close to zero everywhere apart from the boundary condition. This can also been seen in the convergence curve for the PINN in Figure 6.11 (e), where its L1 error actually increases until approximately training step 40,000. One explanation for this is that the PINN is fixating on a different (incorrect) particular solution away from the boundary early-on in training, namely the easier particular solution $u(x, t) = 0$, causing the optimisation to become stuck in a local minima.

6.6 Discussion

The numerical tests above confirm that FBPINNs provide a promising approach for scaling PINNs to large domains. They are able to accurately solve all of the smaller and larger scale problems studied, whilst in many cases the standard PINN struggles. For the problems studied with smaller domains, such as the Burgers equation and the low-frequency sinusoidal problem, the FBPINN generally matches the PINN in performance. For the problems with larger domains, such as the wave equation and high-frequency sinusodial case studies, the FBPINN outperforms the PINN. The largest differences are seen in the high-frequency sinusoidal problems, where across all tests the FBPINN converges with much higher accuracy and much less training steps than the PINN. For the wave equation problem, the FBPINN more robustly converges to the solution. These findings demonstrate that the combined use of domain

decomposition, separate subdomain normalisation and flexible training schedules helps to alleviate some of the major issues related to scaling PINNs to large domains.

FBPINNs also appear to be more data-efficient than standard PINNs. Across all experiments we find that FBPINNs are able to converge using smaller network sizes in their subdomains than the network size required by PINNs. The total number of forward inference FLOPs required during training of FBPINNs only depends on the subdomain network size, and not the number of subdomains (see D.1 for proof), and thus the FBPINNs studied here require much less computation than the PINNs to train them. This is likely because of their "divide and conquer" strategy; each subdomain appears to present an easier optimisation problem which only requires a small number of free parameters to solve. Indeed, FBPINNs with more subdomains and smaller network sizes than those tested could be even more data-efficient, and in the future we plan to study in detail how reducing the subdomain size further affects their accuracy, and whether there is an optimal subdomain and network size to use (e.g. similar to h-p refinement in FEM). Similarly, it would be interesting to understand whether their is an optimal overlap width between subdomains. The optimal configurations of all of these hyperparameters could be found by scanning over them when training FBPINNs, or even by treating them as learnable parameters during training. For the simple low frequency sinusoidal case study we find the overlap width has little effect on accuracy (Figure 6.5), although it may have more of an effect on more complex solutions.

It is important to note that each problem studied requires different configurations of the FBPINN to converge well. For example, for the high-frequency 1D first order sinusoidal problem a FBPINN with an "all-active" training schedule performs well, whilst for the second order variant a FBPINN with a "learning outwards" training schedule is required to learn an accurate solution. In general, the training schedule should be selected for the physics problem at hand. For example, "learning outwards" from a single point in the domain is not appropriate for a problem which has boundary conditions at multiple locations, as the training schedule may not sufficiently communicate the outer boundary conditions to inner subdomain networks

which are fixed at earlier training times. More generic training schedules appropriate for any type of boundary condition could be investigated, for example iteratively switching between learning inwards and outwards in the domain.

Another configuration which the accuracy of the FBPINN is sensitive to is the location of the subdomain interfaces in the domain. Specifically, we find that the FBPINN performs slightly worse for the Burgers equation problem when its subdomain interfaces coincide with the discontinuity in the solution. This may be because the overlapping networks are struggling to communicate their individual solutions over such a high frequency change in the solution. It would be interesting to investigate this further, for example by adding more physics loss training points over the interface, or by using mathematical construction in the FBPINN ansatz to strongly impose discontinuities if their character is known a priori.

Whilst we have focused on the issues related to scaling PINNs to large domains, another important consideration is the scaling of PINNs to higher dimensions. Similar to classical methods, a major challenge is likely to be the exponentially increasing number of (training) points required to sample the domain as the number of dimensions increases. It is important to note that domain decomposition may still help to reduce the complexity of the ensuing optimisation problem, and indeed FBPINNs are effective across all of the 1D, 2D and 3D problems studied. However, FBPINNs still require the same number of training points as standard PINNs and so issues such as the increased computational workload required are likely to remain. Specific to FBPINNs, the number of overlapping models summed in each overlapping region using the hyperrectangular subdivision grows exponentially with the number of dimensions, which could negatively affect the underlying FBPINN optimisation problem. We plan to investigate the scaling of FBPINNs to higher dimensions in future work.

A future direction is to study the performance of the multi-threaded version of FBPINNs in detail. For the single-threaded implementation of our parallel training algorithm used here, the FBPINNs are typically 2 to 10 times slower to train than their corresponding PINNs, despite the FBPINNs being more data-efficient. This is because the single thread updates each subdomain network sequentially, and also

because each subdomain has a small network size and number of training points, meaning that the parallelism of the GPU is not fully utilised. The multi-threaded version (as described in Section 6.4.4) should reduce these training times by a factor proportional to the number of subdomains and yield a significant performance increase. A potential bottleneck is that each subdomain thread must wait for its neighbouring subdomain threads to run before summing the solution in its overlapping regions; this step could be made fully asynchronous by e.g. caching the "latest available" outputs from the neighbouring subdomains instead of necessarily waiting for their outputs at the current training step.

Another important direction is to test FBPINNs using irregular domains and subdomains. This is an essential step in many state-of-the-art classical approaches and FBPINNs are readily extendable in this regard. The same FBPINN framework can be used, and only the subdomain window functions and the functions which sample points from each subdomain and define the neighbours and overlapping regions of each subdomain in the parallel training algorithm (Figure 6.4 (a)) need to be changed. Going further, one could draw inspiration from classical methods where adaptive grids are used to solve multi-scale problems; it may be useful to adaptively change the subdomain definition and/or subdomain network in FBPINNs to dynamically fit to the solution. It is also important to note that in comparison to classical methods, where mesh refinement can be highly non-trivial, this could be relatively simple to implement using the mesh-free environment of FBPINNs.

Many other directions for applying and improving FBPINNs are possible. For example, FBPINNs could be used to solve inverse problems in the same way as PINNs and it would be interesting to compare their performance. There are many other types of differential equations which could be tested. Whilst we use simple fully connected subdomain networks here, other architectures and activation functions could be investigated. Another promising direction would be to use transfer learning within our flexible training schedules, for example by using neighbouring fixed models to initialise the free parameters of newly active models, which may improve accuracy and reduce training times further. Specifically for the wave equation, it would be

interesting to train a FBPINN on the problems studied in Chapter 5 and compare its performance to the models studied there. The same FBPINN from Section 6.5.5 could be used, except with a different input velocity model. In particular, it would be interesting to investigate how accurately the FBPINN models seismic reflections across discrete velocity interfaces, which our PINN struggled to model.

A major goal in the field of SciML is to provide ML tools which are practically useful for real-world problems and can extend or complement existing classical methods. For PINNs, one of the key remaining challenges is computational efficiency; training a PINN typically takes much more computational resources than using finite difference methods or FEM. Specifically for our wave equation problem in Section 6.5.5, training the PINN / single-threaded FBPINN takes of the order of 10 hours on a single GPU, whilst FD modelling takes of the order of 1 minute on a single CPU. We noted above that the data-efficiency of FBPINNs increases as the size of its subdomain network decreases, and therefore with a small enough network size and a multi-threaded implementation FBPINNs may be able to match the efficiency of finite difference methods or FEM. It could also be powerful to combine efforts to learn families of solutions, such as the DeepONets mentioned above, with FBPINNs, allowing multiple large-scale solutions to be learnt without needing to retrain FBPINNs. Ultimately, this may lead to approaches that are faster and more accurate than classical methods, opening up many new potential applications. We also believe that standard benchmarks should be established to allow PINNs and their wide variety of derivatives to be more robustly compared, which will help the field achieve this goal.

6.7 Summary

We have shown that PIML can carry out larger-scale simulations over higher frequencies, although substantial changes to the PINNs used were required in order to do so. In particular, we showed that as the domain size increased, the performance of the PINNs tested significantly degraded due to issues such as the increased complexity of their underlying optimisation problem and the spectral bias of neural

networks. Our proposed framework, FBPINNs, needed to use a combination of domain decomposition, individual subdomain normalisation and flexible training schedules to alleviate these issues. Future work is needed to improve the efficiency of FBPINNs so that they become competitive with traditional simulation approaches, and to understand optimal domain decomposition and training strategies.

7 Conclusions

This thesis provides multiple insights into our central research question (namely: how well do PIML techniques scale to real-world problems?). In this chapter we discuss these insights (Section 7.1) and recommend future research directions (Section 7.2).

7.1 Discussion

Perhaps the most important observation from this work is that significant challenges arose across all the PIML concepts and techniques tested when attempting to scale them from simple to real-world problems. For example, our deep neural networks with physics-informed architectures for simulating seismic waves struggled to scale to more complex and arbitrary Earth models. Whilst they performed well on simple Earth models, their performance decreased when tested on more complex and general Earth models. The PINNs we studied struggled to solve differential equations with larger domains and more complex solutions. Whilst they were able to reliably solve problems with low frequency solutions, they took significantly longer to converge as higher frequencies were introduced. Whilst our VAE for discovering underlying factors of variation in the lunar surface temperature and our PIML approach for denoising lunar surface images both performed well with real data and noise, their underlying models became increasingly difficult to interpret and validate.

In the following sections we discuss the underlying scaling challenges of these PIML approaches, the changes required to alleviate them, and their comparison to the challenges typically encountered when scaling traditional approaches.

7.1.1 Underlying scaling challenges

There were major underlying challenges relating to the scaling issues described above. For example, a key difficulty when scaling our deep neural networks for simulating seismic waves to more complex and arbitrary Earth models was the *limited generalisation ability of the neural networks*. Even though we used physics-informed architectures, the networks we defined were only able to accurately simulate Earth models within their training distribution. As the required generality and complexity of the simulations increased, significantly larger neural networks and more training data were needed. Another difficulty was in appropriately asserting physical constraints in our workflow as the problem complexity increased. Whilst we were able to take advantage of causality when simulating seismic waves in horizontally layered media by using a WaveNet architecture, this architecture was not appropriate when simulating the dynamics in more complex faulted media and it was unclear how best to incorporate this constraint.

A major challenge when scaling the PINNs tested to solve differential equations with more complex solutions was the *increasing difficulty of their underlying optimisation problem*. As the complexity of the solution increased, the number of free parameters and training points required by the PINN increased, leading to a significantly harder and more computationally demanding problem. In addition, the spectral bias of neural networks made the optimisation problem harder. These factors meant that problems with larger domains and higher frequency solutions struggled to converge.

A key challenge when using our VAE for discovering underlying factors of variation in the lunar surface temperature and our PIML approach for denoising lunar surface images was the *"black-box" nature of their neural networks*. Whilst these techniques were able to cope with real data and noise, their neural networks had a distributed

representation. This made it increasingly difficult to interpret their outputs when dealing with more complex underlying processes and real-world noise.

Finally, when using our PIML approach for denoising lunar surface images, a key challenge was the *generation of realistic training data*. This was important to ensure that our algorithm generalised well to real images. However, the real noise contained in the images had a complex character and was composed of many different noise sources. This made it challenging to model when generating training data.

7.1.2 Changes required to scale

Significant changes to our PIML algorithms were required to alleviate some of these scaling challenges. For example, when simulating seismic waves, we needed to significantly change the neural network architecture as the complexity of the Earth model increased. Whilst a WaveNet architecture was effective for simulating seismic waves in horizontally layered media an encoder-decoder architecture was required for simulating seismic waves in more complex faulted media.

When using PINNs to solve differential equations, large alterations were required to allow them to solve problems with large domains. Specifically, we needed to use domain decomposition and flexible training schedules in order to alleviate the issues related to spectral bias and the increasing numbers of free parameters required by the PINN, allowing the proposed FBPINNs to solve problems with large domains much more efficiently than standard PINNs.

When using our PIML algorithm to denoise lunar images, we needed to develop a sophisticated physical noise model of the camera when generating training data to ensure the algorithm generalised well to real images. This model combined both synthetic noise models and noise sampled from real dark frames captured by the camera.

Furthermore, significant software engineering effort was required to make these changes. For example, whilst FBPINNs were more efficient than PINNs, a sophisticated domain decomposition code and a multi-threaded implementation was required to realise this efficiency gain. Similarly, whilst our PIML algorithm for denoising

lunar images was able to significantly enhance real images, robust, multi-threaded code was required to download, clean, arrange, and transform the terabytes of real images used to generate its realistic training dataset.

7.1.3 Comparison to scaling traditional methods

The fact we encountered these scaling challenges is perhaps not surprising given the difficulties encountered when scaling traditional methods to real-world problems. What is interesting, however, is that the scaling challenges of our PIML approaches were of a different nature to those for traditional methods. In particular, they all related to fundamental issues relating to neural networks; their limited generalisation ability, the difficulty of optimising large neural networks and their lack of interpretability. Even though we saw in Chapter 2 that incorporating scientific principles into ML tends to reduce these flaws, it appears that scaling has the reverse effect; i.e. scaling exacerbates them. In addition, it is important to point out the issues encountered heavily depended on the specific problem and PIML technique used. Thus, it is essential to understand the relative benefits and disadvantages of neural networks when replacing traditional methods with PIML workflows.

However, because of their differences to traditional methods, the PIML approaches tested did alleviate some of the scaling issues encountered by traditional methods. For example, our deep neural networks for simulating seismic waves were orders of magnitude faster than FD simulation once trained, significantly reducing the computational resources required. The PINNs we studied presented provided a mesh-free and conceptually simple approach for solving differential equations, potentially removing the need for elaborate discretisation schemes frequently used in traditional methods. Our PIML approach for denoising lunar surface images significantly outperformed the traditional approaches tested and offered a way to learn about the complex real-world noise processes contained within these images.

Therefore, PIML concepts and techniques can be used to alleviate traditional scaling issues where PIML-specific scaling issues are permissible. For example, when carrying out seismic inversion of Earth models given a recorded seismic response, our

deep neural networks for simulating seismic waves could be used as fast surrogate forward models as long as the true Earth model is known to be close to their training distribution. Similarly, our denoised images of the lunar surface can be used for downstream applications as long as the uncertainty stemming from their lack of interpretability can be tolerated.

7.1.4 Other observations

Another positive observation from this thesis, consistent with the observations in Chapter 2, is that the performance of our ML workflows always improved when physical principles were incorporated into them. For example, when simulating seismic waves, our deep neural network with a physics-informed WaveNet architecture performed more accurately than the standard convolutional neural network tested. Furthermore our PINN was able to simulate the entire seismic wavefield at timesteps outside of its training data much more accurately than the same network trained without a physics-informed loss function. When denoising lunar surface images, our physics-informed pipeline performed more accurately on real images than a naive end-to-end neural network trained with the same training data. When discovering the underlying factors of variation in the lunar surface temperature, carrying out thermophysical inversion alongside our VAE helped us to interpret the model learned by the VAE, and increased our confidence in the existing thermophysical model. These observations support the central thesis of the field of SciML, which is that incorporating scientific principles into ML leads to more powerful models.

The addition of physical constraints also reduced the amount of labelled training data required; Table E.1 compares the size of the datasets used to train the models presented in this thesis. The weakly constrained models (such as the conditional encoder-decoder for simulating seismic waves) typically required millions of training examples, whilst the strongest constrained models (such as FBPINNs) required much less.

Furthermore, physical constraints could be successfully applied across many different parts of the ML pipeline. For example, we used physical constraints in the

generation of realistic training data when denoising lunar images, inside the neural network architecture when simulating seismic waves, and within the loss function when studying PINNs. The most appropriate place to include physical constraints appeared to be very dependent on the problem in hand. Furthermore, PIML was successfully applied to many different types of scientific problems; the discovery of underlying processes, inverse problems and simulation tasks.

Finally, we note that this thesis had significant real-world impact beyond the field of PIML. During the course of the thesis, we have improved understanding of the Moon's thermophysical environment, peered into the Moon's permanently shadowed regions with unprecedented detail for the first time, provided new ways to accelerate seismic simulations and open-sourced a general tool for solving differential equations. Thus, despite the limitations described above, our work shows the strong potential of PIML for aiding scientific research.

7.2 Future work

It is clear that it is highly non-trivial to scale current PIML concepts to real-world problems. Whilst the PIML algorithms studied were promising in that they were able to address some of the scaling issues related to traditional methods, other PIML-specific issues arose when testing them on real-world problems and alleviating them required significant changes to the algorithm. In this section we recommend future research directions given these findings.

As PIML concepts improve and mature, more focus should be placed on assessing whether they can scale to real-world problems. Carrying out proof-of-principle studies on simplified, toy problems is useful, but even if a PIML technique improves on a traditional method, given our findings it is likely that a PIML-specific scaling issue will arise when it is tested on a real-world problem. These issues must be addressed if PIML concepts and techniques are to be widely applicable for real-world problems.

Once identified, more research is required to investigate how to alleviate these PIML-specific scaling issues. For the PIML techniques tested, their scaling issues mostly related to the limitations of neural networks. Of these, perhaps the largest

challenge was the limited generalisation ability of neural networks. More specifically, the accuracy of all our methods reduced when their neural networks were given inputs outside of their training distributions. This was true even when the workflow was physics-informed. Thus, one could argue that these networks do not truly "learn" about underlying physical processes. Indeed, the hallmark of a good physics theory is that it is able to make novel predictions outside of its experimental data.

Perhaps what is needed is a shift in representation; instead of learning vector functions, one should learn operator mappings, algebraic expressions, symbolic objects and relations or even entire algorithms which generalise outside of their training data better [Veličković and Blundell, 2021]. A different solution may be to use more computational power; a recent trend in ML is to train highly expressive foundational models with billions of parameters, and then use transfer learning to apply them to specific tasks [Ramesh et al., 2022, Chowdhery et al., 2022]. Some current work suggests similar large-scale models can be used effectively in the sciences [Jumper et al., 2021, Pathak et al., 2022].

A clear trend in the field of SciML is that incorporating physical principles into ML leads to more powerful models. Given the scaling issues observed above, we believe another trend may emerge; where PIML approaches become more integrated with traditional algorithms, not only incorporating physical principles but also traditional implementation ideas, i.e. a shift towards the hybrid approaches presented in Chapter 2. One could think of this as asserting stronger priors on our PIML model, and only learning parts of a traditional algorithm where necessary. For example, when scaling PINNs to solve differential equations with larger domains we used domain decomposition with many neural networks instead of a single neural network. This idea was inspired by traditional finite element approaches and led to an improved performance. Other recent work is tending to do the same (e.g. Lienen and Günnemann [2022]).

It is unclear where this path will converge. For some problems, we may end up where we started, realising that traditional algorithms provide the best approach for solving real-world problems. But, given the differing nature and observed benefits of

ML described above, we believe a combined approach is likely to benefit. Eventually, this could provide more powerful real-world algorithms that tightly combine ML and traditional methods.

The field of SciML is blossoming, and many more research questions were out of the scope of this thesis. These remain key questions for the future, for example, will the best SciML approaches offer general techniques which can be applied across different domains, or will they be highly domain-specific? What is the best balance between hard-coded physical principles and learned components when designing SciML algorithms? Can SciML provide new perspectives and ideas for the field of ML? A truly interdisciplinary effort is required to answer these questions. Thus, SciML remains an exciting and rapidly growing field, with many discoveries yet to come.

Appendices

A

Physically-interpretable unsupervised learning of lunar thermodynamics

A.1 VAE definition

The goal of a VAE is to (implicitly) learn a distribution $p(x)$, $x \in X$, with X a vector space, which is as similar as possible to an unknown target distribution $p^*(x)$, given a set of samples from this distribution $\mathcal{D} = \{x_1, ..., x_n\}$. VAEs make the assumption that x depends on some set of underlying latent variables $z \in Z$, which are distributed according to some known distribution $p(z)$, and model the likelihood $p(x \mid z)$ and an approximation of the posterior distribution $p(z \mid x)$.

To help them learn, VAEs use variational Bayes inference [Gelman et al., 2013]. In this approach, instead of computing the posterior $p(z \mid x)$, which is often intractable, one searches for a more tractable distribution $q(z \mid x)$ which approximates the posterior. This is most commonly found by minimising the Kullback-Leibler (KL) divergence between $q(z \mid x)$ and $p(z \mid x)$, which can be written as

$$\begin{aligned} D_{\mathrm{KL}}(q(z \mid x) \,\|\, p(z \mid x)) &= \mathbb{E}_{q(z|x)} \left[\log \frac{q(z \mid x)}{p(z \mid x)} \right] \\ &= -\mathbb{E}_{q(z|x)} \left[\log p(x \mid z) \right] + D_{\mathrm{KL}}(q(z \mid x) \,\|\, p(z)) + \log p(x) \ . \end{aligned} \tag{A.1}$$

As $p(x)$ is fixed with respect to $q(z \mid x)$ maximising the quantity $\mathbb{E}_{q(z|x)} \left[\log p(x \mid z) \right] - D_{\mathrm{KL}}(q(z \mid x) \,\|\, p(z))$ is equivalent to minimising the KL divergence between $q(z \mid x)$

and $p(z \mid x)$. Furthermore, as the KL divergence is always positive this quantity is a lower bound on $\log p(x)$ and is therefore known as the variational or evidence lower bound (ELBO).

VAEs assume parameterised models of the likelihood and the approximate posterior, given by $p(x \mid z, \theta)$ and $q(z \mid x, \theta)$, where θ represent model parameters. By rewriting Equation A.1, one can write an expression bounding the log-likelihood of the training data,

$$\log p(\mathcal{D} \mid \theta) \geq \sum_{i=1}^{n} \mathbb{E}_{q(z_i|x_i,\theta)} \left[\log p(x_i \mid z_i, \theta)\right] - D_{\mathrm{KL}}(q(z_i \mid x_i, \theta) \,\|\, p(z_i)) \ , \tag{A.2}$$

assuming the training samples are independently and identically distributed. Therefore we can use the maximum likelihood method to simultaneously learn θ and ensure $q(z \mid x, \theta)$ approximates $p(z \mid x)$, given the training data. To be able to compute the ELBO, VAEs typically further assume that

$$p(x \mid z, \theta) = \mathcal{N}(x; f(z; \theta_f), \sigma^2 I) \tag{A.3}$$

$$p(z) = \mathcal{N}(z; 0, I) \tag{A.4}$$

$$q(z \mid x, \theta) = \mathcal{N}(z; g(x; \theta_g), \mathrm{diag}(h(x; \theta_h)^2)) \ , \tag{A.5}$$

where f, g and h are all deep neural networks parameterised by $\theta = \{\theta_f, \theta_g, \theta_h\}$, σ is a hyperparameter and $\mathcal{N}$ denotes a normal distribution. From a deep learning standpoint, the networks $g(x; \theta_g)$ and $h(x; \theta_h)$ can be seen as a non-linear encoder which maps an observation x to possible values of z whilst $f(z; \theta_f)$ can be seen as a non-linear decoder which maps a latent vector z back to possible values of x. We note that in this definition (Equation A.4) the latent variables z are assumed to be independent from one another; in general it is also possible to parameterise $p(z)$ along with $p(x \mid z, \theta)$ and $q(z \mid x, \theta)$. Under the definitions above, the task of maximum likelihood estimation becomes

$$\hat{\theta}_f, \hat{\theta}_g, \hat{\theta}_h = \underset{\theta_f, \theta_g, \theta_h}{\arg\min} \left\{ \sum_{i=1}^{n} \mathbb{E}_{q(z_i|x_i,\theta_g,\theta_h)} \left[\frac{\|x_i - f(z_i; \theta_f)\|^2}{2\sigma^2} \right] + D_{\mathrm{KL}}(q(z_i \mid x_i, \theta_g, \theta_h) \,\|\, p(z_i)) \right\} . \tag{A.6}$$

The KL term in Equation A.6 can easily be computed analytically as it involves two normal distributions. When calculating the expectation over $q(z \mid x, \theta)$, which is

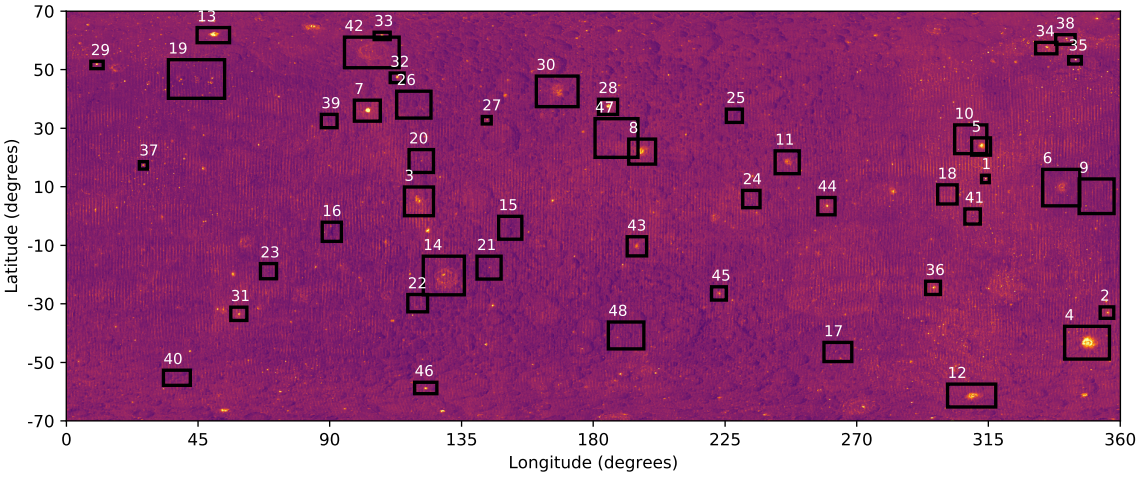


Figure A.1: Locations of the AOIs used for training the VAE, overlain on the latent variable 3 map repeated from Figure 3.8.

usually carried out via sampling, a "reparameterisation trick" is used. One notes that sampling $z \sim q(z \mid x, \theta)$ is equivalent to sampling $\epsilon \sim \mathcal{N}(\epsilon; 0, I)$ and computing $z = h(x; \theta_h)\epsilon + g(x; \theta_g)$, which allows gradient descent methods to be used to find $\hat{\theta}$.

The expectation term in Equation A.6 can be interpreted as a reconstruction loss between the VAE and the training data, whilst the KL term can be interpreted as a regulariser which tries to ensure that the predicted latent variables z are normally distributed. This has the effect of keeping input samples which are similar in X close together in the latent space Z. This regularisation is very useful when one wants to learn "meaningful" representations which vary smoothly over the input space. Typically, one chooses the dimensionality of Z to be much lower than the dimensionality of X, forcing the VAE to learn a salient representation of the input data. Whilst we present a "vanilla" VAE here, many variations have been proposed. One variation is the β-VAE [Higgins et al., 2017]. In this approach the KL term in Equation A.6 is weighted by a hyperparameter $\beta \in \mathbb{R}^+$ which allows more control on the balance between the regularisation and reconstruction terms during training.

A.2 VAE training

Table A.1 lists the AOIs used for training the VAE. The anomalies are classified as 1) thermal anomalies, 2) microwave anomalies, 3) magnetic anomalies, or 4)

Index	Name	Lat	Lon	Class					
1	Marius-A crater	12.58	313.96	1	25	Unnamed crater	34.21	228.28	1
2	Hell-Q crater	-33.00	355.53	1	26	Cantor crater	37.98	118.70	2
3	King crater	4.91	120.47	1	27	Unnamed crater	32.66	143.68	1,2
4	Tycho crater	-43.29	348.67	1,2	28	Moore-F crater	37.23	185.05	1
5	Aristarchus crater	23.70	312.52	1,2	29	Egede-A crater	51.55	10.50	1
6	Copernicus crater	9.60	339.93	1,2	30	Chandler crater	42.46	167.70	1
7	Giordano Bruno crater	35.95	102.88	1,2	31	Furnerius-A crater	-33.51	58.92	1
8	Jackson crater	21.98	196.69	1	32	Rayet-Y crater	47.19	113.05	1
9	Unnamed region	6.72	351.97	1	33	Dugan-J crater	61.45	107.88	1
10	Aristarchus plateau	26.14	308.99	1	34	La Condamine-S crater	57.35	334.78	1
11	Ohm crater	18.30	246.29	1	35	Plato-M crater	53.08	344.54	1
12	Zucchius crater	-61.39	309.32	1	36	Byrgius-A crater	-24.57	296.05	1
13	Thales crater	61.70	50.28	1	37	Dawes crater	17.21	26.36	1
14	Tsiolkovsky crater	-20.38	129.03	1	38	Unnamed crater	60.31	341.33	1
15	Chaplygin-B crater	-4.08	151.67	1	39	Unnamed crater	32.40	89.84	1
16	Unnamed crater	-5.40	90.76	1	40	Hommel-B crater	-55.34	37.79	1
17	Rydberg basin	-46.45	263.59	3	41	Unnamed crater	-0.25	309.53	1
18	Reiner Gamma swirl	7.39	300.97	3	42	Compton crater	55.82	104.44	1
19	Atlas crater	46.78	44.50	1	43	Crookes crater	-10.38	194.98	1
20	Unnamed crater	18.68	121.31	1	44	Pierazzo crater	3.34	259.69	1
21	Unnamed crater	-17.69	144.40	1	45	Das crater	-26.50	222.94	1
22	Unnamed crater	-29.74	120.11	1	46	Fechner-T crater	-58.74	122.80	1
23	Unnamed crater	-18.93	69.14	1	47	Fitzgerald crater	26.62	187.96	4
24	Unnamed crater	5.82	233.99	1	48	Maksutov crater	-40.87	191.31	4

Table A.1: AOIs used for training the VAE. See the appendix text for a description of the assigned classes for each AOI.

Layer	Operation	in chan	out chan	filter size	stride	pad							
1e	Wrap1D	1	1	-	-	1	1d	ConvT1D	n	32	2	2	0
2e	Pad1D	1	1	-	-	3	2d	ConvT1D	32	32	2	2	0
3e	Conv1D	1	32	3	1	1	3d	ConvT1D	32	32	2	2	0
4e	Conv1D	32	32	2	2	0	4d	ConvT1D	32	32	2	2	0
5e	Conv1D	32	32	2	2	0	5d	ConvT1D	32	32	2	2	0
6e	Conv1D	32	32	2	2	0	6d	ConvT1D	32	32	2	2	0
7e	Conv1D	32	32	2	2	0	7d	ConvT1D	32	32	2	2	0
8e	Conv1D	32	32	2	2	0	8d	Conv1D	32	1	1	1	0
9e	Conv1D	32	32	2	2	0	9d	Crop1D	1	1	-	-	-4
10e	Conv1D	32	32	2	2	0							
11e	Conv1D	32	n	1	1	0							
11e	Conv1D	32	n	1	1	0							

Table A.2: VAE network parameters. Each entry shows the parameterisation of each layer in the network. The padding column shows the padding or cropping on each side of the first dimension of each layer's input tensor. The profile input to the network has dimension 120×1 and is padded by 1 at its start and end by wrapping its end values to ensure no edge effects are present at the first and last samples in the first convolutional layer. It is also zero padded by 3 on each side to expand its dimension to 128×1 before being reduced by the encoder. The last two layers of the encoder are two separate convolutional filters which output the mean and standard deviation vectors of the VAE's approximate latent posterior distribution each with dimension $1 \times n$, where n is the number of latent variables. The output of the decoder is a tensor of shape 128×1 which is cropped to 120×1 to match the input profile dimensions.

background. The selection of these training sites is based on surface temperature measurements and their variations in the thermal range [Hayne et al., 2017] and in the microwave range [Zheng et al., 2014], surface roughness and rock abundance

[Bandfield et al., 2011], as well as on magnetic measurements [Tsunakawa et al., 2010]. Thermal and microwave anomalies can either be "hot" or "cold" relative to their surroundings and general environment. Magnetic anomalies show an increased total force in comparison to their immediate surroundings and the general surface of the Moon. The background class contains locations that do not feature any thermal, microwave, or magnetic anomaly. Figure A.1 shows the locations of the AOIs overlain on the latent variable 3 map generated using the VAE.

Table A.2 details the architecture of the VAE. The first convolutional layer of the encoder expands the number of channels (the second dimension) of the input profile from 1 to 32. The 7 subsequent convolutional layers gradually reduce the number of samples (the first dimension) of the input profile to 1, keeping the number of channels constant. The final layer of the encoder consists of two separate convolutional filters which output the mean and standard deviation vectors. The decoder takes a vector of latent values of dimensionality $1 \times n$ and outputs a reconstruction of the input profile. It consists of 7 transposed convolutional layers with 32 channels which gradually increase the number of samples in the latent vector. A final convolutional layer is used to reduce the number of channels from 32 to 1, outputting the reconstructed profile.

A.3 Global maps

In this section we describe how the global maps of the VAE latent values are generated. The first step in our mapping workflow is to run VAE inference and obtain latent values over the Moon's entire sphere. To do this we tessellate the sphere into many latitude-longitude boxes of approximately equal areas. For each box we bin the observed Diviner surface temperature measurements onto a 200×200 m grid, using an orthographic projection centred at the centre of each box, obtaining a profile at each bin location. Each profile is interpolated using GP interpolation, rejecting profiles whose maximum spacing in local lunar time between points is less than 4 hours. The VAE is run on each interpolated profile, obtaining a set of latent values at each bin location, and the central coordinate of each bin is unprojected to obtain a set of latent value latitude-longitude points for each box.

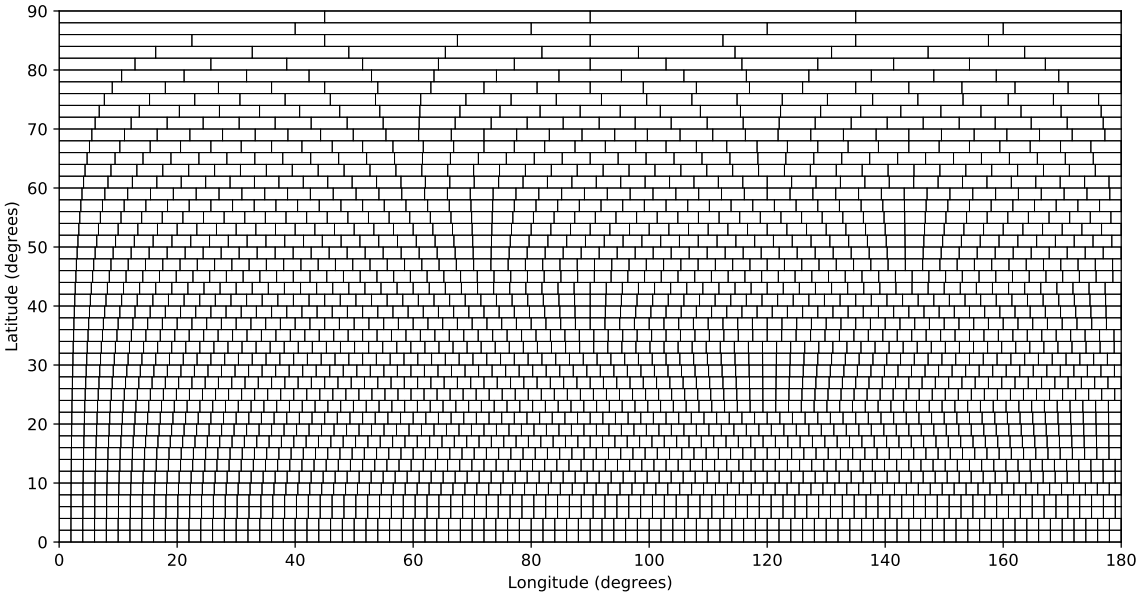


Figure A.2: Tessellation of the Moon's sphere used for generating global maps. We tessellate the sphere into approximately equal area latitude-longitude boxes, keeping the latitude length constant and rounding the longitudinal length in each latitude strip so that an integer number of boxes fit into each strip. 379,448 boxes of approximately 10×10 km shape on the Moon's surface are used for the first VAE inference step and 10,322 boxes of approximately 61×61 km shape are used for the second interpolation step. The first quadrant of boxes for the second step is plotted.

The second step is to project and interpolate the latent value latitude-longitude points onto a global map. We use the standard plate-carrée map projection (equirectangular) and linearly interpolate the points onto a 100×100 m grid. Similar to the first step, we do this by tessellating the sphere into many latitude-longitude boxes of approximately equal area. For each box we project the latent value latitude-longitude points onto the global map and use linear interpolation with Delaunay triangulation to interpolate the points onto the global grid. Interpolated points are set to null values where their Delaunay triangle has a maximum distance between its vertices on the surface of the Moon's sphere greater than 2000 m, to ensure that the map does not include over-interpolated points.

It is important to note that geometrical error in our workflow is introduced in the first step when binning temperature measurements onto the 200×200 m grid, because the bins in the orthographic projection are not exactly equally sized when unprojected onto the surface of the sphere. However, we choose small enough box sizes when

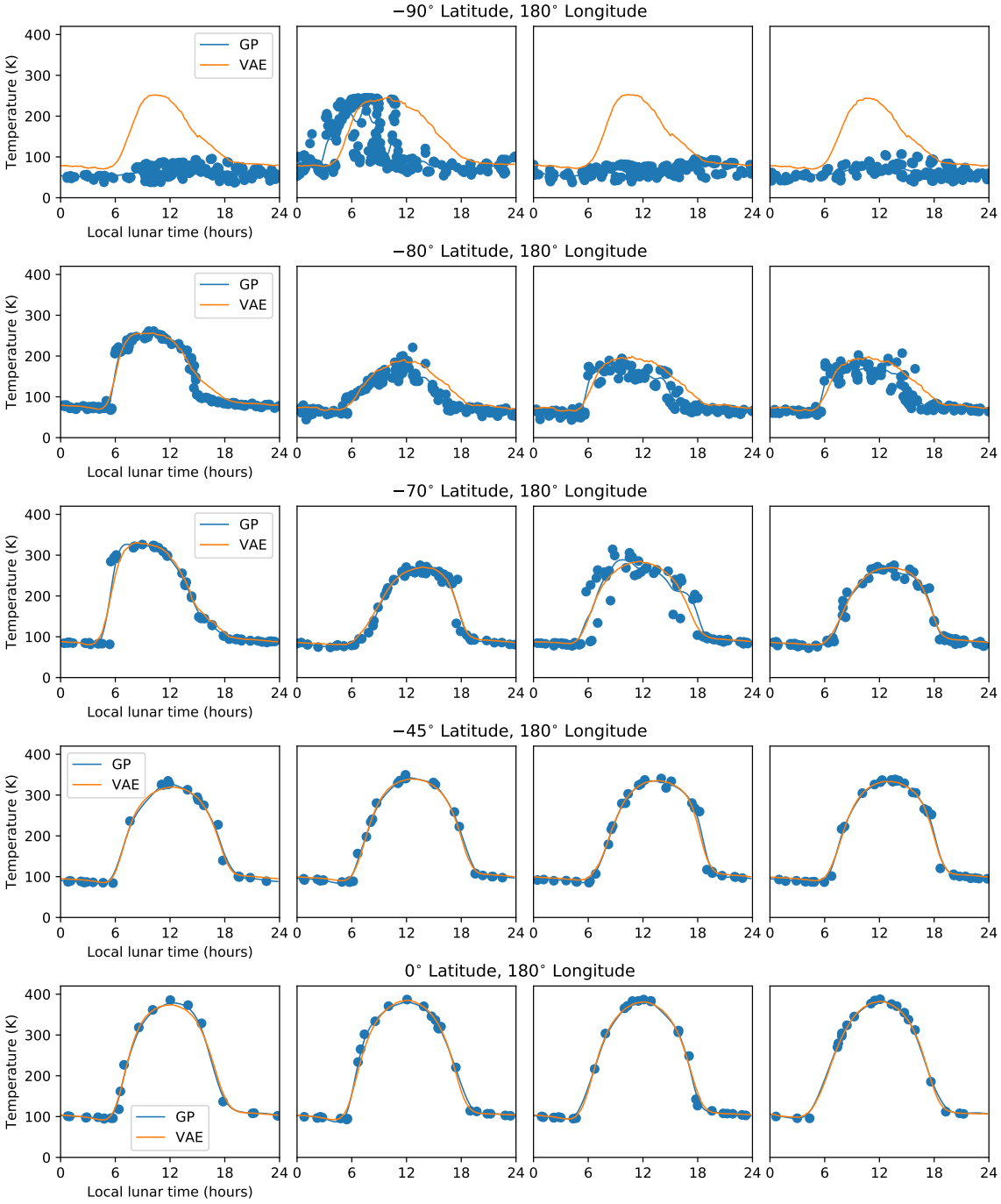


Figure A.3: Example input surface temperature profiles (blue points), GP interpolations (blue lines) and VAE reconstructions (orange lines) with varying latitude. Each row shows four randomly selected example profiles close to the coordinates shown in the row's title. As shown in Figure 3.9, we find that the VAE reconstruction loss is higher closer to the poles. This is expected as the temperature variation at the poles is more complex as a result of the high latitude and the topography present [Williams et al., 2019]. The poles are also more densely sampled than the equator, as can be seen in Figure A.4.

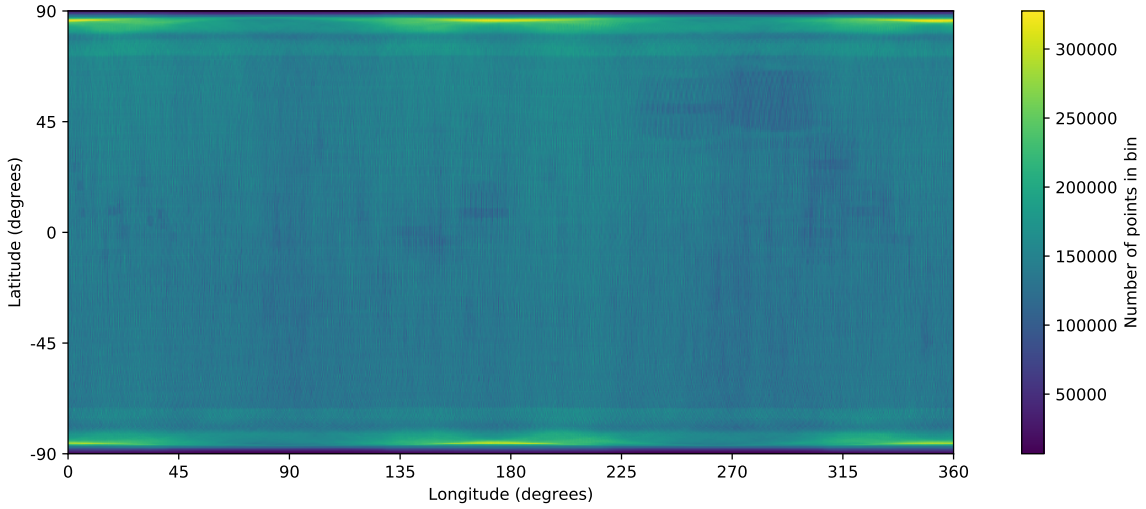


Figure A.4: Population of the 0.5 × 0.5 degree latitude and longitude bins after pre-processing the Diviner data. Over 36 billion point measurements of the lunar surface are extracted from the Diviner instrument spanning 9 years of acquisition. The band of lower coverage at 80N and 80S are caused by systematic Diviner calibration operations.

tessellating the sphere so that this effect is negligible; 379,448 boxes of approximately 10 × 10 km shape on the Moon's surface are used for the first step and 10,322 boxes of approximately 61 × 61 km shape are used for the second step. For both steps we ensure the borders of the boxes are sufficiently overlapping during the processing of each box to ensure that no edge effects are present. Part of the tessellation used for the second step is plotted in Figure A.2. Both steps require considerable computational resource; although tessellation allows the data to be processed in parallel using 100 CPU cores. Our final output products are 4 latent maps and 1 quality control map which shows the L1 reconstruction loss of the VAE. The maps are cropped between -70° and 70° latitude to avoid excessive deformation and areas of high VAE reconstruction loss observed at the poles (see Figure A.3 for more detail).

A.4 Diviner data pre-processing

Figure A.4 shows the population of the 0.5 × 0.5 degree latitude and longitude bins after pre-processing. We observe a higher point density closer to the poles, which is expected because LRO's orbit means the Diviner instrument samples the poles more frequently than the equator.

B

Combining deep learning with a physical model for denoising lunar imagery

B.1 Examples of HORUS denoising on synthetic images

Figure B.1 shows randomly selected examples of HORUS applied to our synthetic test set of images; please see the main text for a detailed description of how these images are generated. For each example, from left to right; the clean image $\tilde{S}$ generated by rescaling a normal sunlit NAC image to low photon counts; the noisy image $\tilde{I}$ after noise from our physical noise model is added; the HORUS denoised image $\hat{S}$; the difference between the clean image and the HORUS denoised image. For each example, the top colorbar is for the clean and HORUS denoised images and shows the mean photon signal, bottom colorbar is for the difference image. The examples are presented in order of decreasing noise. $\tilde{S}$ image credits to LROC/GSFC/ASU.

B.2 Examples of HORUS denoising on real images

Figure B.2 shows more examples of HORUS applied to real NAC images of temporarily- and permanently- shadowed regions on the lunar surface. For each example, left

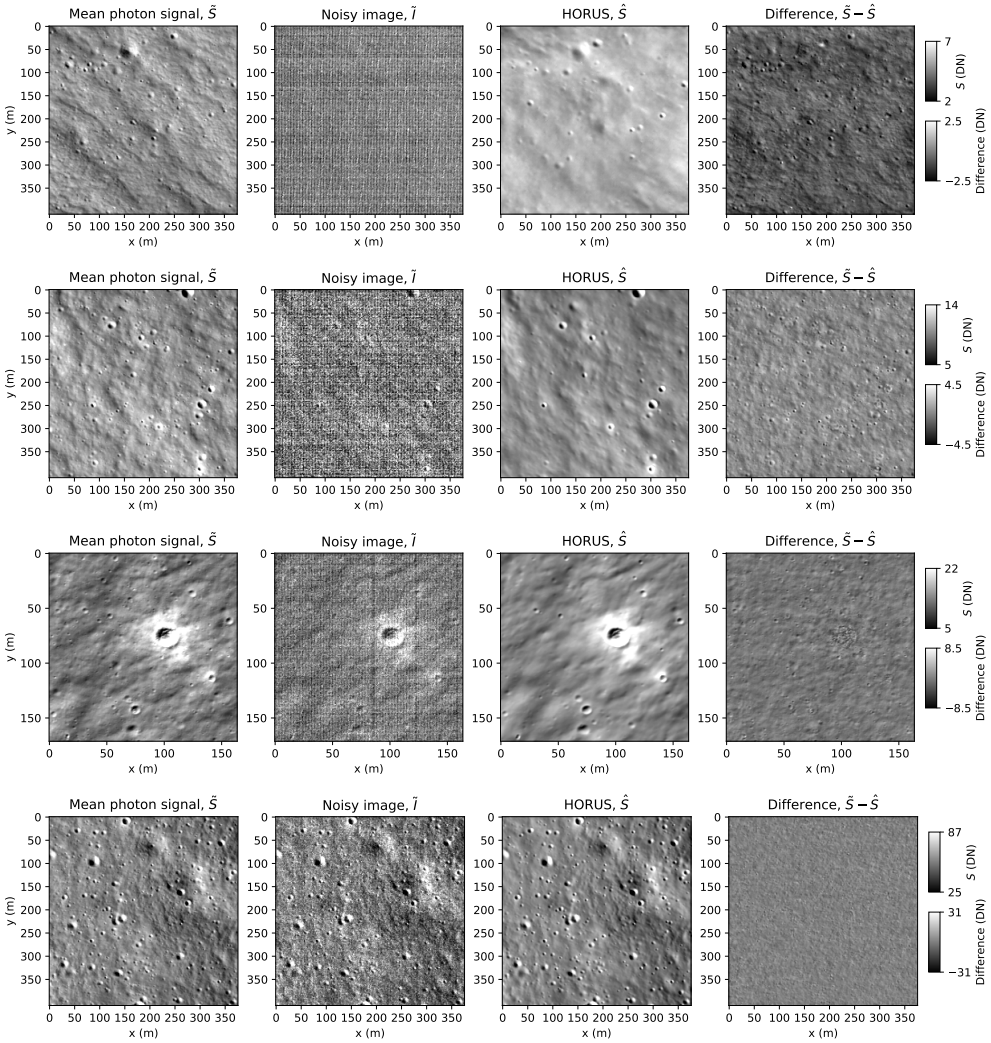


Figure B.1: Examples of HORUS denoising on synthetic images.

shows the input raw NAC Experiment Data Record (EDR) image, middle shows the raw image after the current calibration routine of the camera (ISIS) is applied and right shows the raw image after HORUS is applied. The colorbar is for the HORUS plot and shows the mean photon signal in DN estimated by HORUS. The examples are presented in order of decreasing latitude. Raw/ISIS image credits to LROC/GSFC/ASU.

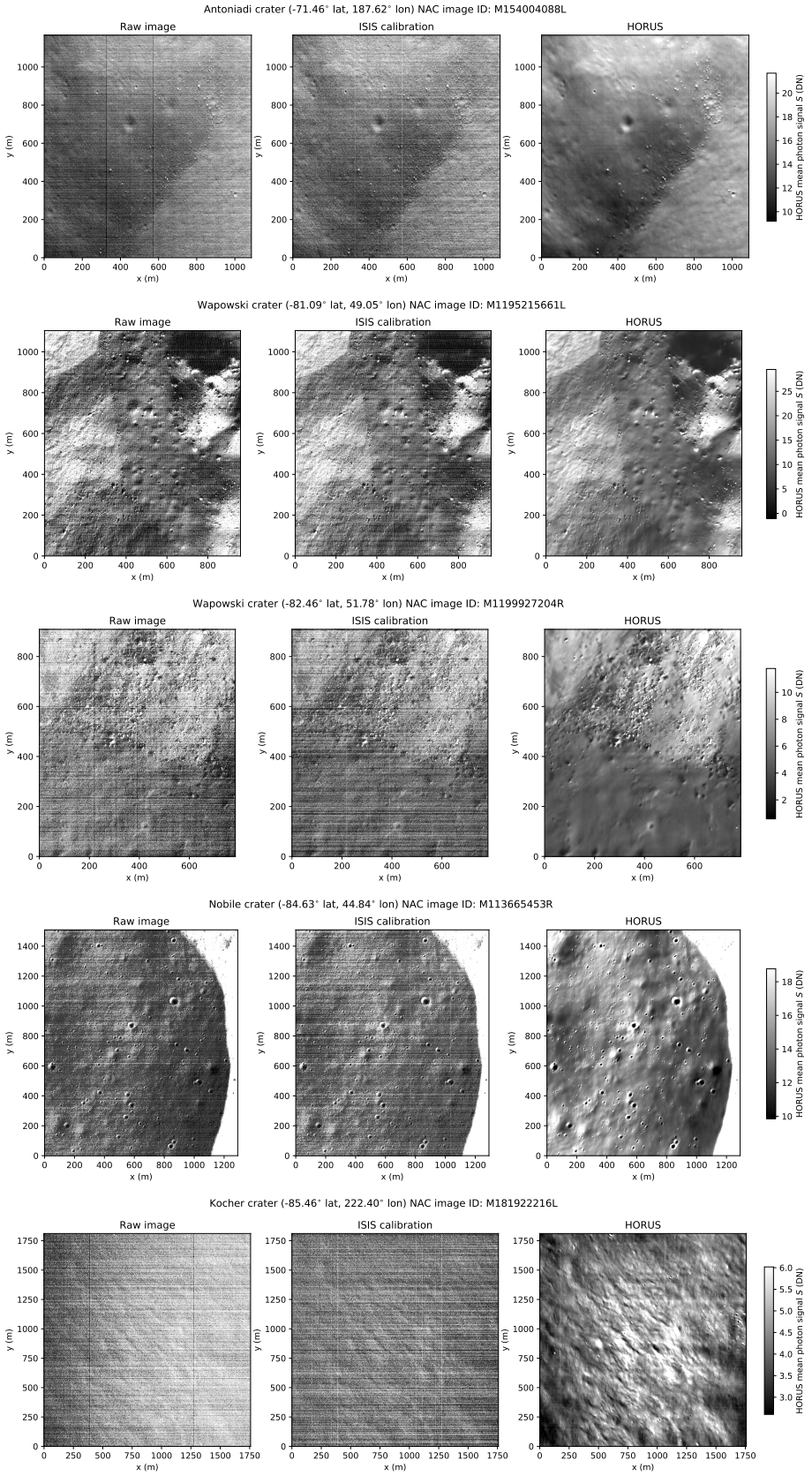


Figure B.2: Examples of HORUS denoising on real images.

C

Physics-informed neural networks for wave simulation

C.1 WaveNet and conditional encoder-decoder comparison and architecture details

Figure C.1 compares the performance of the WaveNet and conditional encoder-decoder networks over the horizontally layered velocity model test set. Table C.1 details the architecture of the conditional encoder-decoder.

C.2 PINN ablation studies

Figure C.3 shows two ablation studies carried out on the PINN used to simulate the full seismic wavefield, namely assessing its performance without prior smoothing of the velocity model and without curriculum learning.

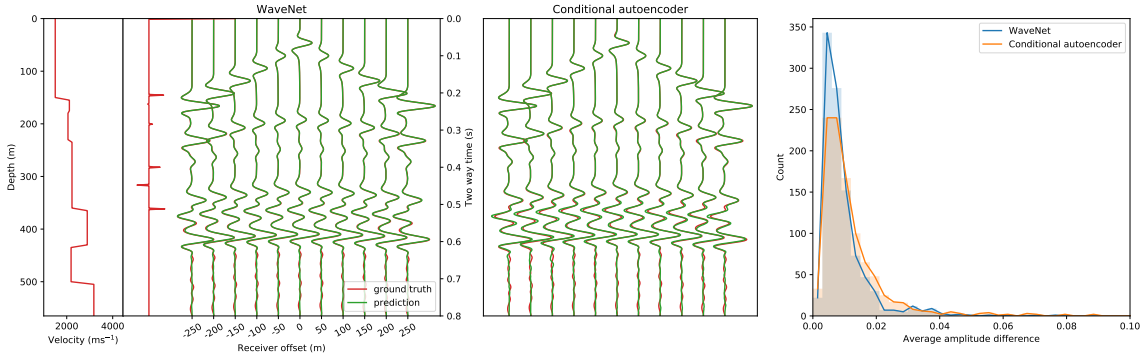


Figure C.1: Comparison of the WaveNet and conditional encoder-decoder simulation accuracy. The left plot shows a velocity model, reflectivity series and ground truth FD simulation for a randomly selected example in the horizontally layered velocity model test set in red. Green shows the WaveNet simulation. The middle plot shows the conditional encoder-decoder simulation for the same velocity model. The right plot shows the histogram of the average absolute amplitude difference between the ground truth FD simulation and the WaveNet and conditional encoder-decoder simulations over this test set. A $t^{2.5}$ gain is applied for display.

Layer	Type	in, out channels	kernel size	stride	padding						
1	Conv2d	(1,8)	(3,3)	(1,1)	(1,1)	14	Conv2d	(512,512)	(3,3)	(1,1)	(1,1)
2	Conv2d	(8,16)	(2,2)	(2,2)	0	15	Conv2d	(512,512)	(3,3)	(1,1)	(1,1)
3	Conv2d	(16,16)	(3,3)	(1,1)	(1,1)	16	ConvT2d	(512,256)	(2,4)	(2,4)	0
4	Conv2d	(16,32)	(2,2)	(2,2)	0	17	Conv2d	(256,256)	(3,3)	(1,1)	(1,1)
5	Conv2d	(32,32)	(3,3)	(1,1)	(1,1)	18	Conv2d	(256,256)	(3,3)	(1,1)	(1,1)
6	Conv2d	(32,64)	(2,2)	(2,2)	0	19	ConvT2d	(256,64)	(2,4)	(2,4)	0
7	Conv2d	(64,128)	(2,2)	(2,2)	0	20	Conv2d	(64,64)	(3,3)	(1,1)	(1,1)
8	Conv2d	(128,256)	(2,2)	(2,2)	0	21	Conv2d	(64,64)	(3,3)	(1,1)	(1,1)
9	Conv2d	(256,512)	(2,2)	(2,2)	0	22	ConvT2d	(64,8)	(2,4)	(2,4)	0
10	Conv2d	(512,1024)	(2,2)	(2,2)	0	23	Conv2d	(8,8)	(3,3)	(1,1)	(1,1)
11	Concat	(1024,1025)				24	Conv2d	(8,8)	(3,3)	(1,1)	(1,1)
12	ConvT2d	(1025,1025)	(2,2)	(2,2)	0	25	Conv2d	(8,1)	(1,1)	(1,1)	0
13	ConvT2d	(1025,512)	(2,4)	(2,4)	0						

Table C.1: Conditional encoder-decoder layer parameters. Each entry shows the parameterisation of each convolutional layer. The padding column shows the padding on each side of the input tensor for each spatial dimension.

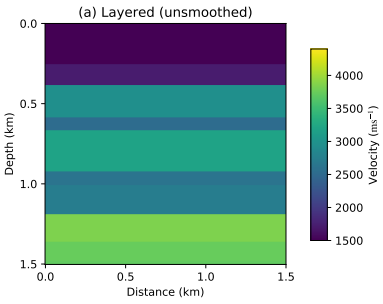


Figure C.2: Layered velocity model without smoothing, used in Figure C.3.

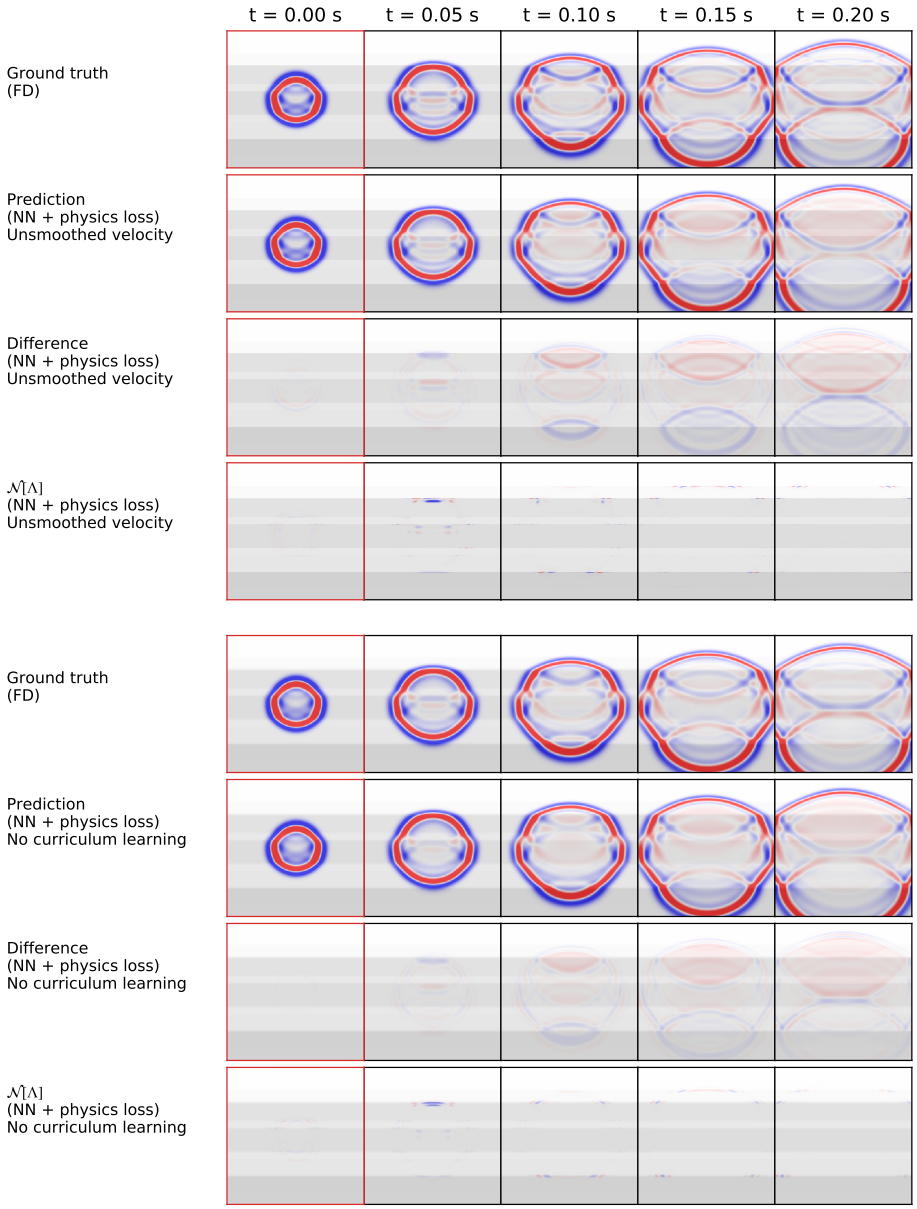


Figure C.3: Comparison of the PINN wavefield prediction to ground truth FD simulation for two additional PINN training setups. In the first test the layered velocity model is used without smoothing ("Unsmooothed velocity"), shown in Figure C.2. In the second test curriculum learning is not used; that is, the physics loss is used from the start of training and an expanding time horizon is not used (i.e. t_j values for computing the physics loss in Equation 5.5 are sampled from their full input range $[0, T_{\max}]$ from the start of training). Top four rows show the results from the first test and bottom four rows show the results from the second test. For each test, top two rows show the FD and PINN wavefield solutions through time. Third row shows the difference of the network prediction to FD simulation. Fourth row shows the residual of the underlying wave equation (from physics loss in Equation 5.5), which is close to zero for an accurate solution to the wave equation. Plots bordered in black indicate wavefield times which are outside of the boundary training data. Plots are overlain on the velocity model, shown in grey. Larger errors can be seen in both cases compared to the PINN in Figure 5.17.

D

Scalable physics-informed neural networks

D.1 Forward inference FLOPs calculation

To compare the computational resources required to train PINNs and FBPINNs, we use a measure which we call the forward inference FLOPs. This is a count of the cumulative number of FLOPs spent during forward inference of their neural network(s) during training (i.e., when computing the function $NN(x;\theta)$ for PINNs or $\overline{NN}(x;\theta)$ for FBPINNs), and therefore gives a measure of data-efficiency.

For a fully connected neural network with tanh activations and a linear output layer, as used here, we assume the number of flops F spent during forward inference of the network is

$$F = N((2d+6)h + (l-1)(2h+6)h + (2h+1)d_u) \ , \tag{D.1}$$

where N is the number of training points input to the network (or the batch size), d is the dimensionality of the input vector, d_u is the dimensionality of the output vector, h is the number of hidden units and l is the number of hidden layers. To see this, we note that the computation of each network layer consists of a matrix-matrix multiply, addition of the bias vector, and application of the activation function. For the first layer of the network, the matrix-matrix multiply requires $2Ndh$ operations, the bias addition requires Nh operations and we assume

the tanh operation requires $5Nh$ operations. Similar calculations follow for the remaining layers and hence Equation D.1 follows.

For PINNs, we use Equation D.1 directly to estimate the forward inference FLOPs. For FBPINNs, the forward inference FLOPs are calculated using

$$F = \sum_{i}^{n} F_i \ , \tag{D.2}$$

where the sum is over all subdomain networks used in the current training step.

We note that this measure only counts FLOPs spent during the forward inference of the networks, and does not count the additional FLOPs spent during gradient computation, backpropagation or any other part of the training algorithm.

D.1.1 Scaling of FBPINNs with network size / number of subdomains

A notable aspect of FBPINNs is that the total number of forward inference FLOPs required during training only depends on the subdomain network size, and not on the number of subdomains. We note that only training points within each subdomain are needed to train each subdomain network. We also note that, for a fixed domain and fixed training point density, as the number of subdomains increases the average number of training points falling over each subdomain decreases at the same rate. Therefore, using Equation D.2 and assuming the same network size per subdomain, it can be shown that the total number of forward FLOPs stays constant. However, this also assumes that the proportion of the domain covered by overlapping regions stays constant; the forward inference FLOPs will increase if this increases, as in these regions the outputs of all overlapping networks need to be computed.

D.2 Finite difference modelling for (2+1)D wave equation

When studying the (2+1)D wave equation (Section 6.5.5) we use finite difference modelling as the ground truth solution. The SEISMIC_CPML library [Komatitsch and Martin, 2007] is used (specifically, the seismic_CPML_2D_pressure_second_order

code) which performs staggered-grid second order finite difference modelling of the time-dependent 2D acoustic wave equation using a convolutional perfectly matched layer (PML) boundary condition at the edges of the domain. For ease of use, we re-implemented the original Fortran code in Python. The simulation is initialised by sampling the spatially varying wave speed, initial wavefield and initial wavefield derivative as defined in Section 6.5.5 on a regular $694 \times 694 \times 1891$ grid ($x_1 \times x_2 \times t$). A high density of grid points ($\sim 5\times$ spatial Nyquist frequency) is used so that the simulation is high-fidelity. An additional 10 grid points are used to define the PML boundary, which are cropped from the final solution before comparing it to the solution from the PINN and FBPINN.

E
Conclusions

E.1 Training dataset sizes

Table E.1 compares the size of the datasets used to train the models presented in this thesis.

Ref	Model	# Examples	Shape	Total # measurements
3.3	VAE	1,985,693	(120,)	238,283,160
4.3	DestripeNet (summed mode, left camera)	9,111	(1024, 2532)	23,622,709,248
4.3	PhotonNet (summed mode, left camera)	1,600,824	(256, 256)	104,911,601,664
5.3	WaveNet	40,000	(500, 11)	220,000,000
5.4	Conditional encoder-decoder	240,000	(500, 32)	3,840,000,000
5.5	PINN (Marmousi model)	100	(300, 300, 20)	180,000,000
6.4	FBPINN	0	()	0

Table E.1: Size of the datasets used to train the models presented in this thesis. For each model, we show the total number of labelled training examples, the shape of each example and the corresponding total number of measurements (the product of these two quantities). Note that whilst FBPINNs do not require any labelled training data, many random input coordinates in the domain are required during training.

References

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

William J. Dally, Stephen W. Keckler, and David B. Kirk. Evolution of the Graphics Processing Unit (GPU). *IEEE Micro*, 41(6):42–51, 2021. ISSN 19374143. doi: 10.1109/MM.2021.3113475.

Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and Decision-Making for Autonomous Vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:187–210, may 2018. ISSN 25735144. doi: 10.1146/annurev-control-060117-105157. URL `https://www.annualreviews.org/doi/abs/10.1146/annurev-control-060117-105157`.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362 (6419):1140–1144, 2018. ISSN 10959203. doi: 10.1126/science.aar6404.

Nathan Baker, Frank Alexander, Timo Bremer, Aric Hagberg, Yannis Kevrekidis, Habib Najm, Manish Parashar, Abani Patra, James Sethian, Stefan Wild, Karen Willcox, and Steven Lee. Workshop Report on Basic Research Needs for Scientific Machine Learning: Core Technologies for Artificial Intelligence. Technical report, USDOE Office of Science (SC) (United States), feb 2019. URL `http://www.osti.gov/servlets/purl/1478744/`.

Tony Hey, Keith Butler, Sam Jackson, and Jeyarajan Thiyagalingam. Machine learning and big scientific data. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 378(2166), mar 2020. ISSN 1364503X. doi: 10.1098/rsta.2019.0054. URL `https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2019.0054`.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A.A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873): 583–589, jul 2021. ISSN 14764687. doi: 10.1038/s41586-021-03819-2. URL `https://www.nature.com/articles/s41586-021-03819-2`.

Gopala K. Anumanchipalli, Josh Chartier, and Edward F. Chang. Speech synthesis from neural decoding of spoken sentences. *Nature*, 568(7753):493–498, apr 2019. ISSN 14764687. doi: 10.1038/s41586-019-1119-1.

Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, feb 2017. ISSN 10959203. doi: 10.1126/science.aag2302.

Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *Proceedings - 2018 IEEE 5th International Conference on Data Science and Advanced Analytics, DSAA 2018*, pages 80–89. Institute of Electrical and Electronics Engineers Inc., jan 2019. ISBN 9781538650905. doi: 10.1109/DSAA.2018.00018.

Davide Castelvecchi. Can we open the black box of AI? *Nature*, 538(7623):20–23, oct 2016. ISSN 14764687. doi: 10.1038/538020a.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20, 2019. ISSN 15337928. URL `https://jmlr.org/papers/v20/18-598.html`.

Timothy M. Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. Meta-Learning in Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 01(01):1–1, may 2021. ISSN 19393539. doi: 10.1109/TPAMI.2021.3079209.

Yoshua Bengio, Yann Lecun, and Geoffrey Hinton. Deep learning for AI. *Communications of the ACM*, 64(7):58–65, jun 2021. ISSN 15577317. doi: 10.1145/3448250. URL `https://dl.acm.org/doi/abs/10.1145/3448250`.

Abbas Ourmazd. Science in the age of machine learning. *Nature Reviews Physics*, 2(7): 342–343, may 2020. ISSN 25225820. doi: 10.1038/s42254-020-0191-7. URL `https://www.nature.com/articles/s42254-020-0191-7`.

Jessica Zosa Forde and Michela Paganini. The Scientific Method in the Science of Machine Learning. *arXiv*, apr 2019. URL `https://arxiv.org/abs/1904.10922v1http://arxiv.org/abs/1904.10922`.

George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, pages 1–19, may 2021. doi: 10.1038/s42254-021-00314-5. URL `www.nature.com/natrevphys`.

Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems. *arXiv*, 1:1, mar 2020. URL `https://arxiv.org/abs/2003.04919v5http://arxiv.org/abs/2003.04919`.

Salvatore Cuomo, Vincenzo Schiano di Cola, Fabio Giampaolo, Gianluigi Rozza, Maizar Raissi, and Francesco Piccialli. Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What's next. *arXiv*, jan 2022. URL `https://arxiv.org/abs/2201.05624v2http://arxiv.org/abs/2201.05624`.

Simon Arridge, Peter Maass, Ozan Öktem, and Carola-Bibiane Schönlieb. Solving inverse problems using data-driven models. *Acta Numerica*, 28:1–174, 2019. ISSN 0962-4929. doi: 10.1017/s0962492919000059.

Anuj Karpatne, Gowtham Atluri, James H. Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vipin Kumar. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on Knowledge and Data Engineering*, 29(10):2318–2331, oct 2017a. ISSN 10414347. doi: 10.1109/TKDE.2017.2720168.

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. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 10902716. doi: 10.1016/j.jcp.2018.10.045. URL `https://doi.org/10.1016/j.jcp.2018.10.045`.

J. Nathan Kutz and Steven L. Brunton. Parsimony as the ultimate regularizer for physics-informed machine learning. *Nonlinear Dynamics*, 107(3):1801–1817, jan 2022. ISSN 1573269X. doi: 10.1007/s11071-021-07118-3. URL `https://link.springer.com/article/10.1007/s11071-021-07118-3`.

Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, and Ali Ramadhan. Universal differential equations for scientific machine learning. *arXiv*, jan 2020. URL `http://arxiv.org/abs/2001.04385`.

Nils Thuerey, Philipp Holl, Maximilian Mueller, Patrick Schnell, Felix Trost, and Kiwon Um. *Physics-based Deep Learning*. WWW, sep 2021. URL `https://arxiv.org/abs/2109.05237v2http://arxiv.org/abs/2109.05237`.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. *arXiv*, 2016. ISSN 0899-7667. doi: 10.1109/ICASSP.2009.4960364. URL `http://arxiv.org/abs/1609.03499`.

Isaac Elias Lagaris, Aristidis Likas, and Dimitrios I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998. ISSN 10459227. doi: 10.1109/72.712178.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural Ordinary Differential Equations. *NIPS*, 109(NeurIPS):31–60, jun 2018a. ISSN 20385757. doi: 10.48550/arxiv.1806.07366. URL `https://arxiv.org/abs/1806.07366v5`.

NeurIPS. The Symbiosis of Deep Learning and Differential Equations (DLDE) Workshop. 2021a. URL `https://dl-de.github.io/`.

NeurIPS. Machine Learning and the Physical Sciences Worshop. 2021b. URL `https://ml4physicalsciences.github.io/2021/`.

ICLR. Workshop on Integration of Deep Neural Models and Differential Equations. 2020. URL `http://iclr2020deepdiffeq.rice.edu/https://openreview.net/group?id=ICLR.cc/2020/Workshop/DeepDiffEq`.

AAAI. Symposium on Combining Artificial Intelligence and Machine Learning with Physics Sciences. 2021. URL https://sites.google.com/view/aaai-mlps.

AAAI. Symposium on Physics-Guided AI to Accelerate Scientific Discovery. 2020. URL https://sites.google.com/vt.edu/pgai-aaai-20.

ICERM. Workshop on Scientific Machine Learning. 2019. URL https://icerm.brown.edu/events/ht19-1-sml/.

NSF. NSF National Artificial Intelligence Research Institutes, 2020. URL https://www.nsf.gov/cise/ai.jsp.

Stephan Rasp, Michael S. Pritchard, and Pierre Gentine. Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences of the United States of America*, 115(39):9684–9689, sep 2018. ISSN 10916490. doi: 10.1073/pnas.1810286115.

Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 426:109951, feb 2021. ISSN 10902716. doi: 10.1016/j.jcp.2020.109951.

Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and E. Weinan. Deep Potential Molecular Dynamics: A Scalable Model with the Accuracy of Quantum Mechanics. *Physical Review Letters*, 120(14):143001, apr 2018. ISSN 10797114. doi: 10.1103/PhysRevLett.120.143001. URL https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.120.143001.

Christopher Irrgang, Niklas Boers, Maike Sonnewald, Elizabeth A. Barnes, Christopher Kadow, Joanna Staneva, and Jan Saynisch-Wagner. Towards neural Earth system modelling by integrating artificial intelligence in Earth system science. *Nature Machine Intelligence*, 3(8):667–674, aug 2021. ISSN 25225839. doi: 10.1038/s42256-021-00374-3. URL https://www.nature.com/articles/s42256-021-00374-3.

Alexander Lavin, Hector Zenil, Brooks Paige, David Krakauer, Justin Gottschlich, Tim Mattson, Anima Anandkumar, Sanjay Choudry, Kamil Rocki, Atılım Güneş Baydin, Carina Prunkl, Brooks Paige, Olexandr Isayev, Erik Peterson, Peter L. McMahon, Jakob Macke, Kyle Cranmer, Jiaxin Zhang, Haruko Wainwright, Adi Hanuka, Manuela Veloso, Samuel Assefa, Stephan Zheng, and Avi Pfeffer. Simulation Intelligence: Towards a New Generation of Scientific Methods. *arXiv*, dec 2021. URL https://arxiv.org/abs/2112.03235v1http://arxiv.org/abs/2112.03235.

Rahul Rai and Chandan K. Sahu. Driven by Data or Derived through Physics? A Review of Hybrid Physics Guided Machine Learning Techniques with Cyber-Physical System (CPS) Focus. *IEEE Access*, 8:71050–71073, 2020. ISSN 21693536. doi: 10.1109/ACCESS.2020.2987324.

Mark Alber, Adrian Buganza Tepole, William R Cannon, Suvranu De, Salvador Dura-Bernal, Krishna Garikipati, George Karniadakis, William W Lytton, Paris Perdikaris, Linda Petzold, and Ellen Kuhl. Integrating machine learning and multiscale modeling—perspectives, challenges, and opportunities in the biological, biomedical, and

behavioral sciences. *npj Digital Medicine*, 2(1):115, 2019. ISSN 2398-6352. doi: 10.1038/s41746-019-0193-y. URL https://doi.org/10.1038/s41746-019-0193-y.

Arka Daw, R. Quinn Thomas, Cayelan C. Carey, Jordan S. Read, Alison P. Appling, and Anuj Karpatne. Physics-guided architecture (pga) of neural networks for quantifying uncertainty in lake temperature modeling. In *Proceedings of the 2020 SIAM International Conference on Data Mining, SDM 2020*, pages 532–540. Society for Industrial and Applied Mathematics Publications, nov 2020. ISBN 9781611976236. doi: 10.1137/1.9781611976236.60. URL http://arxiv.org/abs/1911.02682.

Julia Ling, Andrew Kurzawski, and Jeremy Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, nov 2016. ISSN 14697645. doi: 10.1017/jfm.2016.615. URL https://doi.org/10.1017/jfm.2016.615.

Rui Wang, Robin Walters, and Rose Yu. Incorporating symmetry into deep dynamics models for improved generalization. In *ICLR 2021*. arXiv, feb 2021a. URL http://arxiv.org/abs/2002.03061.

Brandon Anderson, Truong Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. In *Advances in Neural Information Processing Systems*, volume 32. Neural information processing systems foundation, jun 2019. URL https://arxiv.org/abs/1906.04015v3.

Kristof T. Schütt, P. J. Kindermans, Huziel E. Sauceda, Stefan Chmiela, Alexandre Tkatchenko, and K. R. Müller. SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 992–1002. Neural information processing systems foundation, jun 2017. URL http://arxiv.org/abs/1706.08566.

Silviu Marian Udrescu, Andrew Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark. AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. In *Advances in Neural Information Processing Systems*, volume 2020-Decem. Neural information processing systems foundation, jun 2020. URL https://arxiv.org/abs/2006.10782v2.

Yunhao Ba, Guangyuan Zhao, and Achuta Kadambi. Blending diverse physical priors with neural networks. *arXiv*, oct 2019. ISSN 23318422. URL http://arxiv.org/abs/1910.00201.

Maysum Panju and Ali Ghodsi. A Neuro-Symbolic Method for Solving Differential and Functional Equations. *arXiv*, nov 2020. URL http://arxiv.org/abs/2011.02415.

Nicholas Geneva and Nicholas Zabaras. Transformers for modeling physical systems. *Neural Networks*, 146:272–289, feb 2022. ISSN 18792782. doi: 10.1016/j.neunet.2021.11.022.

Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):1–10, nov 2018. ISSN 20411723. doi: 10.1038/s41467-018-07210-0. URL https://www.nature.com/articles/s41467-018-07210-0.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Machine learning of linear differential equations using Gaussian processes. *Journal of Computational Physics*, 348: 683–693, nov 2017. ISSN 10902716. doi: 10.1016/j.jcp.2017.07.050.

Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, mar 2018. ISSN 10902716. doi: 10.1016/j.jcp.2017.11.039.

Arvind T. Mohan, Nicholas Lubbers, Daniel Livescu, and Michael Chertkov. Embedding hard physical constraints in neural network coarse-graining of 3D turbulence. *arXiv*, jan 2020. ISSN 23318422. URL http://arxiv.org/abs/2002.00021.

Anuj Karpatne, William Watkins, Jordan Read, and Vipin Kumar. Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling. *arXiv*, 2017b. URL http://arxiv.org/abs/1710.11431.

N. Benjamin Erichson, Michael Muehlebach, and Michael W. Mahoney. Physics-informed autoencoders for lyapunov-stable fluid flow prediction. *arXiv*, may 2019. ISSN 23318422. URL http://arxiv.org/abs/1905.10866.

You Xie, Erik Franz, Mengyu Chu, and Nils Thuerey. TempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow. *ACM Transactions on Graphics*, 37(4): 15, jul 2018. ISSN 15577368. doi: 10.1145/3197517.3201304. URL https://dl.acm.org/doi/abs/10.1145/3197517.3201304.

Johann Brehmer, Gilles Louppe, Juan Pavez, and Kyle Cranmer. Mining gold from implicit models to improve likelihood-free inference. *Proceedings of the National Academy of Sciences of the United States of America*, 117(10):5242–5249, mar 2020. ISSN 10916490. doi: 10.1073/pnas.1915980117. URL www.pnas.org/cgi/doi/10.1073/pnas.1915980117.

Tom Beucler, Michael Pritchard, Stephan Rasp, Jordan Ott, Pierre Baldi, and Pierre Gentine. Enforcing Analytic Constraints in Neural Networks Emulating Physical Systems. *Physical Review Letters*, 126(9):098302, mar 2021. ISSN 10797114. doi: 10.1103/PhysRevLett.126.098302. URL https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.126.098302.

Yang Zeng, Jin Long Wu, and Heng Xiao. Enforcing imprecise constraints on generative adversarial networks for emulating physical systems. *Communications in Computational Physics*, 30(3):635–665, 2021. ISSN 19917120. doi: 10.4208/CICP.OA-2020-0106.

Sam Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, volume 32. Neural information processing systems foundation, jun 2019. URL https://arxiv.org/abs/1906.01563v3.

Peter Toth, Danilo Jimenez Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian Generative Networks. *ICLR 2020*, sep 2020. doi: 10.48550/arxiv.1909.13789. URL https://arxiv.org/abs/1909.13789v2http://arxiv.org/abs/1909.13789.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian Neural Networks. *ICLR 2020 Workshop DeepDiffEq*, mar 2020. ISSN 2331-8422. URL https://arxiv.org/abs/2003.04630v2http://arxiv.org/abs/2003.04630.

Luke de Oliveira, Michela Paganini, and Benjamin Nachman. Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis. *Computing and Software for Big Science*, 1(1):1–24, nov 2017. ISSN 25102044. doi: 10.1007/s41781-017-0004-6. URL https://link.springer.com/article/10.1007/s41781-017-0004-6.

Ehsan Kharazmi, Zhongqiang Zhang, and George E.M. Karniadakis. VPINNs: Variational physics-informed neural networks for solving partial differential equations. *arXiv*, nov 2019. URL http://arxiv.org/abs/1912.00873.

Liu Yang, Xuhui Meng, and George Em Karniadakis. B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. *Journal of Computational Physics*, 425:109913, jan 2021a. ISSN 10902716. doi: 10.1016/j.jcp.2020.109913.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, jan 2022. ISSN 10902716. doi: 10.1016/j.jcp.2021.110768.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Science Advances*, 7(40):8605–8634, oct 2021b. ISSN 23752548. doi: 10.1126/sciadv.abi8605. URL https://www.science.org/doi/full/10.1126/sciadv.abi8605.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-Informed Neural Operator for Learning Partial Differential Equations. *arXiv*, nov 2021. URL https://arxiv.org/abs/2111.03794v1http://arxiv.org/abs/2111.03794.

Yinhao Zhu, Nicholas Zabaras, Phaedon Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394: 56–81, jan 2019. ISSN 10902716. doi: 10.1016/j.jcp.2019.05.024. URL http://arxiv.org/abs/1901.06314http://dx.doi.org/10.1016/j.jcp.2019.05.024.

Nicholas Geneva and Nicholas Zabaras. Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403:109056, feb 2020. ISSN 10902716. doi: 10.1016/j.jcp.2019.109056.

Han Gao, Luning Sun, and Jian Xun Wang. PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. *Journal of Computational Physics*, 428:110079, mar 2021. ISSN 10902716. doi: 10.1016/j.jcp.2020.110079.

Yuyao Chen, Lu Lu, George Em Karniadakis, and Luca Dal Negro. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Optics Express*, 28(8):11618, apr 2020. ISSN 1094-4087. doi: 10.1364/oe.384875.

Zhao Chen, Yang Liu, and Hao Sun. Physics-informed learning of governing equations from scarce data. *Nature Communications*, 12(1):1–13, oct 2021. ISSN 20411723. doi: 10.1038/s41467-021-26434-1. URL https://www.nature.com/articles/s41467-021-26434-1.

Kathleen Champion, Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences of the United States of America*, 116(45):22445–22451, nov 2019. ISSN 10916490. doi: 10.1073/pnas.1906995116. URL https://www.pnas.org/content/116/45/22445https://www.pnas.org/content/116/45/22445.abstract.

Suraj Pawar, Omer San, Burak Aksoylu, Adil Rasheed, and Trond Kvamsdal. Physics guided machine learning using simplified theories. *Physics of Fluids*, 33(1):011701, jan 2021. ISSN 10897666. doi: 10.1063/5.0038929. URL https://aip.scitation.org/doi/abs/10.1063/5.0038929.

Chiyu LMaxr Jiang, Soheil Esmaeilzadeh, Kamyar Azizzadenesheli, Karthik Kashinath, Mustafa Mustafa, Hamdi A. Tchelepi, Philip Marcus, Mr Prabhat, and Anima Anandkumar. Meshfreeflownet: A physics-constrained deep continuous space-time super-resolution framework. In *International Conference for High Performance Computing, Networking, Storage and Analysis, SC*, volume 2020-Novem. IEEE Computer Society, nov 2020. ISBN 9781728199986. doi: 10.1109/SC41405.2020.00013.

Yuxiao Ren, Xinji Xu, Senlin Yang, Lichao Nie, and Yangkang Chen. A Physics-Based Neural-Network Way to Perform Seismic Full Waveform Inversion. *IEEE Access*, 8: 112266–112277, may 2020. ISSN 21693536. doi: 10.1109/access.2020.2997921.

Momchil Minkov, Ian A.D. Williamson, Lucio C. Andreani, Dario Gerace, Beicheng Lou, Alex Y. Song, Tyler W. Hughes, and Shanhui Fan. Inverse Design of Photonic Crystals through Automatic Differentiation. *ACS Photonics*, 7(7):1729–1741, jul 2020. ISSN 23304022. doi: 10.1021/acsphotonics.0c00327. URL https://pubs.acs.org/doi/abs/10.1021/acsphotonics.0c00327.

Tobias Würfl, Mathis Hoffmann, Vincent Christlein, Katharina Breininger, Yixin Huang, Mathias Unberath, and Andreas K. Maier. Deep Learning Computed Tomography: Learning Projection-Domain Weights From Image Domain in Limited Angle Problems. *IEEE Transactions on Medical Imaging*, 37(6):1454–1463, 2018. ISSN 1558254X. doi: 10.1109/TMI.2018.2833499.

Liang Zhang, Gang Wang, and Georgios B. Giannakis. Real-Time Power System State Estimation and Forecasting via Deep Unrolled Neural Networks. *IEEE Transactions on Signal Processing*, 67(15):4069–4077, aug 2019a. ISSN 19410476. doi: 10.1109/TSP.2019.2926023.

Zichao Long, Yiping Lu, and Bin Dong. PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics*, 399, nov 2019. ISSN 10902716. doi: 10.1016/j.jcp.2019.108925. URL http://arxiv.org/abs/1812.04426http://dx.doi.org/10.1016/j.jcp.2019.108925.

Kiwon Um, Robert Brand, Yun Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative PDE-solvers. In *Advances in Neural Information Processing Systems*, volume 2020-Decem. Neural information processing systems foundation, jun 2020. doi: 10.48550/arxiv.2007.00016. URL https://arxiv.org/abs/2007.00016v2.

Jonas Adler and Ozan Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):124007, nov 2017. ISSN 13616420. doi: 10.1088/1361-6420/aa9581. URL https://iopscience.iop.org/article/10.1088/1361-6420/aa9581https://iopscience.iop.org/article/10.1088/1361-6420/aa9581/meta.

Warren R. Morningstar, Laurence Perreault Levasseur, Yashar D. Hezaveh, Roger Blandford, Phil Marshall, Patrick Putzky, Thomas D. Rueter, Risa Wechsler, and Max Welling. Data-driven Reconstruction of Gravitationally Lensed Galaxies Using Recurrent Inference Machines. *The Astrophysical Journal*, 883(1):14, 2019. ISSN 1538-4357. doi: 10.3847/1538-4357/ab35d7.

Kerstin Hammernik, Teresa Klatzer, Erich Kobler, Michael P. Recht, Daniel K. Sodickson, Thomas Pock, and Florian Knoll. Learning a variational network for reconstruction of accelerated MRI data. *Magnetic Resonance in Medicine*, 79(6):3055–3071, 2018. ISSN 15222594. doi: 10.1002/mrm.26977.

Housen Li, Johannes Schwab, Stephan Antholzer, and Markus Haltmeier. NETT: Solving inverse problems with deep neural networks. *Inverse Problems*, 36(6):065005, 2020a. ISSN 13616420. doi: 10.1088/1361-6420/ab6d57. URL https://doi.org/10.1088/1361-6420/ab6d57.

Sebastian Lunz, Ozan Öktem, and Carola Bibiane Schönlieb. Adversarial regularizers in inverse problems. *Advances in Neural Information Processing Systems*, 2018-Decem (NeurIPS):8507–8516, 2018. ISSN 10495258.

Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G. Dimakis. Compressed sensing using generative models. *34th International Conference on Machine Learning, ICML 2017*, 2: 822–841, 2017.

L. Mosser, O. Dubrule, and M. Blunt. Stochastic seismic waveform inversion using generative adversarial networks as a geological prior. *1st EAGE/PESGB Workshop on Machine Learning*, 2018. doi: 10.3997/2214-4609.201803018.

D. J. Gross. The role of symmetry in fundamental physics. *Proceedings of the National Academy of Sciences of the United States of America*, 93(25):14256–14259, dec 1996. ISSN 00278424. doi: 10.1073/pnas.93.25.14256. URL https://www.pnas.org/content/93/25/14256https://www.pnas.org/content/93/25/14256.abstract.

Silviu Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, apr 2020. ISSN 23752548. doi: 10.1126/sciadv.aay2631. URL http://advances.sciencemag.org/.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv*, jun 2018. ISSN 23318422. URL http://arxiv.org/abs/1806.09055.

Steven L. Brunton, Marko Budišić, Eurika Kaiser, and J. Nathan Kutz. Modern Koopman Theory for Dynamical Systems. *arXiv*, feb 2021. URL https://arxiv.org/abs/2102.12086v2http://arxiv.org/abs/2102.12086.

B. O. Koopman. Hamiltonian Systems and Transformation in Hilbert Space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, may 1931. ISSN 0027-8424. doi: 10.1073/pnas.17.5.315. URL https://www.pnas.org.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv*, jun 2017. ISSN 0140-525X. doi: 10.1017/S0140525X16001837. URL http://arxiv.org/abs/1706.03762.

Laura Swiler, Mamikon Gulian, Ari Frankel, Cosmin Safta, and John Jakeman. A Survey of Constrained Gaussian Process Regression: Approaches and Implementation Challenges. *arXiv*, 1(2):119–156, jun 2020. ISSN 23318422. doi: 10.1615/JMachLearnModelComput.2020035155. URL http://arxiv.org/abs/2006.09319http://dx.doi.org/10.1615/JMachLearnModelComput.2020035155.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 3, pages 2672–2680. Neural information processing systems foundation, 2014. doi: 10.3156/jsoft.29.5_177_2.

Michael Crawshaw. Multi-Task Learning with Deep Neural Networks: A Survey. *arXiv*, sep 2020. doi: 10.48550/arxiv.2009.09796. URL https://arxiv.org/abs/2009.09796v1http://arxiv.org/abs/2009.09796.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *34th International Conference on Machine Learning, ICML 2017*, volume 6, pages 4043–4055, 2017. ISBN 9781510855144. doi: 10.5555/3305890.3305954. URL https://github.com/openai/improved-gan/.

G. Pang, M. D'Elia, M. Parks, and G. E. Karniadakis. nPINNs: Nonlocal physics-informed neural networks for a parametrized nonlocal universal Laplacian operator. Algorithms and applications. *Journal of Computational Physics*, 422:109760, dec 2020. ISSN 10902716. doi: 10.1016/j.jcp.2020.109760.

Guofei Pang, L. U. Lu, and George E.M. Karniadakis. FPinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4): A2603–A2626, aug 2019. ISSN 10957197. doi: 10.1137/18M1229845. URL https://doi.org/10.1137/18M1229845.

Ehsan Kharazmi, Zhongqiang Zhang, and George Em Karniadakis. hp-VPINNs: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374, mar 2021. ISSN 00457825. doi: 10.1016/j.cma.2020.113547. URL http://arxiv.org/abs/2003.05385http://dx.doi.org/10.1016/j.cma.2020.113547.

Siddhartha Mishra and Roberto Molinaro. Estimates on the generalization error of physics-informed neural networks for approximating PDEs. *IMA Journal of Numerical Analysis*, 00:1–43, jan 2022. ISSN 0272-4979. doi: 10.1093/imanum/drab093. URL `https://academic.oup.com/imajna/advance-article/doi/10.1093/imanum/drab093/6503953`.

Yeonjong Shin, J´er^ome Darbon, and George Em Karniadakis. On the Convergence of Physics Informed Neural Networks for Linear Second-Order Elliptic and Parabolic Type PDEs. *Communications in Computational Physics*, 28(5):2042–2074, apr 2020. ISSN 1815-2406. doi: 10.4208/cicp.oa-2020-0193. URL `http://arxiv.org/abs/2004.01806`.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):3055–3081, sep 2021c. ISSN 10957197. doi: 10.1137/20M1318043. URL `https://epubs.siam.org/doi/abs/10.1137/20M1318043`.

Cosmin Anitescu, Elena Atroshchenko, Naif Alajlan, and Timon Rabczuk. Artificial neural network methods for the solution of second order boundary value problems. *Computers, Materials and Continua*, 59(1):345–359, jan 2019. ISSN 15462226. doi: 10.32604/cmc.2019.06641.

Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *arXiv*, jul 2019. URL `http://arxiv.org/abs/1907.04502`.

Ameya D. Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404, jun 2020a. ISSN 10902716. doi: 10.1016/j.jcp.2019.109136. URL `http://arxiv.org/abs/1906.01170http://dx.doi.org/10.1016/j.jcp.2019.109136`.

Ameya D. Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 476(2239):20200334, jul 2020b. ISSN 1364-5021. doi: 10.1098/rspa.2020.0334. URL `https://royalsocietypublishing.org/doi/10.1098/rspa.2020.0334`.

V. I. Avrutskiy. Enhancing Function Approximation Abilities of Neural Networks by Training Derivatives. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–9, apr 2020. ISSN 2162-237X. doi: 10.1109/tnnls.2020.2979706.

Hwijae Son, Jin Woo Jang, Woo Jin Han, and Hyung Ju Hwang. Sobolev Training for the Neural Network Solutions of PDEs. *arXiv*, jan 2021. URL `http://arxiv.org/abs/2101.08932`.

Liyao Lyu, Zhen Zhang, Minxin Chen, and Jingrun Chen. MIM: A deep mixed residual method for solving high-order partial differential equations. *Journal of Computational Physics*, 452:110930, mar 2022. ISSN 10902716. doi: 10.1016/j.jcp.2021.110930.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation

theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, mar 2021. ISSN 25225839. doi: 10.1038/s42256-021-00302-5. URL https://doi.org/10.1038/s42256-021-00302-5.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. *ICLR 2021*, oct 2020b. URL http://arxiv.org/abs/2010.08895.

Steven L. Brunton, Joshua L. Proctor, J. Nathan Kutz, and William Bialek. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences of the United States of America*, 113 (15):3932–3937, apr 2016. ISSN 10916490. doi: 10.1073/pnas.1517384113. URL https://www.pnas.org/content/113/15/3932https://www.pnas.org/content/113/15/3932.abstract.

Bharath Ramsundar, Dilip Krishnamurthy, and Venkatasubramanian Viswanathan. Differentiable Physics: A Position Piece. *arXiv*, sep 2021. doi: 10.48550/arxiv.2109.07573. URL https://arxiv.org/abs/2109.07573v1http://arxiv.org/abs/2109.07573.

Filipe de A. Belbute-Peres, Kelsey R. Allen, Kevin A. Smith, Joshua B. Tenenbaum, and J. Zico Kolter. End-to-end differentiable physics for learning and control. In *Advances in Neural Information Processing Systems*, volume 2018-Decem, pages 7178–7189, 2018. URL http://www.mit.edu/{~}k2smith/publication/differentiable{_}phys/.

Vishal Monga, Yuelong Li, and Yonina C. Eldar. Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing. *IEEE Signal Processing Magazine*, 38(2):18–44, mar 2021. ISSN 15580792. doi: 10.1109/MSP.2020.3016905.

Atilim Günes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *The Journal of Machine Learning Research*, 18(1):5595–5637, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H Wallach, H Larochelle, A Beygelzimer, F d\textquotesingle Alché-Buc, E Fox, and R Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR 2015*, dec 2015. URL https://arxiv.org/abs/1412.6980http://arxiv.org/abs/1412.6980.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on*

Computer Vision and Pattern Recognition, volume 2016-Decem, pages 770–778. IEEE Computer Society, dec 2016. ISBN 9781467388504. doi: 10.1109/CVPR.2016.90. URL http://image-net.org/challenges/LSVRC/2015/.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, pages 399–406, 2010.

Marcin Andrychowicz, Misha Denil, Sergio Gómez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in Neural Information Processing Systems*, (Nips):3988–3996, 2016. ISSN 10495258.

Lars Ruthotto and Eldad Haber. Deep Neural Networks Motivated by Partial Differential Equations. *Journal of Mathematical Imaging and Vision*, 62(3):352–364, apr 2020. ISSN 15737683. doi: 10.1007/s10851-019-00903-1. URL http://arxiv.org/abs/1804.04272.

Pratik Chaudhari, Adam Oberman, Stanley Osher, Stefano Soatto, and Guillaume Carlier. Deep relaxation: Partial differential equations for optimizing deep neural networks. *Research in Mathematical Sciences*, 5(3):1–30, sep 2018. ISSN 21979847. doi: 10.1007/s40687-018-0148-y. URL https://doi.org/10.1007/s40687-018-0148-y.

Bo Chang, Ed H. Chi, Minmin Chen, and Eldad Haber. AntisymmetricRNN: A dynamical system view on recurrent neural networks. In *7th International Conference on Learning Representations, ICLR 2019*. International Conference on Learning Representations, ICLR, feb 2019. URL https://arxiv.org/abs/1902.09689v1.

Logan G. Wright, Tatsuhiro Onodera, Martin M. Stein, Tianyu Wang, Darren T. Schachter, Zoey Hu, and Peter L. McMahon. Deep physical neural networks trained with backpropagation. *Nature*, 601(7894):549–555, jan 2022. ISSN 0028-0836. doi: 10.1038/s41586-021-04223-6. URL https://www.nature.com/articles/s41586-021-04223-6.

Tyler W. Hughes, Ian A.D. Williamson, Momchil Minkov, and Shanhui Fan. Wave physics as an analog recurrent neural network. *Science Advances*, 5(12), dec 2019. ISSN 23752548. doi: 10.1126/sciadv.aay6946. URL https://www.science.org/doi/abs/10.1126/sciadv.aay6946.

Spudis and Paul D. *The once and future moon*. Washington : Smithsonian Institution Press, 1996. URL https://ui.adsabs.harvard.edu/abs/1996ofm..book.....S/abstract.

Sarah A. Fagents, M. Elise Rumpf, Ian A. Crawford, and Katherine H. Joy. Preservation potential of implanted solar wind volatiles in lunar paleoregolith deposits buried by lava flows. *Icarus*, 207(2):595–604, jun 2010. ISSN 00191035. doi: 10.1016/j.icarus.2009.11.033.

Katherine H. Joy, Ian A. Crawford, Natalie M. Curran, Michael Zolensky, Amy F. Fagan, and David A. Kring. The Moon: An Archive of Small Body Migration in the Solar System. *Earth, Moon and Planets*, 118(2-3):133–158, oct 2016. ISSN 15730794. doi:

10.1007/s11038-016-9495-0. URL
https://link.springer.com/article/10.1007/s11038-016-9495-0.

E M Shoemaker. Interpretation of lunar craters. In *Physics and Astronomy of the Moon*, pages 283–359. 1962.

Raph B. Baldwin. Lunar crater counts. *The Astronomical Journal*, 69(5):377, jun 1964. ISSN 00046256. doi: 10.1086/109289. URL
https://ui.adsabs.harvard.edu/abs/1964AJ.....69..377B/abstract.

William K. Hartmann. Terrestrial and lunar flux of large meteorites in the last two billion years. *Icarus*, 4(2):157–165, may 1965. ISSN 10902643. doi: 10.1016/0019-1035(65)90057-6.

Ashwin R. Vasavada, Joshua L. Bandfield, Benjamin T. Greenhagen, Paul O. Hayne, Matthew A. Siegler, Jean Pierre Williams, and David A. Paige. Lunar equatorial surface temperatures and regolith properties from the Diviner Lunar Radiometer Experiment. *Journal of Geophysical Research E: Planets*, 117(4):0–18, dec 2012. ISSN 01480227. doi: 10.1029/2011JE003987. URL
https://onlinelibrary.wiley.com/doi/full/10.1029/2011JE003987https://onlinelibrary.wiley.com/doi/abs/10.1029/2011JE003987https://agupubs.onlinelibrary.wiley.com/doi/10.1029/2011JE003987.

J. P. Williams, D. A. Paige, B. T. Greenhagen, and E. Sefton-Nash. The global surface temperatures of the moon as measured by the diviner lunar radiometer experiment. *Icarus*, 283:300–325, feb 2017. ISSN 10902643. doi: 10.1016/j.icarus.2016.08.012.

Joshua L. Bandfield, Rebecca R. Ghent, Ashwin R. Vasavada, David A. Paige, Samuel J. Lawrence, and Mark S. Robinson. Lunar surface rock abundance and regolith fines temperatures derived from LRO Diviner Radiometer data. *Journal of Geophysical Research E: Planets*, 116(12):0–02, dec 2011. ISSN 01480227. doi: 10.1029/2011JE003866. URL
https://onlinelibrary.wiley.com/doi/full/10.1029/2011JE003866https://onlinelibrary.wiley.com/doi/abs/10.1029/2011JE003866https://agupubs.onlinelibrary.wiley.com/doi/10.1029/2011JE003866.

NASA. NASA's Lunar Exploration Program Overview. 2020.

ESA. ESA Space Resources Strategy. 2019.

D. A. Paige, M. C. Foote, B. T. Greenhagen, J. T. Schofield, S. Calcutt, A. R. Vasavada, D. J. Preston, F. W. Taylor, C. C. Allen, K. J. Snook, B. M. Jakosky, B. C. Murray, L. A. Soderblom, B. Jau, S. Loring, J. Bulharowski, N. E. Bowles, I. R. Thomas, M. T. Sullivan, C. Avis, E. M. De Jong, W. Hartford, and D. J. McCleese. The lunar reconnaissance orbiter diviner lunar radiometer experiment. *Space Science Reviews*, 150 (1-4):125–160, jan 2010. ISSN 00386308. doi: 10.1007/s11214-009-9529-2.

Erwan Mazarico, Gregory A. Neumann, Michael K. Barker, Sander Goossens, David E. Smith, and Maria T. Zuber. Orbit determination of the Lunar Reconnaissance Orbiter: Status after seven years. *Planetary and Space Science*, 162:2–19, nov 2018. ISSN 00320633. doi: 10.1016/j.pss.2017.10.004.

Joshua L. Bandfield, Eugenie Song, Paul O. Hayne, Brittany D. Brand, Rebecca R. Ghent, Ashwin R. Vasavada, and David A. Paige. Lunar cold spots: Granular flow features and extensive insulating materials surrounding young craters. *Icarus*, 231:221–231, mar 2014. ISSN 00191035. doi: 10.1016/j.icarus.2013.12.017.

Paul O. Hayne, Joshua L. Bandfield, Matthew A. Siegler, Ashwin R. Vasavada, Rebecca R. Ghent, Jean-Pierre Williams, Benjamin T. Greenhagen, Oded Aharonson, Catherine M. Elder, Paul G. Lucey, and David A. Paige. Global Regolith Thermophysical Properties of the Moon From the Diviner Lunar Radiometer Experiment. *Journal of Geophysical Research: Planets*, 122(12):2371–2400, dec 2017. ISSN 21699097. doi: 10.1002/2017JE005387. URL `http://doi.wiley.com/10.1002/2017JE005387`.

E. Sefton-Nash, J. P. Williams, B. T. Greenhagen, T. J. Warren, J. L. Bandfield, K. M. Aye, F. Leader, M. A. Siegler, P. O. Hayne, N. Bowles, and D. A. Paige. Evidence for ultra-cold traps and surface water ice in the lunar south polar crater Amundsen. *Icarus*, 332:1–13, nov 2019. ISSN 10902643. doi: 10.1016/j.icarus.2019.06.002.

Benjamin T. Greenhagen, Paul G. Lucey, Michael B. Wyatt, Timothy D. Glotch, Carlton C. Allen, Jessica A. Arnold, Joshua L. Bandfield, Neil E. Bowles, Kerri L.Donaldson Hanna, Paul O. Hayne, Eugenie Song, Ian R. Thomas, and David A. Paige. Global silicate mineralogy of the moon from the diviner lunar radiometer. *Science*, 329(5998):1507–1509, sep 2010. ISSN 00368075. doi: 10.1126/science.1192196. URL `https://www.science.org/doi/abs/10.1126/science.1192196`.

Ari Silburt, Mohamad Ali-Dib, Chenchong Zhu, Alan Jackson, Diana Valencia, Yevgeni Kissin, Daniel Tamayo, and Kristen Menou. Lunar crater identification via deep learning. *Icarus*, 317:27–38, jan 2019. ISSN 10902643. doi: 10.1016/j.icarus.2018.06.022.

Gayantha R.L. Kodikara and Lindsay J. McHenry. Machine learning approaches for classifying lunar soils. *Icarus*, 345:113719, jul 2020. ISSN 10902643. doi: 10.1016/j.icarus.2020.113719.

Valentin Tertius Bickel, Charis Lanaras, Andrea Manconi, Simon Loew, and Urs Mall. Automated Detection of Lunar Rockfalls Using a Convolutional Neural Network. *IEEE Transactions on Geoscience and Remote Sensing*, 57(6):3501–3511, jun 2019. ISSN 01962892. doi: 10.1109/TGRS.2018.2885280.

Valentin Bickel, Jérôme Burelbach, Ben Moseley, and Nicole Relatores. A Big Data and AI-Driven Approach for Anomaly Detection on the Lunar Surface. In *51st Lunar and Planetary Science Conference. Program and Abstracts*, page 2066. Lunar and Planetary Institute, mar 2020.

Benjamin Wu, Potter Ross, Philippe Ludivig, Andrew S. Chung, and Timothy Seabrook. Absolute Localization Through Orbital Maps and Surface Perspective Imagery: A Synthetic Lunar Dataset and Neural Network Approach. In *IEEE International Conference on Intelligent Robots and Systems*, pages 3262–3267. Institute of Electrical and Electronics Engineers Inc., nov 2019. ISBN 9781728140049. doi: 10.1109/IROS40897.2019.8968124.

E. Sefton-Nash, J. P. Williams, B. T. Greenhagen, K. M. Aye, and D. A. Paige. Diviner lunar radiometer gridded brightness temperatures from geodesic binning of modeled

fields of view. *Icarus*, 298:98–110, dec 2017. ISSN 10902643. doi: 10.1016/j.icarus.2017.04.007.

Moon E J Speyerer, M S Robinson, B W Denevi, and Science Team. Lunar Reconnaissance Orbiter Camera global morphological map of the Moon. In *42nd Lunar Planetary Science Conference*, 2011. doi: https://www.lpi.usra.edu/meetings/lpsc2011/pdf/2387.pdf.

D.A. Paige, J.T. Schofield, S.B. Calcutt, C.C. Avis, H.B. Mortensen, and M.T. Sullivan. LRO-L-DLRE-4-RDR-V1.0, NASA Planetary Data System, 2011. URL `https://pds-geosciences.wustl.edu/lro/lro-l-dlre-4-rdr-v1/`.

Jianqing Feng, Matthew A. Siegler, and Paul O. Hayne. New Constraints on Thermal and Dielectric Properties of Lunar Regolith from LRO Diviner and CE-2 Microwave Radiometer. *Journal of Geophysical Research: Planets*, 125(1):e2019JE006130, jan 2020. ISSN 21699100. doi: 10.1029/2019JE006130. URL `https://onlinelibrary.wiley.com/doi/full/10.1029/2019JE006130https://onlinelibrary.wiley.com/doi/abs/10.1029/2019JE006130https://agupubs.onlinelibrary.wiley.com/doi/10.1029/2019JE006130`.

Carl Edward Rasmussen and Christopher K I Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *32nd International Conference on Machine Learning, ICML 2015*, volume 1, pages 448–456. International Machine Learning Society (IMLS), 2015. ISBN 9781510810587.

Peter I. Frazier. A Tutorial on Bayesian Optimization. *arXiv*, jul 2018. URL `http://arxiv.org/abs/1807.02811`.

J. P. Williams, E. Sefton-Nash, and D. A. Paige. The temperatures of Giordano Bruno crater observed by the Diviner Lunar Radiometer Experiment: Application of an effective field of view model for a point-based data set. *Icarus*, 273:205–213, jul 2016. ISSN 10902643. doi: 10.1016/j.icarus.2015.10.034.

James R Arnold. Ice in the lunar polar regions. *Journal of Geophysical Research*, 84(B10): 5659, sep 1979. ISSN 0148-0227. doi: 10.1029/jb084ib10p05659. URL `https://onlinelibrary.wiley.com/doi/full/10.1029/JB084iB10p05659https://onlinelibrary.wiley.com/doi/abs/10.1029/JB084iB10p05659https://agupubs.onlinelibrary.wiley.com/doi/10.1029/JB084iB10p05659`.

Elizabeth A. Fisher, Paul G. Lucey, Myriam Lemelin, Benjamin T. Greenhagen, Matthew A. Siegler, Erwan Mazarico, Oded Aharonson, Jean Pierre Williams, Paul O. Hayne, Gregory A. Neumann, David A. Paige, David E. Smith, and Maria T. Zuber. Evidence for surface water ice in the lunar polar regions using reflectance measurements from the Lunar Orbiter Laser Altimeter and temperature measurements from the Diviner Lunar Radiometer Experiment. *Icarus*, 292:74–85, aug 2017. ISSN 10902643. doi: 10.1016/j.icarus.2017.03.023.

A. Colaprete, R.C. Elphic, M. Shirley, K. Ennico-Smith, D.S.S. Lim, and K. Zacny. The Volatiles Investigating Polar Exploration Rover (VIPER) Mission – Measurements and Constraints. In *52nd Lunar and Planetary Science Conference 2021*, number 2548, page 1523, 2021. URL `https://ui.adsabs.harvard.edu/abs/2021LPI....52.1523C/abstract`.

Gordon Chin, Scott Brylow, Marc Foote, James Garvin, Justin Kasper, John Keller, Maxim Litvak, Igor Mitrofanov, David Paige, Keith Raney, Mark Robinson, Anton Sanin, David Smith, Harlan Spence, Paul Spudis, S. Alan Stern, and Maria Zuber. Lunar reconnaissance orbiter overview: The instrument suite and mission. *Space Science Reviews*, 129(4):391–419, apr 2007. ISSN 00386308. doi: 10.1007/s11214-007-9153-y. URL `https://link.springer.com/article/10.1007/s11214-007-9153-y`.

M. S. Robinson, S. M. Brylow, M. Tschimmel, D. Humm, S. J. Lawrence, P. C. Thomas, B. W. Denevi, E. Bowman-Cisneros, J. Zerr, M. A. Ravine, M. A. Caplinger, F. T. Ghaemi, J. A. Schaffner, M. C. Malin, P. Mahanti, A. Bartels, J. Anderson, T. N. Tran, E. M. Eliason, A. S. McEwen, E. Turtle, B. L. Jolliff, and H. Hiesinger. Lunar reconnaissance orbiter camera (LROC) instrument overview. *Space Science Reviews*, 150(1-4):81–124, jan 2010. ISSN 00386308. doi: 10.1007/s11214-010-9634-2. URL `https://asu.pure.elsevier.com/en/publications/lunar-reconnaissance-orbiter-camera-lroc-instrument-overview`.

E. Cisneros, A. Awumah, H.M. Brown, A.C. Martin, K.N. Paris, R.Z. Povilaitis, A.K. Boyd, and M.S. Robinson. Lunar Reconnaissance Orbiter Camera Permanently Shadowed Region Imaging – Atlas and Controlled Mosaics. In *Lunar and Planetary Science XLVIII (2017)*, 2017.

Lisa Gaddis, James L. Anderson, Kris J. Becker, T.L. Becker, D. Cook, K. Edwards, E.M. Eliason, Trent Hare, H. Kieffer, E.˜M. Lee, J. Mathews, L. Soderblom, T. Sucharski, J. M. Torson, A.S. McEwen, and Mark S. Robinson. An Overview of the Integrated Software for Imaging Spectrometers (ISIS). *Lunar and Planetary Institute Science Conference Abstracts*, 28:387, 1997. URL `https://ui.adsabs.harvard.edu/abs/1997LPI....28..387G/abstract`.

Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to See in the Dark. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3291–3300, 2018b. ISBN 9781538664209. doi: 10.1109/CVPR.2018.00347.

Frederick R. Chromey. *To Measure the Sky*. Cambridge University Press, oct 2016. doi: 10.1017/cbo9781316424117.

Yide Zhang, Yinhao Zhu, Evan Nichols, Qingfei Wang, Siyuan Zhang, Cody Smith, and Scott Howard. A poisson-gaussian denoising dataset with real fluorescence microscopy images. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:11702–11710, 2019b. ISSN 10636919. doi: 10.1109/CVPR.2019.01198.

Samuel W. Hasinoff, Jonathan T. Barron, Dillon Sharlet, Ryan Geiss, Florian Kainz, Jiawen Chen, Andrew Adams, and Marc Levoy. Burst photography for high dynamic range and low-light imaging on mobile cameras. *ACM Transactions on Graphics*, 35(6):

1–12, nov 2016. ISSN 15577368. doi: 10.1145/2980179.2980254. URL https://dl.acm.org/doi/10.1145/2980179.2980254.

Ahmet Serdar Karadeniz, Erkut Erdem, and Aykut Erdem. Burst Photography for Learning to Enhance Extremely Dark Images. *arXiv*, jun 2020. URL http://arxiv.org/abs/2006.09845https://hucvl.github.io/dark-burst-photography/.

A. Buades, B. Coll, and J. M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling and Simulation*, 4(2):490–530, jul 2005. ISSN 15403459. doi: 10.1137/040616024.

Bhawna Goyal, Ayush Dogra, Sunil Agrawal, B. S. Sohi, and Apoorav Sharma. Image denoising review: From classical to state-of-the-art approaches. *Information Fusion*, 55: 220–244, mar 2020. ISSN 15662535. doi: 10.1016/j.inffus.2019.09.003.

Linwei Fan, Fan Zhang, Hui Fan, and Caiming Zhang. Brief review of image denoising techniques. *Visual Computing for Industry, Biomedicine, and Art*, 2(1):7, dec 2019. doi: 10.1186/s42492-019-0016-7. URL https://vciba.springeropen.com/articles/10.1186/s42492-019-0016-7.

H. D. Cheng and X. J. Shi. A simple and effective histogram equalization approach to image enhancement. *Digital Signal Processing: A Review Journal*, 14(2):158–170, 2004. ISSN 10512004. doi: 10.1016/j.dsp.2003.07.002.

Artur Łoza, David R. Bull, Paul R. Hill, and Alin M. Achim. Automatic contrast enhancement of low-light images based on local statistics of wavelet coefficients. *Digital Signal Processing: A Review Journal*, 23(6):1856–1866, dec 2013. ISSN 10512004. doi: 10.1016/j.dsp.2013.06.002.

Seonhee Park, Soohwan Yu, Byeongho Moon, Seungyong Ko, and Joonki Paik. Low-light image enhancement using variational optimization-based retinex model. *IEEE Transactions on Consumer Electronics*, 63(2):178–184, may 2017. ISSN 00983063. doi: 10.1109/TCE.2017.014847.

Jun Xu, Yingkun Hou, Dongwei Ren, Li Liu, Fan Zhu, Mengyang Yu, Haoqian Wang, and Ling Shao. STAR: A Structure and Texture Aware Retinex Model. *IEEE Transactions on Image Processing*, 29:5022–5037, jun 2020a. ISSN 19410042. doi: 10.1109/TIP.2020.2974060. URL http://arxiv.org/abs/1906.06690http://dx.doi.org/10.1109/TIP.2020.2974060.

Xiaojie Guo, Yu Li, and Haibin Ling. LIME: Low-light image enhancement via illumination map estimation. *IEEE Transactions on Image Processing*, 26(2):982–993, feb 2017. ISSN 10577149. doi: 10.1109/TIP.2016.2639450.

Joseph Salmon, Zachary Harmany, Charles Alban Deledalle, and Rebecca Willett. Poisson noise reduction with non-local PCA. *Journal of Mathematical Imaging and Vision*, 48 (2):279–294, apr 2014. ISSN 09249907. doi: 10.1007/s10851-013-0435-6. URL https://link.springer.com/article/10.1007/s10851-013-0435-6.

Ke Xu, Xin Yang, Baocai Yin, and Rynson W.H. Lau. Learning to restore low-light images via decomposition-and-enhancement. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2278–2287, 2020b. doi: 10.1109/CVPR42600.2020.00235.

Kaixuan Wei, Ying Fu, Jiaolong Yang, and Hua Huang. A Physics-based Noise Formation Model for Extreme Low-light Raw Denoising. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020. URL `http://arxiv.org/abs/2003.12751`.

Paras Maharjan, Li Li, Zhu Li, Ning Xu, Chongyang Ma, and Yue Li. Improving extreme low-light image denoising via residual learning. In *Proceedings - IEEE International Conference on Multimedia and Expo*, volume 2019-July, pages 916–921. IEEE Computer Society, jul 2019. ISBN 9781538695524. doi: 10.1109/ICME.2019.00162.

Wenqi Ren, Sifei Liu, Lin Ma, Qianqian Xu, Xiangyu Xu, Xiaochun Cao, Junping Du, and Ming Hsuan Yang. Low-Light Image Enhancement via a Deep Hybrid Network. *IEEE Transactions on Image Processing*, 28(9):4364–4375, sep 2019. ISSN 19410042. doi: 10.1109/TIP.2019.2910412.

Tal Remez, Or Litany, Raja Giryes, and Alex M. Bronstein. Deep Convolutional Denoising of Low-Light Images. *arXiv*, jan 2017. URL `http://arxiv.org/abs/1701.01687`.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *arXiv*, may 2015. ISSN 16113349. doi: 10.1007/978-3-319-24574-4_28. URL `http://lmb.informatik.uni-freiburg.de/`.

Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. A High-Quality Denoising Dataset for Smartphone Cameras. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1692–1700, 2018. ISBN 9781538664209. doi: 10.1109/CVPR.2018.00182.

Tobias Plötz and Stefan Roth. Benchmarking denoising algorithms with real photographs. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 2750–2759. Institute of Electrical and Electronics Engineers Inc., jul 2017. ISBN 9781538604571. doi: 10.1109/CVPR.2017.294. URL `http://arxiv.org/abs/1707.01313`.

Shi Guo, Zifei Yan, Kai Zhang, Wangmeng Zuo, and Lei Zhang. Toward convolutional blind denoising of real photographs. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June, pages 1712–1722. IEEE Computer Society, jul 2019. ISBN 9781728132938. doi: 10.1109/CVPR.2019.00181. URL `http://arxiv.org/abs/1807.04686`.

H. M. Sargeant, V. T. Bickel, C. I. Honniball, S. N. Martinez, A. Rogaski, S. K. Bell, E. C. Czaplinski, B. E. Farrant, E. M. Harrington, G. D. Tolometti, and D. A. Kring. Using Boulder Tracks as a Tool to Understand the Bearing Capacity of Permanently Shadowed Regions of the Moon. *Journal of Geophysical Research: Planets*, 125(2), feb 2020. ISSN 21699100. doi: 10.1029/2019JE006157. URL `https://onlinelibrary.wiley.com/doi/abs/10.1029/2019JE006157`.

D. C. Humm, M. Tschimmel, S. M. Brylow, P. Mahanti, T. N. Tran, S. E. Braden, S. Wiseman, J. Danton, E. M. Eliason, and M. S. Robinson. Flight Calibration of the LROC Narrow Angle Camera. *Space Science Reviews*, 200(1-4):431–473, 2016. ISSN 15729672. doi: 10.1007/s11214-015-0201-8. URL http://dx.doi.org/10.1007/s11214-015-0201-8.

Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010. ISBN 1848829345.

H. Riris, G. Neuman, J. Cavanaugh, X. Sun, P. Liiva, and M. Rodriguez. The Lunar Orbiter Laser Altimeter (LOLA) on NASA's Lunar Reconnaissance Orbiter (LRO) mission. In Naoto Kadowaki, editor, *International Conference on Space Optics — ICSO 2010*, volume 10565, page 77. SPIE, nov 2017. ISBN 9781510616196. doi: 10.1117/12.2309209. URL https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10565/2309209/The-Lunar-Orbiter-Laser-Altimeter-LOLA-on-NASAs-Lunar-Reconnaissance/10.1117/12.2309209.full.

Padhraic Smyth. On Loss Functions Which Minimize to Conditional Expected Values and Posterior Probabilities. *IEEE Transactions on Information Theory*, 39(4):1404–1408, jul 1993. ISSN 15579654. doi: 10.1109/18.243457. URL http://ieeexplore.ieee.org/document/243457/.

Heiner Igel. *Computational seismology: a practical introduction*. Oxford University Press, 2017. ISBN 9780198717409.

David M. Boore. Simulation of ground motion using the stochastic method. *Pure and Applied Geophysics*, 160(3-4):635–676, 2003. ISSN 00334553. doi: 10.1007/PL00012553.

Yifeng Cui, Kim B Olsen, Thomas H Jordan, Kwangyoon Lee, Jun Zhou, Patrick Small, Daniel Roten, Geoffrey Ely, Dhabaleswar K. Panda, Amit Chourasia, John Levesque, Steven M. Day, and Philip Maechling. Scalable Earthquake Simulation on Petascale Supercomputers. In *2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, nov 2010. ISBN 978-1-4244-7557-5. doi: 10.1109/SC.2010.45. URL http://ieeexplore.ieee.org/document/5644908/.

Satinder Chopra and Kurt J. Marfurt. *Seismic Attributes for Prospect Identification and Reservoir Characterization*. Society of Exploration Geophysicists and European Association of Geoscientists and Engineers, jan 2007. doi: 10.1190/1.9781560801900.

David E. Lumley. Time-lapse seismic reservoir monitoring. *Geophysics*, 66(1):50–53, jan 2001. ISSN 19422156. doi: 10.1190/1.1444921.

Kasra Hosseini, Karin Sigloch, Maria Tsekhmistrenko, Afsaneh Zaheri, Tarje Nissen-Meyer, and Heiner Igel. Global mantle structure from multifrequency tomography using P, PP and P-diffracted waves. *Geophysical Journal International*, 220(1):96–141, 2019. ISSN 0956-540X. doi: 10.1093/gji/ggz394. URL https://doi.org/10.1093/gji/ggz394.

Ebru Bozdağ, Daniel Peter, Matthieu Lefebvre, Dimitri Komatitsch, Jeroen Tromp, Judith Hill, Norbert Podhorszki, and David Pugmire. Global adjoint tomography: first-generation model. *Geophysical Journal International*, 207(3):1739–1766, nov 2016.

ISSN 0956-540X. doi: 10.1093/gji/ggw356. URL https://doi.org/10.1093/gji/ggw356.

Martin Van Driel, Savas Ceylan, John Francis Clinton, Domenico Giardini, Hector Alemany, Amir Allam, David Ambrois, Julien Balestra, Bruce Banerdt, Dirk Becker, Maren Böse, Marc S. Boxberg, Nienke Brinkman, Titus Casademont, Jérôme Chèze, Ingrid Daubar, Anne Deschamps, Fabian Dethof, Manuel Ditz, Melanie Drilleau, David Essing, Fabian Euchner, Benjamin Fernando, Raphael Garcia, Thomas Garth, Harriet Godwin, Matthew P. Golombek, Katharina Grunert, Celine Hadziioannou, Claudia Haindl, Conny Hammer, Isabell Hochfeld, Kasra Hosseini, Hao Hu, Sharon Kedar, Balthasar Kenda, Amir Khan, Tabea Kilchling, Brigitte Knapmeyer-Endrun, Andre Lamert, Jiaxuan Li, Philippe Lognonné, Sarah Mader, Lorenz Marten, Franziska Mehrkens, Diego Mercerat, David Mimoun, Thomas Möller, Naomi Murdoch, Paul Neumann, Robert Neurath, Marcel Paffrath, Mark P. Panning, Fabrice Peix, Ludovic Perrin, Lucie Rolland, Martin Schimmel, Christoph Schröer, Aymeric Spiga, Simon Christian Stähler, René Steinmann, Eleonore Stutzmann, Alexandre Szenicer, Noah Trumpik, Maria Tsekhmistrenko, Cédric Twardzik, Renee Weber, Philipp Werdenbach-Jarklowski, Shane Zhang, and Yingcai Zheng. Preparing for InSight: Evaluation of the blind test for martian seismicity. *Seismological Research Letters*, 90 (4):1518–1534, jul 2019. ISSN 19382057. doi: 10.1785/0220180379.

Simon C. Stähler, Mark P. Panning, Steven D. Vance, Ralph D. Lorenz, Martin van Driel, Tarje Nissen-Meyer, and Sharon Kedar. Seismic Wave Propagation in Icy Ocean Worlds. *Journal of Geophysical Research: Planets*, 123(1):206–232, jan 2018. ISSN 21699100. doi: 10.1002/2017JE005338. URL http://doi.wiley.com/10.1002/2017JE005338.

Peter Moczo, Johan O.A. Robertsson, and Leo Eisner. The Finite-Difference Time-Domain Method for Modeling of Seismic Wave Propagation. *Advances in Geophysics*, 48:421–516, 2007. ISSN 00652687. doi: 10.1016/S0065-2687(06)48008-0. URL http://linkinghub.elsevier.com/retrieve/pii/S0065268706480080.

Dimitri Komatitsch and Jeroen Tromp. Spectral-element simulations of global seismic wave propagation - I. Validation. *Geophysical Journal International*, 149(2):390–412, may 2002a. ISSN 0956540X. doi: 10.1046/J.1365-246X.2002.01653.X/2/149-2-390-FIG015.JPEG. URL https://academic.oup.com/gji/article/149/2/390/727101.

Dimitri Komatitsch and Jeroen Tromp. Spectral-element simulations of global seismic wave propagation - II. Three-dimensional models, oceans, rotation and self-gravitation. *Geophysical Journal International*, 150(1):303–318, jul 2002b. ISSN 0956540X. doi: 10.1046/J.1365-246X.2002.01716.X/2/150-1-303-FIG021.JPEG. URL https://academic.oup.com/gji/article/150/1/303/593406.

Kuangdai Leng, Tarje Nissen-Meyer, Martin van Driel, Kasra Hosseini, and David Al-Attar. AxiSEM3D: broad-band seismic wavefields in 3-D global earth models with undulating discontinuities. *Geophysical Journal International*, 217(3):2125–2146, feb 2019. ISSN 0956-540X. doi: 10.1093/gji/ggz092. URL https://doi.org/10.1093/gji/ggz092.

Martin van Driel and Tarje Nissen-Meyer. Optimized viscoelastic wave propagation for weakly dissipative media. *Geophysical Journal International*, 199(2):1078–1093, sep

2014a. ISSN 0956-540X. doi: 10.1093/gji/ggu314. URL https://doi.org/10.1093/gji/ggu314.

Martin van Driel and Tarje Nissen-Meyer. Seismic wave propagation in fully anisotropic axisymmetric media. *Geophysical Journal International*, 199(2):880–893, sep 2014b. ISSN 0956-540X. doi: 10.1093/gji/ggu269. URL https://doi.org/10.1093/gji/ggu269.

Jeroen Tromp, Carl Tape, and Qinya Liu. Seismic tomography, adjoint methods, time reversal and banana-doughnut kernels. *Geophysical Journal International*, 160(1): 195–216, 2005. ISSN 0956540X. doi: 10.1111/j.1365-246X.2004.02453.x.

Max Rietmann, Peter Messmer, Tarje Nissen-Meyer, Daniel Peter, Piero Basini, Dimitri Komatitsch, Olaf Schenk, Jeroen Tromp, Lapo Boschi, and Domenico Giardini. Forward and adjoint simulations of seismic wave propagation on emerging large-scale GPU architectures. In *International Conference for High Performance Computing, Networking, Storage and Analysis, SC*, 2012. ISBN 9781467308069. doi: 10.1109/SC.2012.59.

Thomas Bohlen. Parallel 3-D viscoelastic finite difference seismic modelling. *Computers & Geosciences*, 28(8):887–899, oct 2002. ISSN 00983004. doi: 10.1016/S0098-3004(02)00006-7. URL http://linkinghub.elsevier.com/retrieve/pii/S0098300402000067.

Max Rietmann, Daniel Peter, Olaf Schenk, Bora Ucar, and Marcus Grote. Load-Balanced Local Time Stepping for Large-Scale Wave Propagation. In *Proceedings - 2015 IEEE 29th International Parallel and Distributed Processing Symposium, IPDPS 2015*, pages 925–935. Institute of Electrical and Electronics Engineers Inc., jul 2015. ISBN 9781479986484. doi: 10.1109/IPDPS.2015.10.

Xiao Ma, Dinghui Yang, Meixia Wang, and Guojie Song. A low-dispersive symplectic partitioned runge–kutta method for solving seismic-wave equations: I. scheme and theoretical analysis. *Bulletin of the Seismological Society of America*, 104(5):2206–2225, oct 2014. ISSN 19433573. doi: 10.1785/0120120210. URL http://pubs.geoscienceworld.org/ssa/bssa/article-pdf/104/5/2206/2675429/2206.pdf.

Daniel Peter, Dimitri Komatitsch, Yang Luo, Roland Martin, Nicolas Le Goff, Emanuele Casarotti, Pieyre Le Loher, Federica Magnoni, Qinya Liu, Céline Blitz, Tarje Nissen-Meyer, Piero Basini, and Jeroen Tromp. Forward and adjoint simulations of seismic wave propagation on fully unstructured hexahedral meshes. *Geophysical Journal International*, 186(2):721–739, aug 2011. ISSN 0956540X. doi: 10.1111/j.1365-246X.2011.05044.x. URL https://onlinelibrary.wiley.com/doi/full/10.1111/j.1365-246X.2011.05044.xhttps://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-246X.2011.05044.xhttps://onlinelibrary.wiley.com/doi/10.1111/j.1365-246X.2011.05044.x.

Christian Pelties, Josep De La Puente, Jean Paul Ampuero, Gilbert B. Brietzke, and Martin Käser. Three-dimensional dynamic rupture simulation with a high-order discontinuous Galerkin method on unstructured tetrahedral meshes. *Journal of Geophysical Research: Solid Earth*, 117(2), feb 2012. ISSN 21699356. doi: 10.1029/2011JB008857.

Wolfgang Friederich and Jörg Dalkolmo. Complete synthetic seismograms for a spherically symmetric earth by a numerical computation of the Green's function in the frequency domain. *Geophysical Journal International*, 122(2):537–550, sep 1995. ISSN 1365246X. doi: 10.1111/j.1365-246X.1995.tb07012.x. URL `https://academic.oup.com/gji/article/122/2/537/964934`.

Kenji Kawai, Nozomu Takeuchi, and Robert J. Geller. Complete synthetic seismograms up to 2 Hz for transversely isotropic spherically symmetric media. *Geophysical Journal International*, 164(2):411–424, feb 2006. ISSN 0956540X. doi: 10.1111/j.1365-246X.2005.02829.x. URL `https://academic.oup.com/gji/article/164/2/411/776381`.

Genti Toyokuni and Hiroshi Takenaka. FDM computation of seismic wavefield for an axisymmetric earth with a moment tensor point source. *Earth, Planets and Space*, 58 (8):e29–e32, sep 2006. ISSN 18805981. doi: 10.1186/bf03352593. URL `https://earth-planets-space.springeropen.com/articles/10.1186/BF03352593`.

T Nissen-Meyer, M. Van Driel, S C Stähler, K Hosseini, S Hempel, L Auer, A Colombi, and A Fournier. AxiSEM: Broadband 3-D seismic wavefields in axisymmetric media. *Solid Earth*, 5(1):425–445, jun 2014. ISSN 18699529. doi: 10.5194/se-5-425-2014. URL `https://www.solid-earth.net/5/425/2014/`.

Keiiti Aki and Paul G Richards. *Quantitative seismology*. W. H. Freeman and Co., 1980.

V. Vinje, E. Iversen, and H. Gjoystdal. Traveltime and amplitude estimation using wavefront construction. *Geophysics*, 58(8):1157–1166, feb 1993. ISSN 00168033. doi: 10.1190/1.1443499.

Mirko Van Der Baan and Christian Jutten. Neural networks in geophysical applications. *Geophysics*, 65(4):1032–1047, feb 2000. ISSN 00168033. doi: 10.1190/1.1444797.

Michael E. Murat and Albert J. Rudman. Automated first arrival picking: a neural network approach. *Geophysical Prospecting*, 40(6):587–604, aug 1992. ISSN 0016-8025. doi: 10.1111/j.1365-2478.1992.tb00543.x. URL `http://doi.wiley.com/10.1111/j.1365-2478.1992.tb00543.x`.

Farid U Dowla, Steven R Taylor, and Russell W Anderson. Seismic discrimination with artificial neural networks: Preliminary results with regional spectral data. *Bulletin of the Seismological Society of America*, 80(5):1346–1373, oct 1990. ISSN 0037-1106.

M. M. Poulton, B. K. Sternberg, and C. E. Glass. Location of subsurface targets in geophysical data using neural networks. *Geophysics*, 57(12):1534–1544, feb 1992. ISSN 00168033. doi: 10.1190/1.1443221.

Gunter Röth and Albert Tarantola. Neural networks and inversion of seismic data. *Journal of Geophysical Research*, 99(B4):6753, apr 1994. ISSN 0148-0227. doi: 10.1029/93JB01563. URL `http://doi.wiley.com/10.1029/93JB01563`.

Sankar Kumar Nath, Subrata Chakraborty, Sanjiv Kumar Singh, and Nilanjan Ganguly. Velocity inversion in cross-hole seismic tomography by counter-propagation neural network, genetic algorithm and evolutionary programming techniques. *Geophysical Journal International*, 138(1):108–124, jul 1999. ISSN 0956540X. doi: 10.1046/j.1365-246X.1999.00835.x.

Karianne J. Bergen, Paul A. Johnson, Maarten V. De Hoop, and Gregory C. Beroza. Machine learning for data-driven discovery in solid Earth geoscience. *Science*, 363 (6433), 2019. ISSN 10959203. doi: 10.1126/science.aau0323.

Qingkai Kong, Daniel T. Trugman, Zachary E. Ross, Michael J. Bianco, Brendan J. Meade, and Peter Gerstoft. Machine learning in seismology: Turning data into insights. *Seismological Research Letters*, 90(1):3–14, 2019. ISSN 19382057. doi: 10.1785/0220180259.

R. J. R. Devilee, A. Curtis, and K. Roy-Chowdhury. An efficient, probabilistic neural network approach to solving inverse problems: Inverting surface wave velocities for Eurasian crustal thickness. *Journal of Geophysical Research: Solid Earth*, 104(B12): 28841–28857, dec 1999. ISSN 2169-9356. doi: 10.1029/1999jb900273.

Andrew P. Valentine and Jeannot Trampert. Data space reduction, quality assessment and searching of seismograms: autoencoder networks for waveform data. *Geophysical Journal International*, 189(2):1183–1202, may 2012. ISSN 0956540X. doi: 10.1111/j.1365-246X.2012.05429.x. URL `https://academic.oup.com/gji/article-lookup/doi/10.1111/j.1365-246X.2012.05429.x`.

Thibaut Perol, Michaël Gharbi, and Marine Denolle. Convolutional neural network for earthquake detection and location. *Science Advances*, 4(2):e1700578, feb 2018. ISSN 23752548. doi: 10.1126/sciadv.1700578. URL `http://arxiv.org/abs/1702.02073http://advances.sciencemag.org/lookup/doi/10.1126/sciadv.1700578`.

Mauricio Araya-Polo, Joseph Jennings, Amir Adler, and Taylor Dahlke. Deep-learning tomography. *The Leading Edge*, 37(1):58–66, 2018. ISSN 1070-485X. doi: 10.1190/tle37010058.1. URL `https://library.seg.org/doi/10.1190/tle37010058.1`.

Yue Wu and Youzuo Lin. InversionNet: A Real-Time and Accurate Full Waveform Inversion with CNNs and continuous CRFs. *arXiv*, pages 1–14, 2018. URL `http://arxiv.org/abs/1811.07875`.

Fangshu Yang and Jianwei Ma. Deep-learning inversion: A next-generation seismic velocity model building method. *Geophysics*, 84(4):R583–R599, 2019. ISSN 0016-8033. doi: 10.1190/geo2018-0249.1.

Hongyu Sun and Laurent Demanet. Low frequency extrapolation with deep learning. *2018 SEG International Exposition and Annual Meeting, SEG 2018*, pages 2011–2015, 2018. doi: 10.1190/segam2018-2997928.1.

Weiqiang Zhu, Yixiao Sheng, and Yi Sun. Wave-dynamics simulation using deep neural networks. *Stanford Report*, 2017.

Ali Siahkoohi, Mathias Louboutin, and Felix J. Herrmann. Neural network augmented wave-equation simulation. *arXiv*, sep 2019. URL `http://arxiv.org/abs/1910.00925`.

Lion Krischer and A Fichtner. Generating Seismograms with Deep Neural Networks. *AGU Fall Meeting Abstracts*, page kr, 2017.

Yan Yang, Angela F. Gao, Jorge C. Castellanos, Zachary E. Ross, Kamyar Azizzadenesheli, and Robert W. Clayton. Seismic Wave Propagation and Inversion with Neural Operators. *The Seismic Record*, 1(3):126–134, oct 2021b. ISSN 2694-4006. doi: 10.1785/0320210026. URL `https://orcid.org/0000-0003-3323-`.

Jonathan D. Smith, Kamyar Azizzadenesheli, and Zachary E. Ross. EikoNet: Solving the Eikonal Equation with Deep Neural Networks. *IEEE Transactions on Geoscience and Remote Sensing*, 59(12):10685–10696, dec 2021. ISSN 15580644. doi: 10.1109/TGRS.2020.3039165.

Sadegh Karimpouli and Pejman Tahmasebi. Physics informed machine learning: Seismic wave equation. *Geoscience Frontiers*, 11(6):1993–2001, nov 2020. ISSN 16749871. doi: 10.1016/j.gsf.2020.07.007.

Majid Rasht-Behesht, Christian Huber, Khemraj Shukla, and George Em Karniadakis. Physics-informed Neural Networks (PINNs) for Wave Propagation and Full Waveform Inversions. *arXiv*, aug 2021. URL `https://arxiv.org/abs/2108.12035v1http://arxiv.org/abs/2108.12035`.

Yiran Xu, Jingye Li, and Xiaohong Chen. Physics informed neural networks for velocity inversion. In *SEG International Exposition and Annual Meeting 2019*, pages 2584–2588. Society of Exploration Geophysicists, 2019a. doi: 10.1190/segam2019-3216823.1.

Khemraj Shukla, Patricio Clark Di Leoni, James Blackshire, Daniel Sparkman, and George Em Karniadakis. Physics-Informed Neural Network for Ultrasound Nondestructive Quantification of Surface Breaking Cracks. *Journal of Nondestructive Evaluation*, 39(3):1–20, sep 2020. ISSN 15734862. doi: 10.1007/s10921-020-00705-1. URL `https://link.springer.com/article/10.1007/s10921-020-00705-1`.

Chengping Rao, Hao Sun, and Yang Liu. Physics-Informed Deep Learning for Computational Elastodynamics without Labeled Data. *Journal of Engineering Mechanics*, 147(8):04021043, may 2021. ISSN 0733-9399. doi: 10.1061/(asce)em.1943-7889.0001947. URL `https://ascelibrary.org/doi/abs/10.1061/{%}28ASCE{%}29EM.1943-7889.0001947`.

Chao Song, Tariq Alkhalifah, and Umair Bin Waheed. Solving the frequency-domain acoustic VTI wave equation using physics-informed neural networks. *Geophysical Journal International*, 225(2):846–859, apr 2021. ISSN 1365246X. doi: 10.1093/gji/ggab010. URL `https://academic.oup.com/gji/article/225/2/846/6081098`.

Guihua Long, Yubo Zhao, and Jun Zou. A temporal fourth-order scheme for the first-order acoustic wave equations. *Geophysical Journal International*, 194(3): 1473–1485, sep 2013. ISSN 1365-246X. doi: 10.1093/gji/ggt168. URL `http://academic.oup.com/gji/article/194/3/1473/642909/A-temporal-fourthorder-scheme-for-the-firstorder`.

Brian H Russell. *Introduction to Seismic Inversion Methods*. Society of Exploration Geophysicists, 1988. doi: 10.1190/1.9781560802303. URL `https://library.seg.org/doi/abs/10.1190/1.9781560802303`.

Vinod Nair and Geoffrey Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair. In *Proceedings of ICML*, volume 27, pages 807–814, 2010.

Dimitri Komatitsch and Roland Martin. An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation. *Geophysics*, 72(5): SM155–SM167, 2007.

Gary F. Margrave and Michael P. Lamoureux. *Numerical Methods of Exploration Seismology*. Cambridge University Press, dec 2018. doi: 10.1017/9781316756041.

B. Gutenberg. The amplitudes of waves to be expected in seismic prospecting. *Geophysics*, 1(2):252–256, jun 1936. ISSN 0016-8033. doi: 10.1190/1.1437101.

P. Newman. Divergence effects in a layered earth. *Geophysics*, 38(3):481–488, jun 1973. ISSN 0016-8033. doi: 10.1190/1.1440353.

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*, pages 265–283, 2016. ISBN 9781931971331. doi: 10.5555/3026877.3026899.

Gary S. Martin, Robert Wiley, and Kurt J. Marfurt. Marmousi2: An elastic upgrade for Marmousi. *Leading Edge (Tulsa, OK)*, 25(2):156–166, apr 2006. ISSN 1070485X. doi: 10.1190/1.2172306.

Zhongping Zhang and Youzuo Lin. Data-driven Seismic Waveform Inversion: A Study on the Robustness and Generalization. *arXiv*, pages 1–13, 2018. URL `http://arxiv.org/abs/1809.10262`.

Stephanie Earp and Andrew Curtis. Probabilistic neural network-based 2D travel-time tomography. *Neural Computing and Applications*, pages 1–19, may 2020. ISSN 14333058. doi: 10.1007/s00521-020-04921-8.

Öz Yilmaz. *Seismic Data Analysis*. Society of Exploration Geophysicists, jan 2001. doi: 10.1190/1.9781560801580.

Eman Ahmed, Alexandre Saint, Abd El Rahman Shabayek, Kseniya Cherenkova, Rig Das, Gleb Gusev, Djamila Aouada, and Bjorn Ottersten. A survey on Deep Learning Advances on Different 3D Data Representations. *arXiv*, aug 2018. URL `http://arxiv.org/abs/1808.01462`.

Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016. URL `http://www.cs.ox.ac.uk/people/yarin.gal/website/thesis/thesis.pdf`.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepOnets. *arXiv*, mar 2021d. URL `https://arxiv.org/abs/2103.10974v1http://arxiv.org/abs/2103.10974`.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. *arXiv*, jun 2020. URL http://arxiv.org/abs/2006.09661.

Zhi Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv*, jan 2019b. ISSN 1815-2406. doi: 10.4208/cicp.oa-2020-0085. URL http://arxiv.org/abs/1901.06523http://dx.doi.org/10.4208/cicp.OA-2020-0085.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxlcr, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *36th International Conference on Machine Learning, ICML 2019*, volume 2019-June, pages 9230–9239. International Machine Learning Society (IMLS), jun 2019. ISBN 9781510886988. URL http://arxiv.org/abs/1806.08734.

Ronen Basri, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies. In *Advances in Neural Information Processing Systems*, volume 32. Neural information processing systems foundation, jun 2019. URL http://arxiv.org/abs/1906.00425.

Yuan Cao, Zhiying Fang, Yue Wu, Ding-Xuan Zhou, and Quanquan Gu. Towards Understanding the Spectral Bias of Deep Learning. *arXiv*, dec 2019. URL http://arxiv.org/abs/1912.01198.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, oct 2021e. ISSN 00457825. doi: 10.1016/j.cma.2021.113938.

Ziqi Liu, Wei Cai, and Zhi Qin John Xu. Multi-scale deep neural network (MscaleDNN) for solving poisson-boltzmann equation in complex domains. *Communications in Computational Physics*, 28(5):1970–2001, nov 2020. ISSN 19917120. doi: 10.4208/CICP.OA-2020-0179.

Alexander Heinlein, Axel Klawonn, Martin Lanser, and Janine Weber. Combining machine learning and domain decomposition methods for the solution of partial differential equations—A review. *GAMM-Mitteilungen*, 44(1):e202100001, mar 2021. ISSN 0936-7195. doi: 10.1002/gamm.202100001. URL https://onlinelibrary.wiley.com/doi/10.1002/gamm.202100001.

Ameya D. Jagtap and George Em Karniadakis. Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5):2002–2041, nov 2020. ISSN 19917120. doi: 10.4208/CICP.OA-2020-0164.

Khemraj Shukla, Ameya D. Jagtap, and George Em Karniadakis. Parallel physics-informed neural networks via domain decomposition. *Journal of Computational Physics*, 447:110683, dec 2021. ISSN 10902716. doi: 10.1016/j.jcp.2021.110683.

Vikas Dwivedi, Nishant Parashar, and Balaji Srinivasan. Distributed learning machines for solving forward and inverse problems in partial differential equations. *Neurocomputing*, 420:299–316, jan 2021. ISSN 18728286. doi: 10.1016/j.neucom.2020.09.006.

Suchuan Dong and Zongwei Li. Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 387:114129, dec 2021. ISSN 00457825. doi: 10.1016/j.cma.2021.114129.

Guang Bin Huang, Qin Yu Zhu, and Chee Kheong Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1-3):489–501, dec 2006. ISSN 09252312. doi: 10.1016/j.neucom.2005.12.126.

Ke Li, Kejun Tang, Tianfan Wu, and Qifeng Liao. D3M: A Deep Domain Decomposition Method for Partial Differential Equations. *IEEE Access*, 8:5283–5294, 2020c. ISSN 21693536. doi: 10.1109/ACCESS.2019.2957200.

Patrick Stiller, Friedrich Bethke, Maximilian Böhme, Richard Pausch, Sunna Torge, Alexander Debus, Jan Vorberger, Michael Bussmann, and Nico Hoffmann. Large-Scale Neural Solvers for Partial Differential Equations. In Jeffrey Nichols, Becky Verastegui, Arthur 'Barney' Maccabe, Oscar Hernandez, Suzanne Parete-Koon, and Theresa Ahearn, editors, *Driving Scientific and Engineering Discoveries Through the Convergence of HPC, Big Data and AI*, pages 20–34, Cham, 2020. Springer International Publishing. ISBN 978-3-030-63393-6.

Luning Sun, Han Gao, Shaowu Pan, and Jian Xun Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361, jun 2020. ISSN 00457825. doi: 10.1016/j.cma.2019.112732. URL `http://arxiv.org/abs/1906.02382http://dx.doi.org/10.1016/j.cma.2019.112732`.

Isaac Elias Lagaris, Aristidis C. Likas, and Dimitrios G. Papageorgiou. Neural-network methods for boundary value problems with irregular boundaries. *IEEE Transactions on Neural Networks*, 11(5):1041–1049, sep 2000. ISSN 10459227. doi: 10.1109/72.870037.

Jens Berg and Kaj Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, nov 2018. ISSN 18728286. doi: 10.1016/j.neucom.2018.06.056. URL `http://arxiv.org/abs/1711.06464http://dx.doi.org/10.1016/j.neucom.2018.06.056`.

Carl Leake and Daniele Mortari. Deep Theory of Functional Connections: A New Method for Estimating the Solutions of Partial Differential Equations. *Machine Learning and Knowledge Extraction*, 2(1):37–55, mar 2020. ISSN 2504-4990. doi: 10.3390/make2010004. URL `https://europepmc.org/articles/PMC7259480https://europepmc.org/article/pmc/pmc7259480`.

C. Basdevant, M. Deville, P. Haldenwang, J. M. Lacroix, J. Ouazzani, R. Peyret, P. Orlandi, and A. T. Patera. Spectral and finite difference solutions of the Burgers equation. *Computers and Fluids*, 14(1):23–41, jan 1986. ISSN 00457930. doi: 10.1016/0045-7930(86)90036-8.

Petar Veličković and Charles Blundell. Neural algorithmic reasoning. *Patterns*, 2(7), may 2021. ISSN 26663899. doi: 10.1016/j.patter.2021.100273. URL http://arxiv.org/abs/2105.02761http://dx.doi.org/10.1016/j.patter.2021.100273.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical Text-Conditional Image Generation with CLIP Latents. *arXiv*, apr 2022. doi: 10.48550/arxiv.2204.06125. URL https://arxiv.org/abs/2204.06125v1http://arxiv.org/abs/2204.06125.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling Language Modeling with Pathways. *arXiv*, apr 2022. doi: 10.48550/arxiv.2204.02311. URL https://arxiv.org/abs/2204.02311v2http://arxiv.org/abs/2204.02311.

Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar. FourCastNet: A Global Data-driven High-resolution Weather Model using Adaptive Fourier Neural Operators. *arXiv*, feb 2022. URL https://arxiv.org/abs/2202.11214v1http://arxiv.org/abs/2202.11214.

Marten Lienen and Stephan Günnemann. Learning the Dynamics of Physical Systems from Sparse Observations with Finite Element Networks. In *ICLR 2022*, pages 1–29, mar 2022. doi: 10.48550/arxiv.2203.08852. URL https://arxiv.org/abs/2203.08852v1.

Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian data analysis*. CRC Press, jan 2013. ISBN 9781439898208. doi: 10.1201/b16018.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. B-VAE: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.

Y C Zheng, K L Chan, K T Tsang, Y C Zhu, G P Hu, Y L Zou, and Z Y Ouyang. Catalogue of Lunar Thermal Anomalies. In *45th Lunar and Planetary Science Conference*, 2014. doi: https://www.hou.usra.edu/meetings/lpsc2014/pdf/2208.pdf.

Hideo Tsunakawa, Hidetoshi Shibuya, Futoshi Takahashi, Hisayoshi Shimizu, Masaki Matsushima, Ayako Matsuoka, Satoru Nakazawa, Hisashi Otake, and Yuichi Iijima. Lunar magnetic field observation and initial global mapping of lunar magnetic anomalies by MAP-LMAG onboard SELENE (Kaguya). *Space Science Reviews*, 154 (1-4):219–251, may 2010. ISSN 00386308. doi: 10.1007/s11214-010-9652-0. URL `https://link.springer.com/article/10.1007/s11214-010-9652-0`.

J. P. Williams, B. T. Greenhagen, D. A. Paige, N. Schorghofer, E. Sefton-Nash, P. O. Hayne, P. G. Lucey, M. A. Siegler, and K. Michael Aye. Seasonal Polar Temperatures on the Moon. *Journal of Geophysical Research: Planets*, 124(10):2505–2521, oct 2019. ISSN 21699100. doi: 10.1029/2019JE006028.