NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES
Data Science and Information Technologies

SPECIALIZATION
Big Data and Artificial Intelligence

MASTER THESIS

Physics-informed operator learning for solving ODE/PDEs with time-dependent forcing term

Ioannis-Christos Kandias

ATHENS

July 2024

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΔΙΙΔΡΥΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
Επιστήμη Δεδομένων και Τεχνολογίες Πληροφορίας

ΕΙΔΙΚΕΥΣΗ
Μεγάλα δεδομένα και Τεχνητή Νοημοσύνη

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Επιβλεπόμενοι με φυσική νευρωνικοί τελεστές για επίλυση συνήθων και μερικών διαφορικών εξισώσεων (ΟΔΕ/ΜΔΕ) με πηγή εξαρτώμενη στο χρόνο

Ιωάννης-Χρήστος Κάνδιας

ΑΘΗΝΑ

Ιούλιος 2024

MSc THESIS

Physics-informed operator learning for solving ODE/PDEs with time-dependent forcing term

Ioannis-Christos Kandias

S.N.: 7115152100018

SUPERVISORS: **Manolis Koubarakis ,** Professor NKUA
Shuai Guo, Scientist ABB
Sandro Schönborn, Principal Scientist ABB

EXAMINATION COMMITTEE: **Manolis Koubarakis**, Professor NKUA
Konstantinos Koutroumbas, Research Director NOA
Ioannis Panagakis, Associate Professor NKUA

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Επιβλεπόμενοι με φυσική νευρωνικοί τελεστές για επίλυση συνήθων και μερικών διαφορικών εξισώσεων (ΟΔΕ/ΜΔΕ) με πηγή εξαρτώμενη στο χρόνο

Ιωάννης-Χρήστος Κάνδιας

Α.Μ.: 7115152100018

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Μανόλης Κουμπαράκης,** Καθηγητής ΕΚΠΑ
Shuai Guo, Επιστημονικό πρωσοπικό ABB
Sandro Schönborn, Κυρίαρχο Επιστημονικό πρωσοπικό ABB

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **Μανόλης Κουμπαράκης**, Καθηγητής ΕΚΠΑ
Κωνσταντίνος Κουτρούμπας, Διευθυντής Ερευνών ΙΑΑΔΕΤ
Ιωάννης Παναγάκης, Αναπληρωτής Καθηγητής ΕΚΠΑ

ABSTRACT

The advent of powerful computational resources has fostered the simulation of various physics problems by solving their differential equations at accelerated speeds. However, despite these technological advancements, the complexity of the entire process remains a challenge for practitioners while the computational costs remain relatively high and escalate further when considering the multitude of simulations required to explore different inputs, boundaries, and initial conditions within a physical system across various scenarios typically encountered in practice. In this work, motivated by the imperative to address these challenges, we introduce and evaluate the efficacy of a deep operator network (DeepONet) along with its physics-informed variant designed to be trained without paired input-output data, thereby enhancing the generalizability of the operator network. The thesis addresses the operator architecture in two distinct scenarios: an Ordinary Differential Equation (ODE) forced oscillator system and a practical Partial Differential Equation (PDE) lithium-ion battery scenario. Analysis of ODE setup reveals that training with higher frequency inputs and using a hybrid approach (combining data and physics) boosts the accuracy and generalizability of the system to lower frequency inputs while implementing a sinusoidal activation function in solely the first layer of trunk net can offer an additional enhancement. In the case of the battery scenario, a series of approaches are explored to model the system with a primary focus on leveraging physics, considering the limited literature available on this burning issue; revealing that a significant improvement can be achieved by enforcing the PDE further to the initial and boundary conditions on the surface of the system. This groundwork thus sets the stage for future research in this underexplored topic, thereby contributing to this developing domain.

SUBJECT AREA: Scientific Machine Learning

KEYWORDS: Operator Learning, Physics Informed Neural Networks (PINNs), Physics-Informed DeepONet (PI-DeepONet), Scientific Machine Learning (SciML), Lithium-Ion Batteries, Forced Oscillator

ΠΕΡΙΛΗΨΗ

Η τεράστια αύξηση της διαθέσιμης υπολογιστικής ισχύος έχει καταστήσει δυνατή την προσομοίωση διαφόρων προβλημάτων φυσικής επιλύοντας τις διαφορικές εξισώσεις τους γρηγορότερα. Ωστόσο, παρόλες αυτές τις τεχνολογικές εξελίξεις, η πολυπλοκότητα της όλης διαδικασίας παραμένει πρόκληση για τον εκάστοτε επαγγελματία, ενώ το υπολογιστικό κόστος παραμένει ακόμα σχετικά υψηλό. Αυτό το κόστος αυξάνεται περαιτέρω όταν εξετάζεται το πλήθος των προσομοιώσεων που απαιτούνται για την εξερεύνηση διαφορετικών φορτίων, συνοριακών και αρχικών συνθηκών μέσα σε ένα φυσικό σύστημα στα διάφορα σενάρια που συνήθως συναντώνται στην πράξη. Σε αυτήν την εργασία, με κίνητρο την επιτακτική ανάγκη να αντιμετωπιστούν αυτές οι προκλήσεις, παρουσιάζουμε και αξιολογούμε την αποτελεσματικότητα ενός νευρωνικού δικτύου βαθίας μάθησης για τελεστές (DeepONet) καθώς και την ενσωματωμένη με φυσική του προβλήματος παραλλαγή του, η οποία έχει σχεδιαστεί για να εκπαιδεύεται χωρίς ζεύγη δεδομένων εισόδου-εξόδου, ενισχύοντας έτσι την ικανότητα γενίκευσης του νευρωνικού δικτύου-τελεστή. Η διπλωματική προχωρά μετέπειτα στην διευθέτηση της αρχιτεκτονικής του νευρωνικού δικτύου τελεστή με βάση δύο διαφορετικά σενάρια: ένα σύστημα εξαναγκασμένου ταλαντωτή με σύνηθη διαφορική εξίσωση (ΣΔΕ) και ένα πρακτικό σενάριο μπαταρίας ιόντων-λιθίου που περιγράφεται από μερική διαφορική εξισώση (ΜΔΕ). Η ανάλυση του εξαναγκάσμενου ταλαντωτή αποκαλύπτει οτι η εκπαίδευση σε φορτία υψηλότερων συχνοτήτων και η χρήση υβριδικής προσέγγισης (συνδυασμός φυσικής και δεδομένων) ενισχύει την ακρίβεια και την ικανότητα γενίκευσης του μοντέλου σε χαμηλότερες συχνότητες, ενώ η χρήση ημιτονοειδούς συνάρτησης ενεργοποίησης αποκλειστικά στο πρώτο στρώμα του κύριου δικτύου (trunk net) δύναται να επιφέρει περαιτέρω βελτιώσεις. Στην περίπτωση της μπαταρίας, διερευνάται μια σειρά προσεγγίσεων για τη μοντελοποίηση του συστήματος αποκλειστικά με βάση τη φυσική γνώση, λαμβάνοντας υπόψη την περιορισμένη διαθέσιμη βιβλιογραφία σχετικά με αυτό το φλέγον ζήτημα, αποκαλύπτωντας οτι σημαντική βελτίωση μπορεί να επιτευχθεί εάν η ΜΔΕ εφαρμοσθεί επιπρόσθετα στις αρχικές και συνοριακές συνθήκες στην επιφάνεια του συστήματος. Έτσι, παρούσα εργασία θέτει τις βάσεις για μελλοντική έρευνα σε αυτό το ανεξερεύνητο πεδίο, συμβάλλοντας έτσι στην περαιτέρω εξέλιξή του.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Επιστημονική Μηχανική Μάθηση

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Μάθηση τελεστών, Νευρωνικά δίκτυα με εμφωλευμένη φυσική γνώση, Επιστημονική Μηχανική Μάθηση, Βαθιά μάθηση τελεστών με εμφωλευμένη φυσική γνώση (PI-DeepONet), Μπαταρίες λιθίου-ιόντων, Εξαναγκασμένες Ταλάντωσεις

To my grandmother, my parents, friends, and all the people that I met during this master's journey

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Dr. Shuai Guo and Dr. Sandro Schönborn for not only giving me the opportunity to undertake this thesis but also for their invaluable guidance, insightful discussions, and all the advice given that brought this thesis to fruition and enriched my research journey. My gratitude extends also to thank Prof. Manolis Koubarakis for accepting and overseeing this project.

I am also profoundly thankful to Dr. Philipp Sommer for his remarkable support and for providing me with the resources that I needed for my thesis. Additionally, I would like to acknowledge many members of ABB in the research center who assisted me with their expertise and to all the members for their welcoming atmosphere and stimulating discussions.

Last but not least, I owe a debt of gratitude to my family and friends. Their unwavering support and encouragement have been my pillar of strength throughout the journey of this thesis.

CONTENTS

Abstract **5**

Acknowledgements **8**

List of Figures **11**

List of Tables **14**

1 INTRODUCTION **15**

1.1 Motivation 15

1.2 Scope and Objective 16

2 Theory **17**

2.1 Operator Learning and DeepONet 17

2.1.1 Structure of DeepONet 18

2.1.2 Dataset Structure of a DeepONet 19

2.1.3 Other popular operator networks 20

2.2 Physics-informed learning and PI-DeepONet 20

2.2.1 Physics Informed Machine Learning (PIML) 20

2.2.2 Physics Informed DeepONet 24

2.3 Physics-Informed Operator Learning over PINNs 24

2.4 SciML challenges 25

3 ODE case study (Forced Oscillator) **27**

3.1 Problem Description 27

3.2 Dataset 28

3.2.1 Dataset Composition 29

3.3 Neural network architecture 30

3.4 Base model 30

3.5 Trunk Net sinusoidal layer 32

3.6 Lower-length scale training 35

3.6.1 Base model 35

3.6.2 Trunk Net sinusoidal layer 37

3.7 Comparison between hybrid, data, and only physics will be displayed . . . 39

4 PDE case study (battery case) 41
4.1 Introduction . . . 41
4.2 Single Particle Model . . . 42
4.3 Relevant Literature . . . 45
4.4 Dataset . . . 46
4.4.1 PyBaMM . . . 46
4.4.2 DeepONet dataset . . . 46
4.4.3 PI-DeepONet dataset . . . 47
4.5 Neural network architecture . . . 48
4.6 DeepONet Results . . . 49
4.7 PI-DeepONet Results . . . 51
4.7.1 Base architecture . . . 52
4.7.2 Enhanced Base Architecture Utilizing Quadruple Collocation Points 54
4.7.3 Adjusted loss function . . . 55
4.7.3.1 1st epoch division . . . 56
4.7.3.2 50^{th} epoch division & 10 times PDE loss . . . 56
4.7.4 Enforce PDE constraints on BC & IC . . . 57
4.7.4.1 PDE at $BC_{surface}$ . . . 58
4.7.4.2 PDE at IC&$BC_{surface}$ . . . 60
4.7.5 Best model . . . 63

5 CONCLUSIONS AND FUTURE WORK 65
5.1 Summary . . . 65
5.2 Future work . . . 65

ABBREVIATIONS - ACRONYMS 67

REFERENCES 68

LIST OF FIGURES

2.1 A)is an illustration of how the training data look like, the input functions are discretized in a fixed number of sensors per input function (not necessarily equispaced) and random evaluation points B) demonstrates the firstly proposed architecture of DeepONet [15] that receives the discretized input functions and evaluation points in separate networks and performs element-wise multiplication with bias adding for generalizability 18

2.2 An Illustration of how physics can be embedded to a machine learning model and effects that can be passed in each way [8] 21

2.3 Three possible categories and the associated available data [8] 22

2.4 An illustration of a PINN 23

2.5 Physics-Informed DeepONet Illustrated [31] 24

3.1 Oscillator with a forcing term $l = 0.4$ and its exact solution 29

3.2 Oscillator with a forcing term $l = 0.1$ and its exact solution 29

3.3 Training loss of the base PI-DeepONet with $l = 0.4$ 31

3.4 Prediction results of the base PI-DeepONet for $l = 0.4$ (interpolation) 31

3.5 Prediction results of the base PI-DeepONet trained on $l = 0.4$ to the whole spectrum of the examined length scales except $l = 0.4$ 31

3.6 Prediction score capabilities of PI-DeepONet for various length scales . . . 32

3.7 Training loss of the base PI-DeepONet with sinus activation layer with $l = 0.4$ 33

3.8 Prediction results of the base PI-DeepONet with sinus activation layer for $l = 0.4$ (interpolation) 33

3.9 Prediction results of the base PI-DeepONet with sinus activation layer for all the length scales (extrapolation) except the trained $l = 0.4$ 34

3.10 Model generalizability comparison for trained length scale $l = 0.4$ 34

3.11 Prediction results of the base PI-DeepONet with being trained at $l = 0.1$ for $l = 0.1$ (interpolation) 35

3.12 Prediction results of the base PI-DeepONet with being trained at $l = 0.1$ for $l \in [0.2, 0.7]$ (extrapolation) 36

3.13 Prediction score capabilities of PI-DeepONet trained with $l = 0.1$ for various length scales 36

3.14 Prediction results of the base PI-DeepONet with sinus activation layer trained with $l = 0.1$ for $l = 0.1$ (interpolation) 37

3.15 Prediction results of the PI-DeepONet with a sinus activation function in the trunk net and being trained at $l = 0.1$ for $l \in [0.2, 0.7]$ (extrapolation) 38

3.16 Model generalizability comparison for trained length scale $l = 0.1$ 38

4.1 Illustration of an SPM model from [22] 42
4.2 Input current to the DeepONet 47
4.3 Output voltage of the DeepONet 47
4.4 Input current $[\mathrm{A}]$ to the PI-DeepONet for $\tau = 1$ 48
4.5 Concentration $\left[\frac{\mathrm{mol}}{\mathrm{m}^3}\right]$ for the negative particle for $\tau = 1$ 48
4.6 DeepOnet illustration for battery 49
4.7 Physics informed DeepOnet illustration for battery 49
4.8 DeepONet predictions for 300 points with the corresponding discharge rates 50
4.9 DeepONet predictions for 1000 points with the corresponding discharge rates 51
4.10 (a) Actual concentration obtained from PyBaMM, (b) Prediction of the neural network model, and (c) Absolute percentage error for the whole domain . . 53
4.11 (a) Initial concentration prediction $\left[\frac{\mathrm{mol}}{\mathrm{m}^3}\right]$, (b) 1^{st} gradient with respect to the radius at $\mathrm{BC}_{\mathrm{center}}$, and (c) 1^{st} gradient with respect to the radius at $\mathrm{BC}_{\mathrm{surface}}$. 53
4.12 (a) Actual concentration obtained from PyBaMM, (b) Prediction of the neural network model, and (c) Absolute percentage error across the entire domain for the more collocation points 54
4.13 (a) Initial concentration prediction $\left[\frac{\mathrm{mol}}{\mathrm{m}^3}\right]$, (b) 1^{st} gradient with respect to the radius at $\mathrm{BC}_{\mathrm{center}}$, and (c) 1^{st} gradient with respect to the radius at $\mathrm{BC}_{\mathrm{surface}}$ for the more collocation points 55
4.14 (a) Initial concentration prediction $\left[\frac{\mathrm{mol}}{\mathrm{m}^3}\right]$, (b) prediction of BC in the center, and (c) prediction of BC in the surface for the readjusted loss function . . . 56
4.15 (a) Initial concentration prediction $\left[\frac{\mathrm{mol}}{\mathrm{m}^3}\right]$, (b) prediction of BC in the center, and (c) prediction of BC in the surface for the readjusted loss function with 10 times the PDE loss 57
4.16 (a) the PDE applied to the boundary in the surface and (b) the PDE applied at the initial condition 58
4.17 (a) Actual concentration obtained from PyBaMM, (b) Prediction of the neural network model, and (c) Absolute percentage error for the whole domain for the PDE loss applied to $\mathrm{BC}_{\mathrm{surface}}$ model 59
4.18 (a) Initial concentration prediction $\left[\frac{\mathrm{mol}}{\mathrm{m}^3}\right]$, (b) 1^{st} gradient with respect to the radius at $\mathrm{BC}_{\mathrm{center}}$, and (c) 1^{st} gradient with respect to the radius at $\mathrm{BC}_{\mathrm{surface}}$ for the PDE loss applied to $\mathrm{BC}_{\mathrm{surface}}$ model 59
4.19 PDE at $\mathrm{BC}_{\mathrm{surface}}$ 60
4.20 (a) Actual concentration obtained from PyBaMM, (b) Prediction of the neural network model, and (c) Absolute percentage error for the whole domain for the PDE applied to IC&$\mathrm{BC}_{\mathrm{surface}}$ model 61
4.21 (a) Initial concentration prediction $\left[\frac{\mathrm{mol}}{\mathrm{m}^3}\right]$, (b) 1^{st} gradient with respect to the radius at $\mathrm{BC}_{\mathrm{center}}$, and (c) 1^{st} gradient with respect to the radius at $\mathrm{BC}_{\mathrm{surface}}$ for the PDE loss applied to IC&$\mathrm{BC}_{\mathrm{surface}}$ model 61
4.22 (a) the PDE applied to the boundary in the surface and (b) the PDE applied at the initial condition for the PDE at IC&$\mathrm{BC}_{\mathrm{surface}}$ model 62

4.23 Comparison of models with enforcement of PDE at IC and $BC_{surface}$ 62

4.24 (a) Actual concentration obtained from PyBaMM, (b) Prediction of the neural network model, and (c) Absolute percentage error for the whole domain for the best model 64

4.25 (a) Initial concentration prediction $\left[\frac{mol}{m^3}\right]$, (b) 1^{st} gradient with respect to the radius at BC_{center}, and (c) 1^{st} gradient with respect to the radius at $BC_{surface}$ for the best model 64

LIST OF TABLES

3.1 RMSE comparison for physics only, data only, and hybrid trained with $l = 0.1$ models with and without sinusoidal pre-activation layer [1] 39

4.1 Parameters used in the SPM model for the physics-informed case 44

4.2 RMSE for DeepONet testing data 50

4.3 Metrics for PI-DeepONet 54

4.4 Metrics for PI-DeepONet with for 4 times more collocation points 55

4.5 Metrics for PI-DeepONet with a readjusted loss function 56

4.6 Metrics for PI-DeepONet for the readjusted loss function with 10 times the PDE loss 57

4.7 Metrics for PI-DeepONet losses for the PDE loss applied to $BC_{surface}$ model 59

4.8 Metrics for PI-DeepONet losses for the PDE loss applied to IC&$BC_{surface}$ model 61

4.9 Metrics for PI-DeepONet losses for the best model 64

1. INTRODUCTION

In recent years, the pervasive influence of Artificial Intelligence (AI) has become palpable, permeating virtually every facet of modern life. In this explosion of AI, Deep Learning (DL) has stood as a catalyst, propelling unprecedented advancements in diverse fields, including computer vision, natural language processing and speech recognition, revolutionizing how machines perceive and interact with the world. From these advancements one field that couldn't be missing is the venerable realm of physics. The conjunction of these worlds -AI and physics- gave birth to a new field called *Scientific Machine Learning* (SciML). The first efforts for scientific machine learning were recorded a long time ago [11, 19], but it wasn't until more recently that this field started to bloom again with the seminal work of Raissi [21] that introduced the *Physics Informed Neural Networks* (PINNs) as a new approach for deep learning networks to be able to incorporate physical laws. This approach accomplishes its goal by integrating the physical laws into the cost function of the model.

As a natural consequence of the subsequent rise of SciML, newer, more sophisticated techniques were conceived for solving science and engineering problems. These fresher approaches lay on the notion of operator, which is described in section 2.
The method that will be examined in this dissertation is the Deep Operator Network (DeepONet) [15].

1.1 Motivation

SciML represents a novel paradigm within AI and physics that can solve any partial differential equations (PDEs) or ordinary differential equations (ODEs). PDEs and ODEs lie at the core of any engineering application (fluid dynamics, structural dynamics, chemical reactions, etc.) and constitute the way to solve these problems by running time-consuming simulations. For reference only, these simulations are being solved with numerical methods like Finite Difference Methods (FDM), Finite Volume Methods (FVM) and Finite Element Methods (FEM).

As mentioned, SciML can be considered as an alternative way of solving PDEs/ODEs and as a consequence, they appear as a prominent way for many prospects of academia and industry. Some indicative examples where the SciML can be utilized are the energy industry, earthquake and climate forecasting, automotive and aerospace, and many more.

One of the key benefits, that drive the research and development of this rapidly emerging field is that these techniques can deliver fast results in complex problems compared to the traditional solvers (FVM, FEM, FDM) after being trained or serve as surrogate models of the already well-developed approaches for assisting in a shorter simulation execution. The approach that will be analyzed in this dissertation is about the forward problem, but a promising area in which SciML can thrive also is the inverse problems, where repetitive simulations and statistics need to be performed to arrive to the variables that are being searched [1].

[1] Forward problem refers to when one knows all the physical and geometric properties of a system and seeks the response of it. Conversely, inverse problems involve unknown aspects of the system's physical or geometric properties, which must either be determined through experimentation or tailored to optimize a specific objective

Furthermore, another advantageous point of SciML compared to traditional machine learning approaches is that these techniques don't need experimental data. SciML models can be trained with directly incorporating the physical governing equations of the system, thus reducing the need for collecting training data and transforming these methods to cheap. As a consequence of this, the subsequent models will be capable of respecting the corresponding fundamental physical laws that govern each system making them interpretable, which is in contrast to a classical machine learning method that ignores these principles.

1.2 Scope and Objective

The recent blooming field of scientific machine learning has brought a plethora of new inventions and lightning development of promising techniques. Nevertheless, still many of these practices remain theoretical and the understanding of these is being possessed from a small merit of the scientific community. Moreover, the dominant domain of physics and engineering has numerous branches, which the scientific community, that investigates machine learning methods for physical problems, hasn't yet been able to examine or explore thoroughly. As a consequence, many commercial and academic fields of engineering have open-ended questions.

The focus of this dissertation will be on researching and exploring the applicability of these scientific machine-learning techniques for partial and ordinary differential equations that correspond to a forced oscillator and a battery system. The aforementioned will be probed not as individual problems, solving a sole case problem each time an input is given, but from the scope of view of operator learning. For the case of the battery system, one important note that should be made is that the literature for SciML on batteries is at an infant level yet, contributing to the novelty of the work.

For the development and running of all of the subsequent proposed models, the deep learning framework TensorFlow [1] was used.

The structure of the thesis will follow as:

Chapter 1 a brief introduction about the intersection of artificial intelligence with physics, the motivations driving this interdisciplinary direction, and the objective of the thesis
Chapter 2 the theory for the scientific machine learning but from the lens of operator learning, how can the operator-developed techniques be integrated with the underlying physics and a subsection for discussing why physics-informed neural networks are not suitable for the present context
Chapter 3 definition and analysis of the forced oscillator (ODE) problem, including evaluation and comparison of models trained solely on data, solely on physics, and using hybrid DeepONet approaches
Chapter 4 definition and analysis of the lithium-ion battery (PDE) problem, featuring evaluation of models trained on data and physics independently using DeepONet, alongside an exploration of various methodologies to address the inherent challenges posed by the latter approach
Chapter 5 a brief summary of the results, conclusions drawn from the findings, and recommendations for future research work

2. THEORY

2.1 Operator Learning and DeepONet

To begin with, in the terminology of mathematics, an operator represents a mapping or function of one set to produce another, each of which has a specific structure. This mapping can be linear or non-linear and in practice in engineering it symbolizes a particular problem (for example in the heat equation there is a mapping between the spatiotemporal variables and the system's temperature).

Following the above, researchers from Brown University exhibited a pioneering architecture for a neural network to learn operators. It is widely known that neural networks can be universal approximators of continuous functions[1] but it is less known that these networks given the correct architecture can be used to predict also functions, i.e. mappings from functions to functions. (in the case of DeepONet non-linear functions).

This means that these models can predict the solution of a problem, i.e. take a function as input and output another function with the corresponding mapping (for example, a load force applied on a surface with the subsequent displacement across this surface). This innovative network, DeepONet [15], is part of the scientific domain, SciML, and it will be presented below as it constitutes the backbone of the current thesis.

The conception of DeepONet is based on the theorem of Chen & Chen [3], which puts the ground for the *universal approximation theorem for operators*. This theorem had remained unexplored for many years and for the scope of the thesis is presented below:

Firstly, we introduce the corresponding notation that will be needed for establishing the theorem. Let G denote an operator that takes an input function u, yielding the corresponding output function $G(u)$. For any point y within the domain of $G(u)$, the output $G(u)(y)$ is a real number. Thus, the network receives inputs consisting of two components: u and y, and produces $G(u)(y)$ as its output (see Fig. 2.1). While the objective is to learn operators that accept functions as inputs, it is necessary to discretize the input functions to facilitate the application of network approximations. In practice, a straightforward and practical approach is to utilize the function values at highly reasonable discrete point locations $\{x_1, x_2, ..., x_m\}$ that suffice for the examining problem. These point locations are also called "*sensors*".

Theorem 1 (Universal Approximation Theorem for Operator) *Suppose that σ is a continuous nonpolynomial function, X is a Banach Space, $K_1 \subset X, K_2 \subset R^d$ are two compact sets in X and $\mathbb{R}^d$, respectively, V is a compact set in $C(K_1)$, G is a nonlinear continuous operator, which maps V into $C(K_2)$. Then for any $\epsilon > 0$, there are positive integers n,p,m, constants $c_i^k, \xi_i j^k, \theta_i j^k, \zeta_k \in \mathbb{R}, w_k \in \mathbb{R}^d, x_j \in K_1, i = 1, ..., n, k = 1, ..., p, j = 1, ..., m$, such that*

$$\left| G(u)(y) - \sum_{k=1}^{p} \underbrace{\sum_{i=1}^{n} c_i^k \sigma\left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right)}_{\text{branch}} \underbrace{\sigma(w_k \cdot y + \zeta_k)}_{\text{trunk}} \right| < \varepsilon \tag{2.1}$$

[1] A continuous function maintains a smooth transition in its values without any abrupt changes. Specifically, by limiting alterations in the input, we can ensure sufficiently small variations in the output of a continuous function

holds for all $u \in V$ *and* $y \in K_2$.

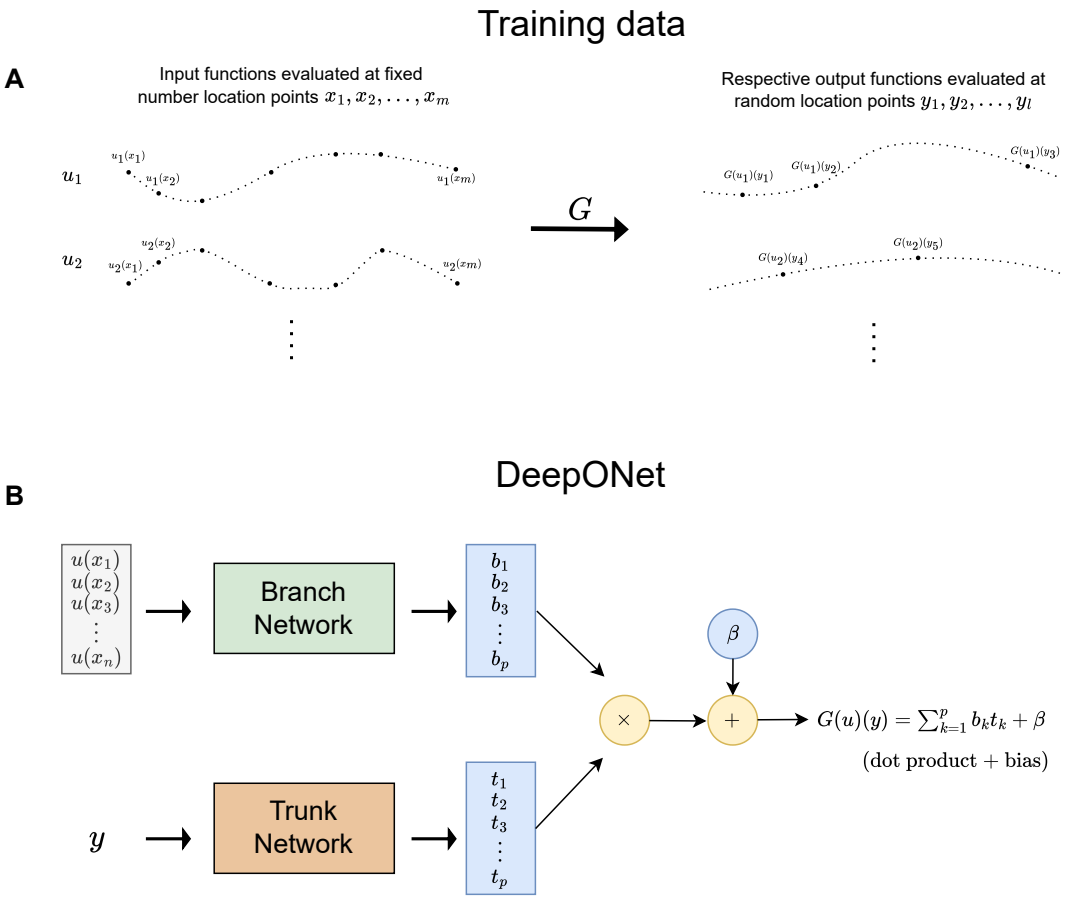


Figure 2.1: A)is an illustration of how the training data look like, the input functions are discretized in a fixed number of sensors per input function (not necessarily equispaced) and random evaluation points B) demonstrates the firstly proposed architecture of DeepONet [15] that receives the discretized input functions and evaluation points in separate networks and performs elementwise multiplication with bias adding for generalizability

2.1.1 Structure of DeepONet

Delving deeper into the architecture of a DeepONet, we see that it consists of two subnetworks, the branch and trunk (called also branchnet and trunknet). The branch as it is evident from figure 2.1 takes as input the force/source function discretized in a sufficient number of location points to be able to capture the dynamic of it. Here, in this network it's where the only constraint comes from, the training dataset need to be consistent with the sensors in the input functions. The trunk net on the other hand receives as input the location where the model will be evaluated. The type of architecture that will be used for either the branch net or the trunk net is flexible (MLP, CNN, RNN) and dependent on the researcher and the type of available data. In the vanilla DeepONet the networks that were used were MLPs since their data didn't have any specific structure. After, the inputs are passed from each of the networks an elementwise multiplication takes place between the trunknet and branchnet (and so the networks at their last layer need to have the same dimensionality), and a bias is added at the product of them.

So, the architecture of the model takes in the branchnet as input, the input function, evaluated at the sensors $[u(x_1), u(x_2), ..., u(x_m)]^T$, where u(·) is the input function and $x_1, x_2 ..., x_m$ are the locations of the sensors and outputs a vector $[b_1, b_2, ..., b_p]^T \in \mathbb{R}^p$ for $k = 1, 2, ..., p$ and trunknet receives the evaluation location y and outputs a vector $[t_1, t_2, ..., t_p]^T \in \mathbb{R}^p$. Then these networks are merged as in the equation 2.1 and a bias $b_0 \in \mathbb{R}$ is added.

$$G(u)(y) \approx \sum_{k=1}^{p} b_k t_k + b_0. \tag{2.2}$$

The bias although is not mentioned in the approximation theorem for operators, it can increase performance by reducing the generalization error. Also in the theorem, one hidden layer network is stated but in practice, the researchers of DeepONet used deeper network architectures for higher expressivity. The output $G(u)(y)$ has two independent variables and can be conceived as a function of y conditioned on u.

2.1.2 Dataset Structure of a DeepONet

So, from all the above, the DeepONet receives a dataset in the form of a triplet $(\boldsymbol{u}^x, y_l^x, G(u^x, y_l^x))$, where the vector $\boldsymbol{u}$ represents the x forcing function discretized at m points, the y_l^x the evaluation points of the inputted forcing functions evaluated at l points and the $G(\boldsymbol{u}^x, y_l^x))$ the output function evaluated at y_l^x evaluation point. The forcing function is drawn from a function space $x \in X$ and the evaluation points from a space $l \in L \times X$, where L is the total points where the output function G is evaluated.

Suppose we have an one dimensional forcing function and one dimensional evaluation point and we evaluate the whole dataset, an example of the illustrated dataset can be seen below:

$$\left(\begin{array}{cccccc}
u_1^1 & \cdots & u_m^1 & y_1^1 & G(\boldsymbol{u}^1, y_1^1) \\
u_1^1 & \cdots & u_m^1 & y_2^1 & G(\boldsymbol{u}^1, y_2^1) \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
u_1^1 & \cdots & u_m^1 & y_l^1 & G(\boldsymbol{u}^1, y_l^1) \\
u_1^2 & \cdots & u_m^2 & y_{l+1}^2 & G(\boldsymbol{u}^2, y_{l+1}^2) \\
\vdots & \ddots & \vdots & \vdots & \vdots \\
u_1^X & \cdots & u_m^X & y_L^X & G(\boldsymbol{u}^X, y_L^X)
\end{array}\right)$$

$$\underbrace{\qquad\qquad}_{\text{forcing function}} \quad \underbrace{\qquad}_{\substack{\text{Evaluation} \\ \text{point}}} \quad \underbrace{\qquad\qquad}_{\substack{\text{Output} \\ \text{Evaluation} \\ \text{point}}}$$

Usually, since the dataset can be quite large, a DeepONet is trained with samples of the original dataset. One beneficial characteristic of this model is that the evaluation points don't need to be in a grid or arranged in a specific pattern. The only requirement is to be sufficient enough to capture the dynamics and transitions of the forcing function. Finally, the DeepONet is getting trained with the widely adopted mean squared error (MSE).

$$L = \frac{1}{X \times L} \sum_{i=1}^{X} \sum_{j=1}^{L} \left(G(\boldsymbol{u}^i, y_j^i) - \widehat{G(\boldsymbol{u}^i, y_j^i)} \right)^2 \tag{2.3}$$

Where $\widehat{G(\boldsymbol{u}^i, y_j^i)}$ is representing the prediction of the model.

2.1.3 Other popular operator networks

For reference, there are two other notable classes of operators in scientific machine learning, the first one is the graph neural operators [14] that have in their base a graph convolution kernel, and the Fourier neural operators [13] that take the input they apply a kernel operator that is based on discrete Fourier transform, then drops the high frequencies and apply the reverse Fourier transformation while maintaining some learnable features. The last kind of operator is very efficient in learning high-frequency inputs and so it has been applied to turbulence problems in fluid dynamics.

The last category of operators, the Fourier Neural Operators (FNO), has been the focus of active research from the scientific community. In this frame, in [16] an attempt is made for a fair comparison of the DeepONet and the FNO, both theoretically and computationally but also by benchmarking them in 16 datasets. The outcome is that these two architectures exhibit close accuracy and performance in terms of the required computational resources, however, the DeepONet preserves the advantage of flexibility in the kind of architecture that will be used for the subnetworks.

2.2 Physics-informed learning and PI-DeepONet

Data-driven solutions for modeling physical problems tend to require a significant amount of data that are usually measured from sensors. These data can be difficult to acquire while they are accompanied by the costs of gathering and preserving them. On top of that, none can guarantee that these data won't be noisy because of any shortage or simply because the sensor recorded a noisy measure. These are some of the characteristic problems that accompany the modeling of physics problems with data-learning techniques. To alleviate these issues, researchers invented solutions that integrate the governing physics in the models.

This rapidly rising domain has been called physics informed machine learning.

2.2.1 Physics Informed Machine Learning (PIML)

In the well-established domain of physics, modeling a physical problem it's an arduous task and a job that requires a domain expert. The domain expert is the one who needs to create the proper mesh, apply the correct parameters and equations, and run time-consuming simulations.

For these reasons and because of the unforeseen surge of machine learning in the last years, there have been attempts to try to model physical problems with machine learning. According to Karniadakis, one of the leading scientists in this field, there are three principal ways to introduce physics in machine learning and that is by steering the learning process with observational, inductive, or learning biases [8].

- *Observational biases* may arise either through the inherent characteristics of the data, which encapsulate the underlying physics. By training a machine learning (ML) system on such data, it can acquire the ability to discern and learn functions,

vector fields, and operators that faithfully represent the inherent physical structure encoded within the data.

- *Inductive biases* refer to assumptions that are drawn from prior knowledge of the task to be predicted and can be integrated through specialized neural network architecture or through customized interventions. These interventions ensure that the model's predictions inherently adhere to a predefined set of physical laws, often articulated as specific mathematical constraints. In essence, modifying the ML model's architecture aligns it with the anticipated physical principles governing the process at hand. A close example of that is the convolution networks that have revolutionized computer vision
- *Learning biases* is knowledge that is passed to the model in a soft manner with penalizing accordingly the loss function to meet physical constraints. The learning biases can be perceived as a multitasking learning problem, where the model attempts to fit more than one loss function, such as the observed data and physical loss

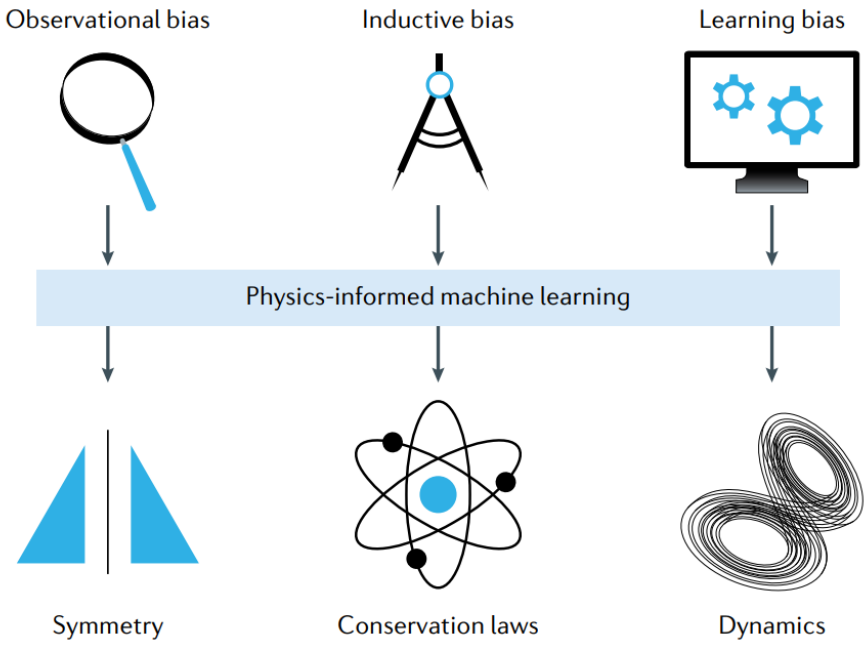


Figure 2.2: An Illustration of how physics can be embedded to a machine learning model and effects that can be passed in each way [8]

Physics-informed neural networks have the ability to combine all the above biases or some of them in pairs.

So, regarding the nature of the problem and the availability of data, the aspiring scientist can choose between three categories for solving a PIML.

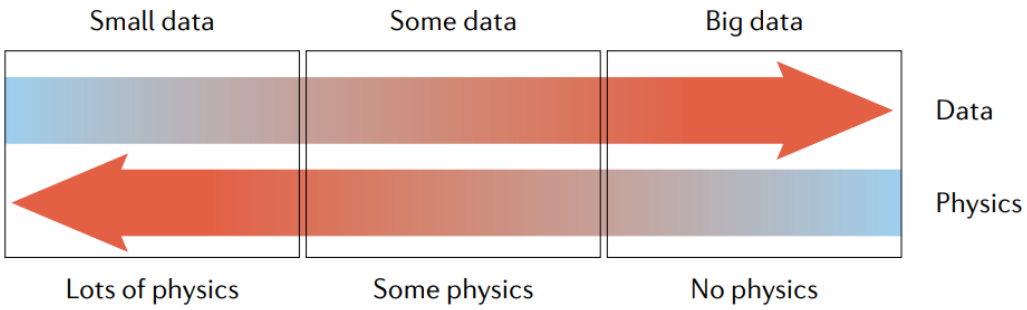


Figure 2.3: Three possible categories and the associated available data [8]

In the small data regime, it is assumed that the physics model is well defined (PDEs/ODEs, boundary, and initial conditions), in the some data regime it is possible that some data or properties of the real world problem are missing and in the big data case, no physics at all is required to make sufficient predictions.

But what is a PIML

PIML are common neural network architectures (commonly referred as Physics Informed Neural Networks, PINNs). What makes them different is the way they get trained. At PIML, the neural network takes advantage of the differentiation capabilities of the deep learning frameworks. They have the model serving as a surrogate and they differentiate it according to the corresponding system's physical laws (PDEs/ODEs), then they take the gradients of the surrogate model and transform them in the governing PDE which is used as a loss function. Moreover, from the model are received the boundaries and initial conditions that describe the system and used as a loss function[2]. Additionally, the model can be trained also with data and use them with a corresponding loss function such as MSE. Then on each of these losses are applied coefficients and the model is trained with the aggregation of these losses as its total loss.

[2]In the realm of physics and maths, a PDE/ODE in order to have a unique solution, needs to be provided with boundary and initial conditions. Otherwise, the examined equation has infinite solutions

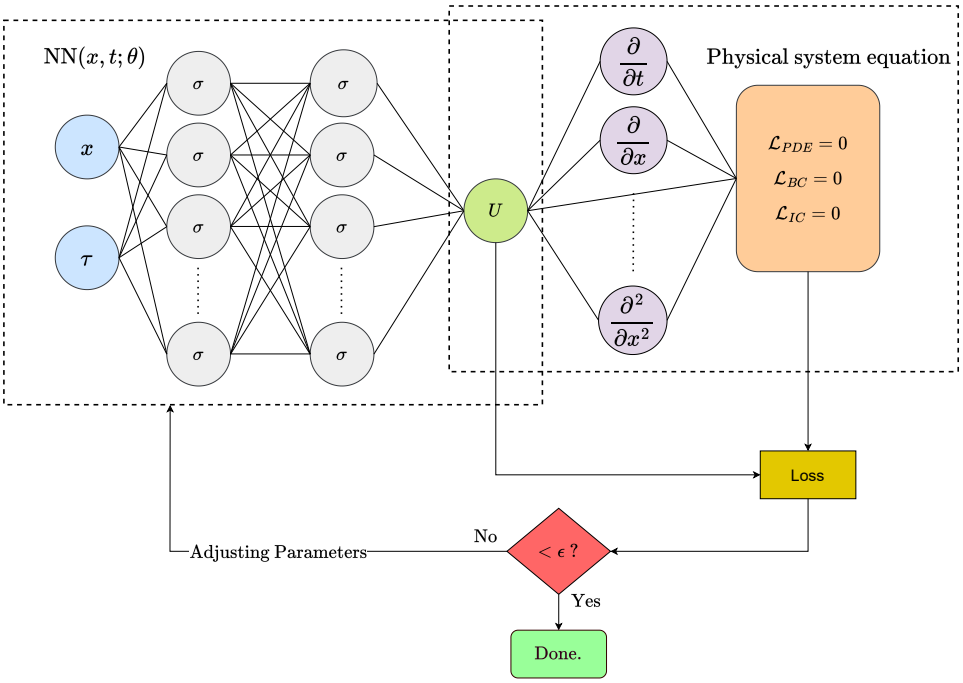


Figure 2.4: An illustration of a PINN

So, suppose we have a system u that is controlled by $\theta \in \Theta$, where θ is either functions or vectors of the system that are embodied in the system's equations (PDEs/ODEs, boundaries and initial conditions). The system receives $x \in \Omega$ spatial variables, where $x = (x_1, x_2, ..., x_d) \in \Omega$ can be any arbitrary number of independent variables, $t \in T$ temporal coordinates and has the $u(x,t)$ state variables to govern it. Taking all these into consideration, this system can be represented with the following equations:

$$\text{Partial differential equation:} \mathcal{F}(u;\theta)(x,t) = 0 \tag{2.4}$$

$$\text{Boundary conditions:} \mathcal{B}(u;\theta)(x,t) = 0 \tag{2.5}$$

$$\text{Initial Conditions:} \mathcal{I}(u;\theta)(x,t_0) = 0 \tag{2.6}$$

The boundary conditions can be either vectors or derivatives and in these equations the $x \in \partial\Omega$, where $\partial\Omega$ are the points in the boundaries. As for the initial equations the $t \in T_0$ represents the initial state of the system.

After defining the system, these functions can be given to a deep learning model as loss functions (corresponds to the learning biases) and apply the proper differentiation as is evident from the Figure 2.4 where the model is differentiated accordingly to meet the specific PDE. Now, using also coefficients for each loss function λ_x where x refers to the corresponding loss coefficient, we have the following equation.

$$Loss = Loss_{Physics} + Loss_{Data}$$

$$\begin{aligned} L = &\frac{\lambda_r}{N_r}\sum_{i=1}^{R}\|\mathcal{F}(u_w;\theta)(x_i)\|^2 + \frac{\lambda_i}{N_i}\sum_{i=1}^{I}\|\mathcal{I}(u_w;\theta)(x_i)\|^2 + \\ &\frac{\lambda_b}{N_b}\sum_{i=1}^{B}\|\mathcal{B}(u_w;\theta)(x_i)\|^2 + \frac{\lambda_d}{N_d}\sum_{i=1}^{D}\|u_w(x_i) - u(x_i)\|^2 \end{aligned} \tag{2.7}$$

where r represents the points inside the domain, i the initial points, b the boundary points, d the data points, and $R, I, B, D \in N^*$ that aggregated comprise the training dataset.

It is important to note that the number of points the model encounters during training, whether from data or physics, can vary and is not subject to a strict constraint. This flexibility also applies to the different components of the physics, including boundary, initial, and domain (PDE) points

2.2.2 Physics Informed DeepONet

As a natural scientific outcome, researchers tried to merge the fields of DeepOnet and PINNs, so in 2021 they developed the physics-informed DeepONet, PI-DeepONet [31].

The PI-DeepONet is the intersection of these two fields. The way they accomplished this is by applying the governing equations of each specific system as in the PINNs but by using the architecture of the DeepONet as the surrogate model. First, they introduce the forcing function in the branch net of the model and the system coordinates in the trunk net. As a result, the PI-DeepONet is differentiated through the trunk net, since it holds the independent variables of the system. Their model was tested over five different cases in their publication, but many other researchers have demonstrated the ability of the model to learn from physics in a variety of other scientific problems.

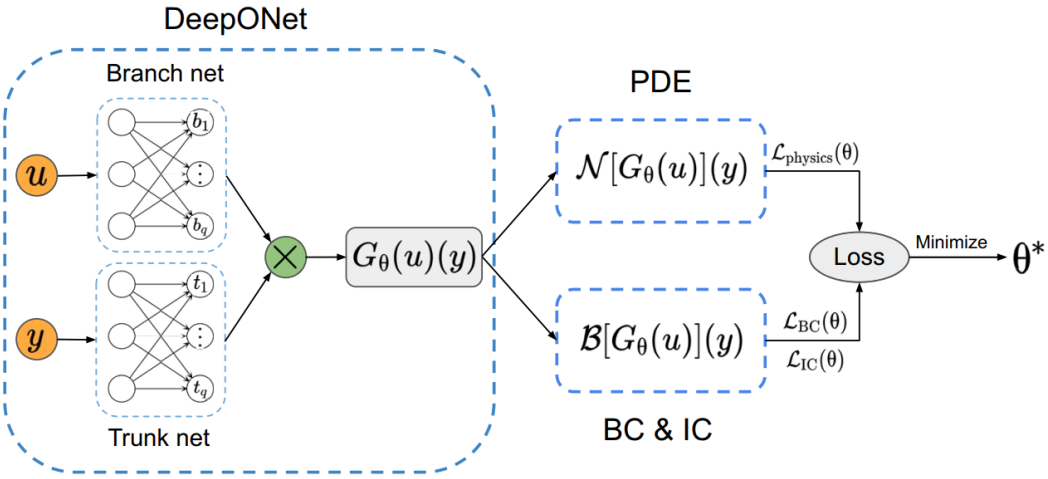


Figure 2.5: Physics-Informed DeepONet Illustrated [31]

2.3 Physics-Informed Operator Learning over PINNs

One major difference between PINNs and physics-informed operator learning is lying on the fact that physics-informed operator learning aims to learn the underlying physics that governs a system over a family of input functions while PINNs are learning each time one input function.

We can see this with a simple example. For this, we will use the heat equation.

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u + f(x,t) \tag{2.8}$$

where α is the thermal diffusivity, u the temperature distribution, t the time, ∇^2 is the laplacian operator and $f(x,t)$ is a source term[3] representing the heat generation or absorption at position x and t[4].

- **PINNs**

In the PINNs case, we would handle the equation 2.8 with passing everything on the right side of the equation and using a deep learning model as a surrogate for solving it. The deep learning model would then receive as input the independent variables, x and t, and output the u. Then, the right-hand side of the equation will be used in the loss function. Nonetheless, the loss function in the PINN doesn't change, so as a result, the source term needs to remain constant. Subsequently, the resulting model will be able to handle only one source term and so learn one unique solution (however the model will be able to extrapolate or interpolate spatiotemporally in the independent variables).

Nevertheless, the most important reason that the PINNs can account for one problem only (here one source term) is that the model receives as inputs only the independent variables and not the source term.

It's important to note, however, that a PINN, with appropriate adjustments to its input, can function as an operator by incorporating a parametric element, such as a forcing function. Nonetheless, using PINNs in this manner is not recommended due to several challenges. They are susceptible to the curse of dimensionality, are more difficult to train, and generally underperform compared to operator networks.

- **Physics Informed Operator learning**

In the operator case (for the reasons of this dissertation, the PI-DeepONet model will be examined only), the workaround would be the same as in the PINNs.

However, in the PI-DeepONet case, the loss function can be variable, but most importantly it will receive as input also the source term as a family of input functions (source term in the branch net, coordinates in the trunknet).

In this way, the trained model instead of dealing each time with one problem only (one unique source term), it will learn the underlying physics for an area spanning from the input functions and so, will predict for multiple source terms. This case provides more benefits for commercial and industry applications as it will be able to produce faster results (forward problem) compared to the expensive simulations (time-consuming and a domain expert need).

This practical implication is where this thesis inspires motivation to study operator learning techniques.

2.4 SciML challenges

In light of these theoretical advancements, numerous researchers have demonstrated various ways in which scientific machine learning (SciML) can make significant contribu-

[3] Usually referred also as forcing term

[4] The initial and boundary conditions have been omitted for simplicity reasons but their presence is implied as otherwise the problem cannot be well posed

tions. One key advantage is that SciML can be orders of magnitude faster compared to conventional PDE solvers [9]. Furthermore, physics-informed modeling is robust to imperfect (noisy) data and exhibits strong generalizability, even in small or zero data regimes. Additionally, SciML can provide capabilities for uncertainty quantification and the ability to address high-dimensional problems [8].

Nevertheless, scientific machine learning despite the fast development of the last years, still has many limitations in the practical implementation. These limitations are centered around the multimodal training, the neural network's capabilities, architecture, and the physics itself.
In the following, the challenges encountered during this thesis will be discussed in more detail.

In the realm of physics, some systems manifest complex phenomena like multiscale and chaotic behavior (turbulence), a behavior that is difficult to capture. On top of that, it has been shown in the past that the neural networks are struggling with learning high frequencies[20, 33] and they prefer the low ones, a problem that is called spectral bias (this is one of the reasons that neural networks were successful in computer vision). To address this issue, researchers have tried to use periodic functions as activation functions. This range of where these applications found success in overcoming the spectral bias span from computer vision, where the researchers applied the sinus function [23] or Fourier feature mapping before an MLP [27], reinforcement learning, where the researchers applied the sinus activation function only in the first layer [34] and of course the PINNs, where the researches showed that the sinusoidal function only in the first layer can provide better results [31] (it has to be noted that other activation functions have been used as well in the first layer such as the exponential).

The second limitation is correlated with the training of the model where the loss components of the objective function display differences in the convergence speeds (also the coefficients that are applied in the components of the objective function in PINNs are most frequently sub-optimal). This limitation has been proved also theoretically with the neural tangent kernels [32], where it was shown also that PINNs suffer from spectral bias, and the researchers suggested a novel algorithm to balance the convergence rate of the loss function. Similar approaches, for balancing the weights of the loss function have been adapted also in this publication [30] where the researchers suggested a new DNN architecture for better accuracy. Similar approaches have been used also in the broader field of multimodal training [4].

Other weaknesses of SciML lie in the locations of the collocation points [28, 35] or an implicit bias of the PINNs to minimize firstly the PDE residuals, before even minimizing the initial data resulting in a model that fails to respect causality [29].

3. ODE CASE STUDY (FORCED OSCILLATOR)

In classical mechanics, an oscillator is a second-order ODEs dynamical system that exhibits periodical behavior around a stable equilibrium position. These oscillations are ubiquitous in nature and technology, and are found in a wide range of systems and devices across different fields in domains such as electrical, acoustical and mechanical engineering.

In this chapter, we will analyze a 2nd order ODEs system as a pedagogical example with a PI-DeepONet.

3.1 Problem Description

One of the most characteristic examples of oscillators are the springs, where a mass is suspended back and forth. In real applications, friction or damping is existent and acts upon this as a frictional force that is resistant to the movement of the mass and proportional to its velocity, causing it to slow down. Such system's governing equation[1] is provided below:

$$m\frac{d^2s}{dt^2} + c\frac{ds}{dt} + ks = U(t) \tag{3.1}$$

where:

s is the displacement,
m is the mass of the oscillator,
c is the viscous damping coefficient,
k is the spring constant,
$U(t)$ is the external forcing term

In the equation above, the $U(t)$ is an external force applied to the system and dictates it on how will it move. The behavior of the forced oscillator system then is governed by the interplay between the natural frequency of the system and the frequency and amplitude of the external force.

The equation described above will be the subject of interest for this chapter.

Below there are given the characteristic values of the system that will be analyzed :

$$\begin{aligned} m &= 0.5, \\ c &= 1, \\ k &= 8 \end{aligned} \tag{3.2}$$

Furthermore, for the system to have a unique solution each time, the initial states of it need to be provided. For the current problem, the initial condition will be:

[1] which is essentially the second law of Newton

$$\frac{\partial s_0}{\partial t} = 0,$$
$$s_0 = 0 \tag{3.3}$$

where s_0 corresponds to the initial condition $t = 0$.

The generation of $U(t)$ will be described in the section 3.2 where the dataset generation will be explained. Different evaluations of the $U(t)$ will produce each time different set of displacements s.

3.2 Dataset

In the 2.1.2, the formation of a DeepONet dataset was described. There, the required dataset for a DeepONet was defined as a triplet of forcing term (discretized in the coordinated points), the coordinates, and the exact solution.

To create this dataset, since we don't use experimental data, we need to generate them artificially.

For this reason, for the forcing term, we employ a Gaussian random regression function with a radial basis function as its kernel and we sample randomly data from there. The hyperparameters that are provided for this kernel are: l and σ. In our problem, we had:

$$l = 0.4$$
$$\sigma^2 = 1$$

where:

- σ^2 is the variance parameter, controlling the overall scale of the function,
- l is the length scale parameter, controlling the smoothness of the function. Higher values translate to smoother function, while smaller values to more chaotic behavior

The point coordinates interval where the oscillator is evaluated is $t \in [0, 1]$ and they represent temporal points.

As for the exact solution, it is obtained from a Runge-Kutta 45[2] scheme.

When the model is trained completely with physics the exact solution isn't needed for the training. The exact solution will be used only for evaluating the accuracy of our result.

Below it will be visualized how the forcing term and the exact solution for $l = 0.4$ and $l = 0.1$ is looking like.

[2]Runge-Kutta is a numerical scheme method that solves iteratively ODEs. They are considered a cornerstone method in numerical methods and are widely used today

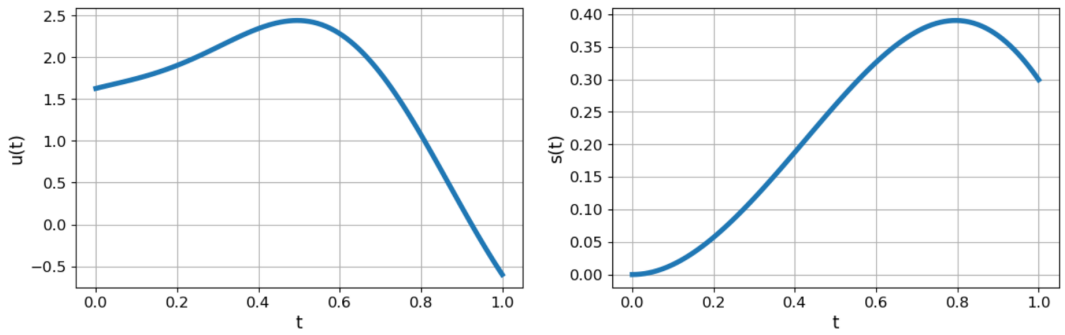


Figure 3.1: Oscillator with a forcing term $l = 0.4$ and its exact solution

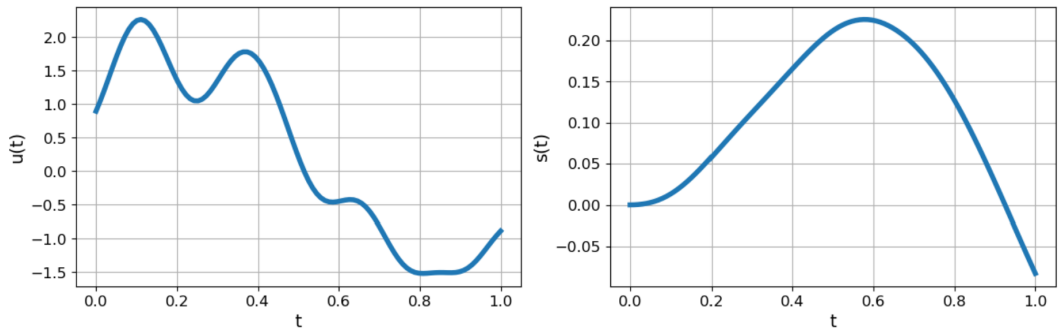


Figure 3.2: Oscillator with a forcing term $l = 0.1$ and its exact solution

From the above plots, the effect of the length scale on the generated forcing term is evident.

3.2.1 Dataset Composition

In consideration of the preceding, we will now proceed to present the composition of the dataset.

For the dataset, 100 sensors equispaced were used for the forcing term U. This number of sensors was considered adequate to describe efficiently the forcing term. Next, 2000 realizations of the forcing term were included in the training of the model and 100 were used for evaluation and another 100 for testing[3].

For the evaluation of the output function were used also 100 points with the interval of them to be equispaced at $t \in [0, 1]$. This number of points, despite that, coincides with the number and location of sensors of the forcing term, it doesn't have to be the same and it can be arbitrary.

For the third element of the triplet, an exact solution s was used that is assessed on the output location point t. In the present case, the result of the exact solution is a scalar value.

So the dataset consists of 200000 triplets[4] (2000×100).

[3] The evaluation and testing datasets are using the same length scale unless it is described otherwise

[4] 1st element: a 100 vector of the forcing term, 2nd element: the output evaluation point, 3rd element: the scalar point of the exact solution on the output evaluated point

3.3 Neural network architecture

The forced oscillator problem will be analyzed primarily from the lens of the physics-informed solution.

The implemented architecture was 3 layers of 50 neurons for both trunk and branch[5]. The input of the branch was a 100-dimension vector while the trunk's was a scalar value. As an activation function, the hyperbolic tangent function (tanh) was selected. After performing the elementwise multiplication of the branch and trunk net, bias is also added as in the DeepONet proposed model. The model was then treated as the $s(t)$, so the corresponding differentiations took place to match with the governing equation (3.1) that has the (3.2) variables and the boundaries of it (3.3).

The model training was executed in CPU and the coefficients of the boundary loss and the ODE loss were equally 1. Furthermore, the learning rate of the model was 1e-3 with a reduction of learning rate in the plateaus by half after 30 epochs and saving up of the best model.

Finally, the model was trained for 300 epochs with normalization taking place before the data get into the model and the adam optimizer algorithm was used. Each training lasted for approximately 1 hour and 22 minutes.

3.4 Base model

In this section, it will be presented the model that has been described until this point. To quantify the error of the models, we used the Root Mean Square Error (RMSE).
The mathematical formula for RMSE is provided below.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(s_i - \hat{s}_i)^2} \tag{3.4}$$

where:

$\hat{s}$ is the predicted value,
s is the actual value,
n is the total number of observations

The training of the model is shown below along with the corresponding graphs that exhibit its performance.

[5]except if another way is mentioned

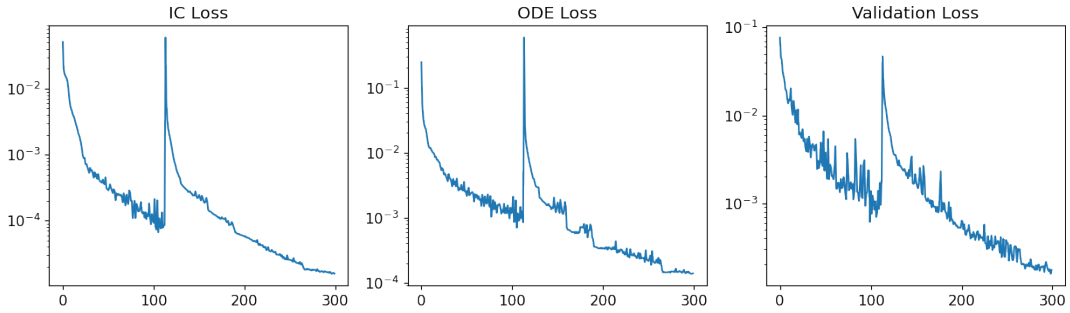


Figure 3.3: Training loss of the base PI-DeepONet with $l = 0.4$

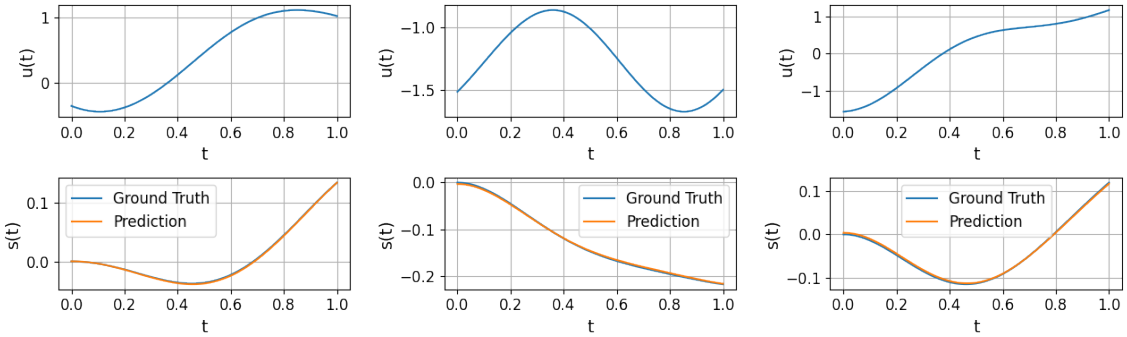


Figure 3.4: Prediction results of the base PI-DeepONet for $l = 0.4$ (interpolation)

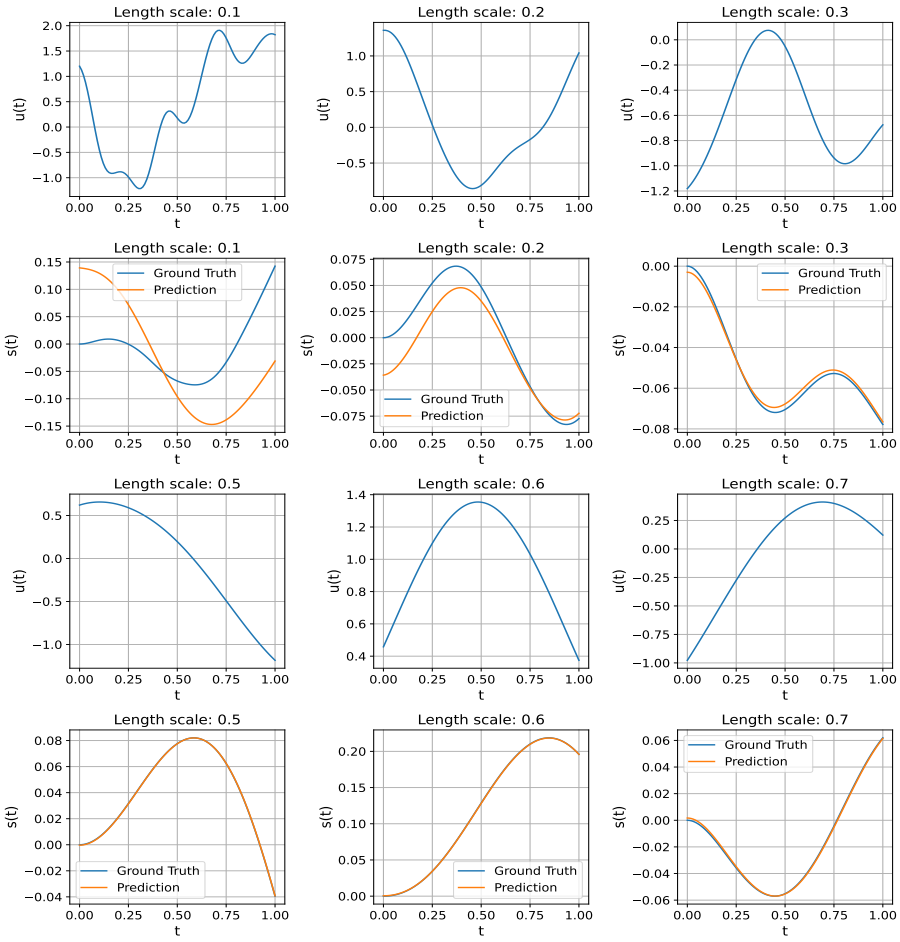


Figure 3.5: Prediction results of the base PI-DeepONet trained on $l = 0.4$ to the whole spectrum of the examined length scales except $l = 0.4$

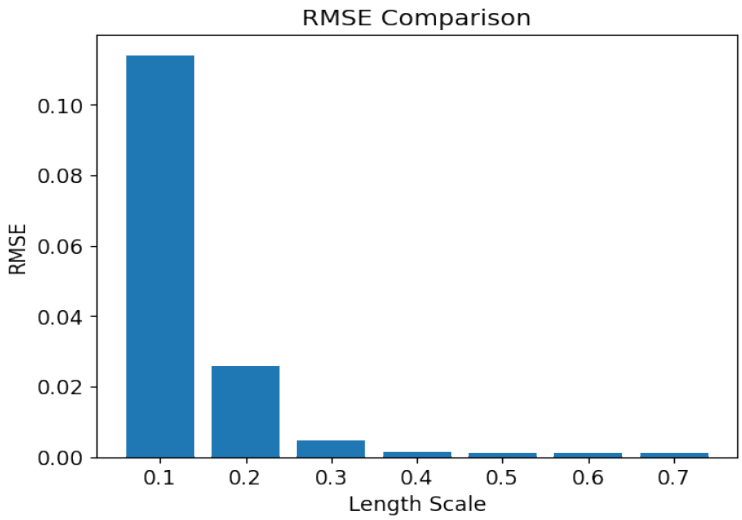


Figure 3.6: Prediction score capabilities of PI-DeepONet for various length scales

From Figures 3.4 and 3.6, we observe how effectively the model learns and generalizes to unseen input functions within the trained length scale. This demonstrates, as discussed in Section 2, PI-DeepONet's capability to handle a family of input functions, unlike PINNs, which struggle with this task. Additionally, we see that the model generalizes well to higher length scale input functions (smoother functions).
However, the analysis also reveals a limitation in the model's ability to extrapolate accurately, particularly for lower length scales (higher frequency transitions).

Given this perspective, introducing an additional layer in the trunk network to capture nonlinearities may offer potential improvements.

3.5 Trunk Net sinusoidal layer

Within this section, an augmentation of the trunk network is introduced through the incorporation of an additional layer. This supplementary layer comprises 50 neurons, with weights and biases and a sinusoidal function serving as its activation function. This methodology, known in the literature as Fourier Feature Embeddings, can be seen as the representation of both cosine and sine functions or solely one of them. In our particular implementation, the inclusion of a sinusoidal layer alone sufficed to yield improvements [6].

Subsequently, the forthcoming step involves generating prediction graphs in a manner similar to those presented earlier.

[6]Notably, the utilization of both cosine and sine functions did not enhance performance; instead, it led to overfitting of the model

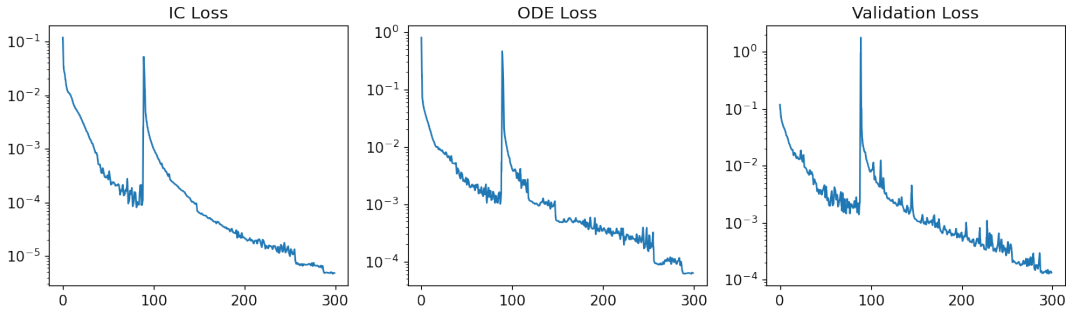


Figure 3.7: Training loss of the base PI-DeepONet with sinus activation layer with $l = 0.4$

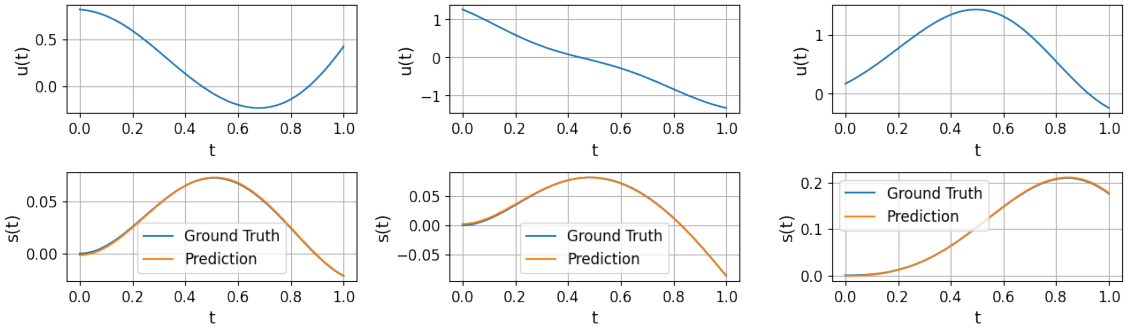


Figure 3.8: Prediction results of the base PI-DeepONet with sinus activation layer for $l = 0.4$ (interpolation)

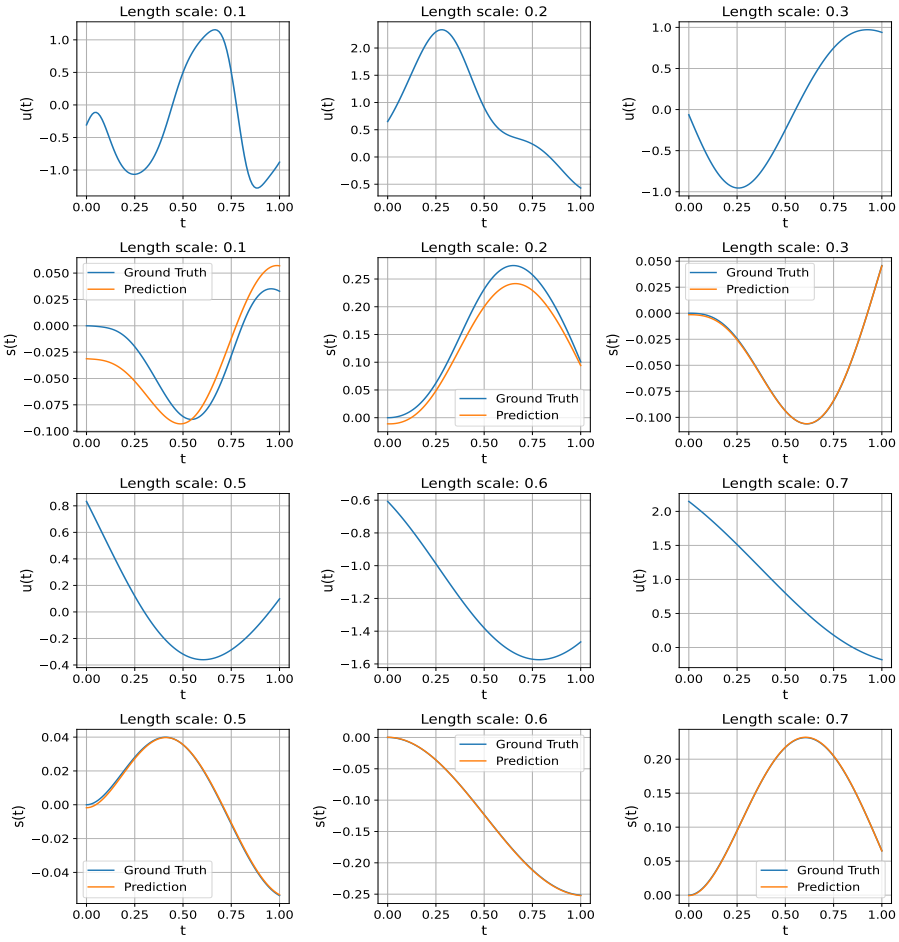
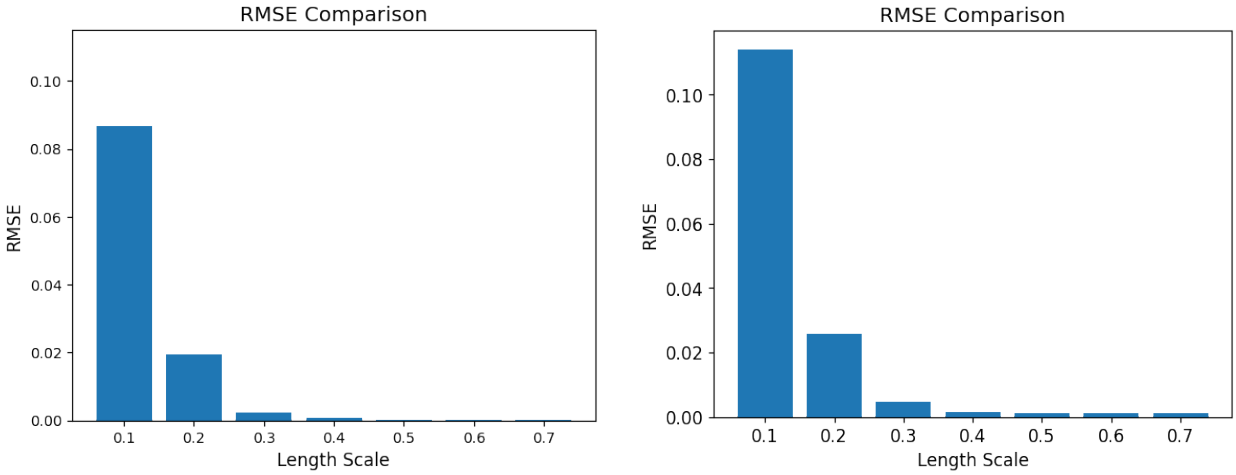


Figure 3.9: Prediction results of the base PI-DeepONet with sinus activation layer for all the length scales (extrapolation) except the trained $l = 0.4$



(a) Prediction score capabilities of PI-DeepONet with sinus activation layer for various length scales

(b) Figure 3.6 for comparison

Figure 3.10: Model generalizability comparison for trained length scale $l = 0.4$

Based on the above plots, it is evident that the utilization of the sinusoidal function results in enhanced prediction performance, facilitating improved representation of both the trained

length scale and higher length scales. Notably, it demonstrates superior extrapolation abilities to lower length scale values, with a marked improvement observed.

This reaffirms the significance of employing the sinusoidal function in accommodating high-frequency input functions.

3.6 Lower-length scale training

Both sections 3.4 and 3.5 have made apparent that the model demonstrates superior extrapolation capabilities when confronted with smoother functions compared to more intricate ones. This prompts the inquiry regarding the optimal approach for training the model: whether employing high-frequency input artificially generated data on the physical equations, as explored in this dissertation, would yield the most advantageous results for developing a generalized predictive model capable of accommodating a wider range of input functions.

Consequently, the ensuing investigation entails training a model on lower length scales and evaluating its predictive performance on smoother input functions, including the length scale for which it was trained.

Both the architectures in 3.4 and 3.5 will be examined.

3.6.1 Base model

Firstly, we will examine the base architecture without the sinusoidal layer and we will draw the same graphs as previously.

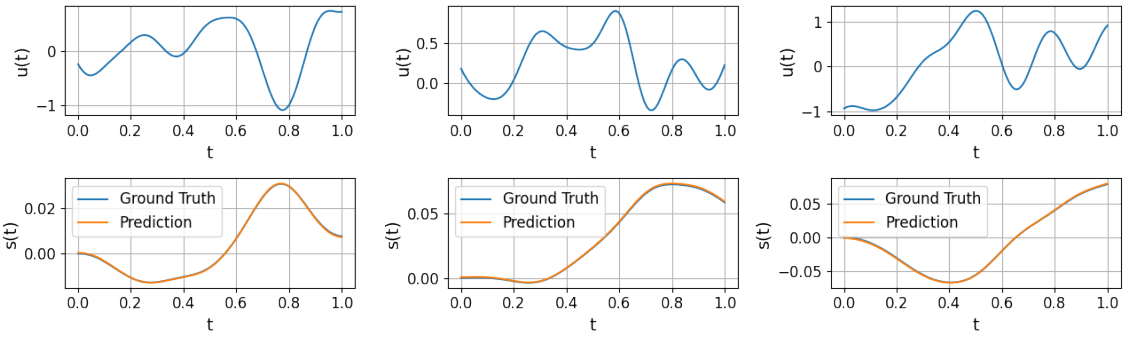


Figure 3.11: Prediction results of the base PI-DeepONet with being trained at $l = 0.1$ for $l = 0.1$ (interpolation)

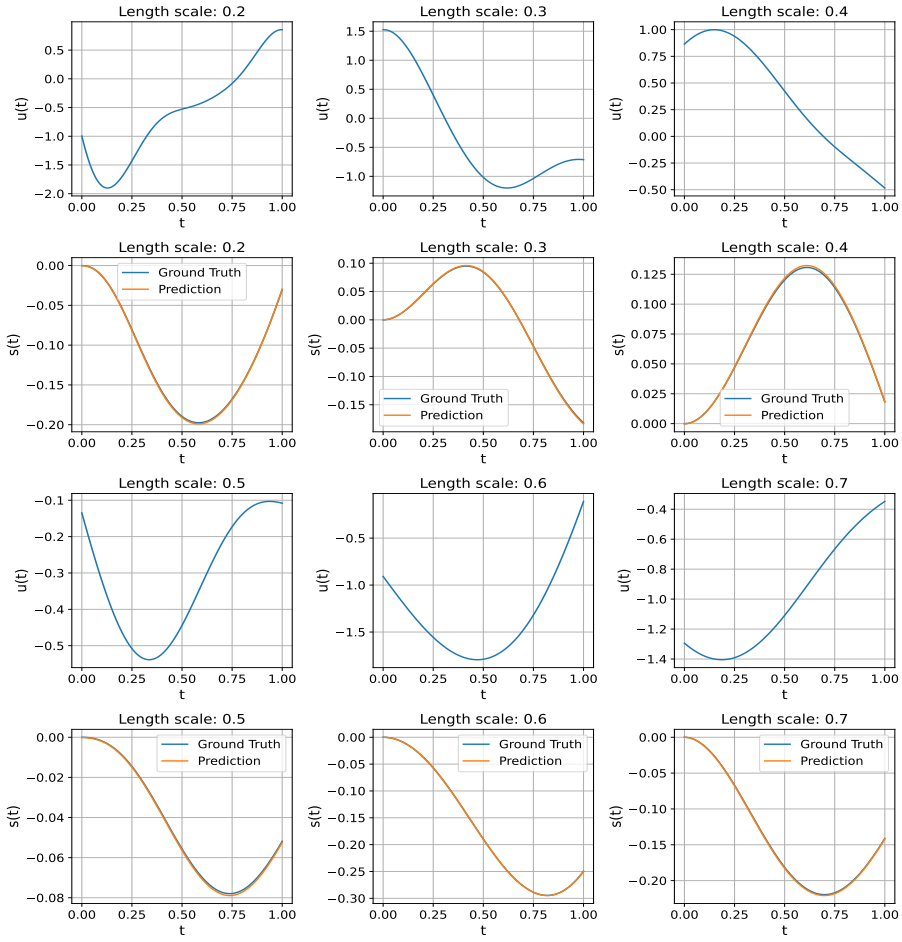


Figure 3.12: Prediction results of the base PI-DeepONet with being trained at $l = 0.1$ for $l \in [0.2, 0.7]$ (extrapolation)

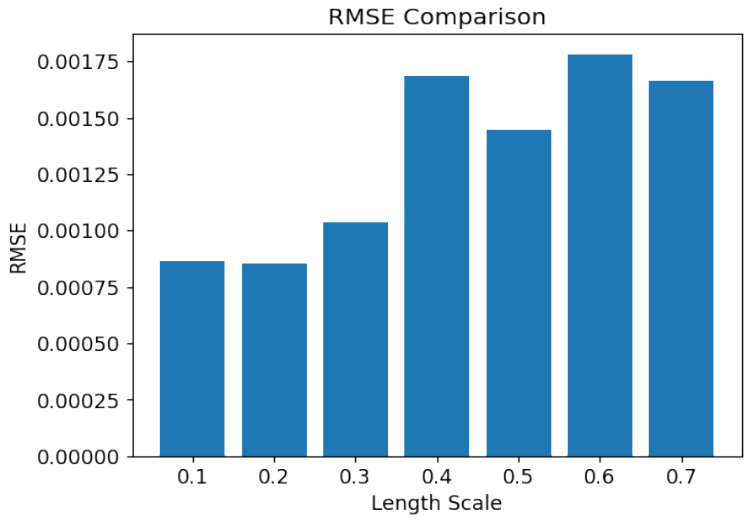


Figure 3.13: Prediction score capabilities of PI-DeepONet trained with $l = 0.1$ for various length scales

It is remarkable to observe that the model effectively captures the low-length scale for which it was trained and, significantly, demonstrates the ability to extrapolate to a broader range. An unexpected yet crucial finding is that the model achieves lower scores in the extrapolation process.

This indicates the potential benefits of training physics-informed models on more complex training datasets.

3.6.2 Trunk Net sinusoidal layer

The methodology employed in 3.5 will be replicated, with the distinction that $l = 0.1$ will be utilized.

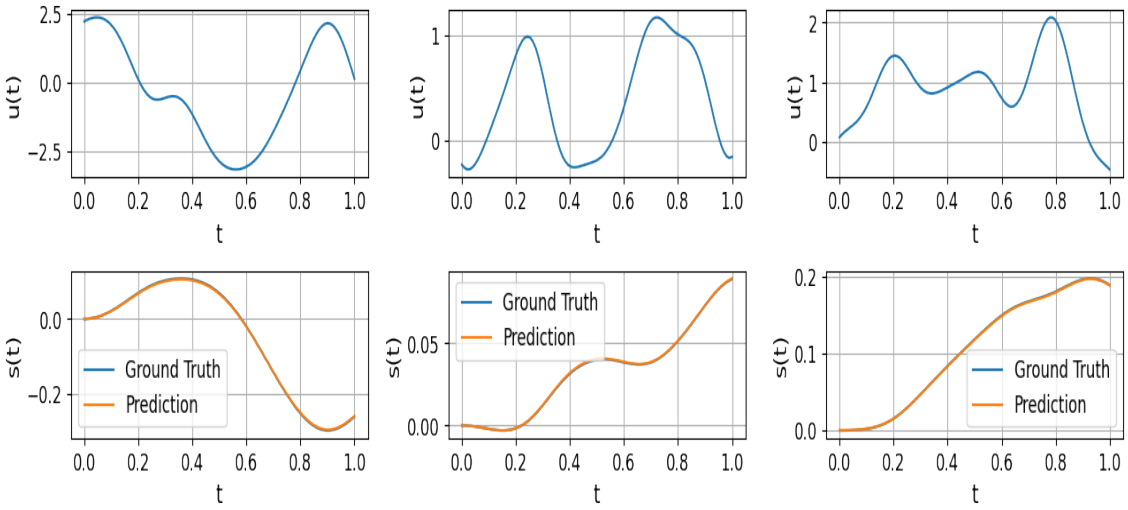


Figure 3.14: Prediction results of the base PI-DeepONet with sinus activation layer trained with $l = 0.1$ for $l = 0.1$ (interpolation)

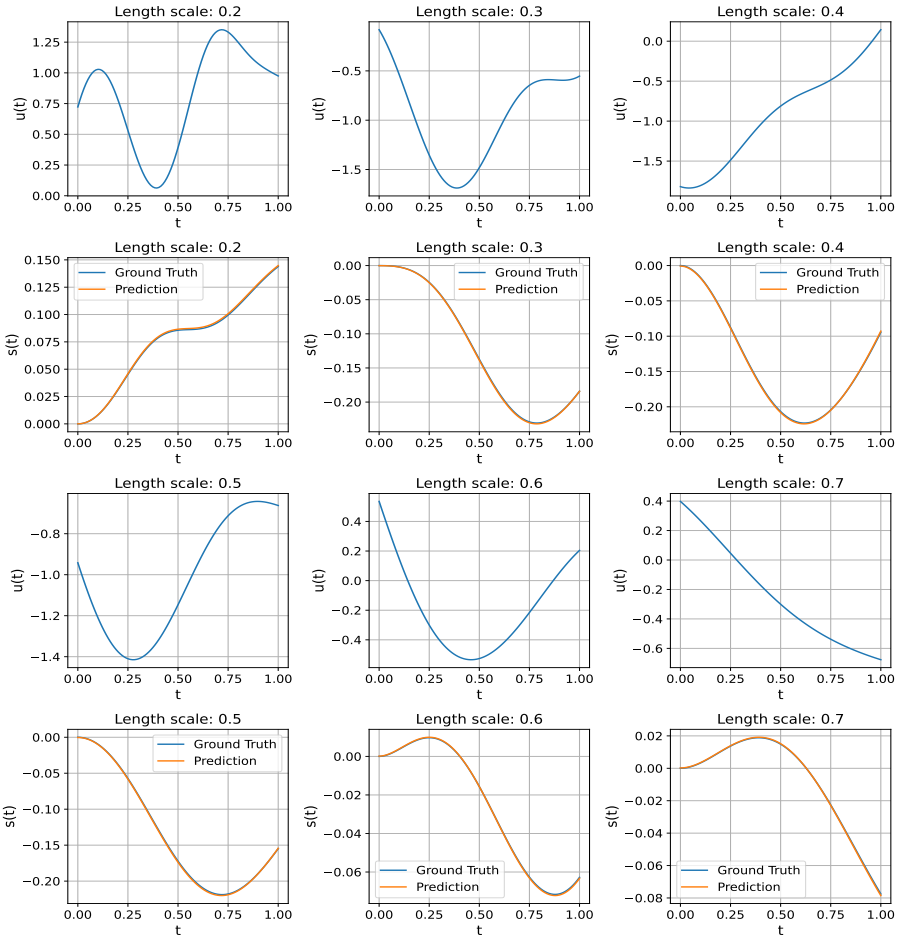
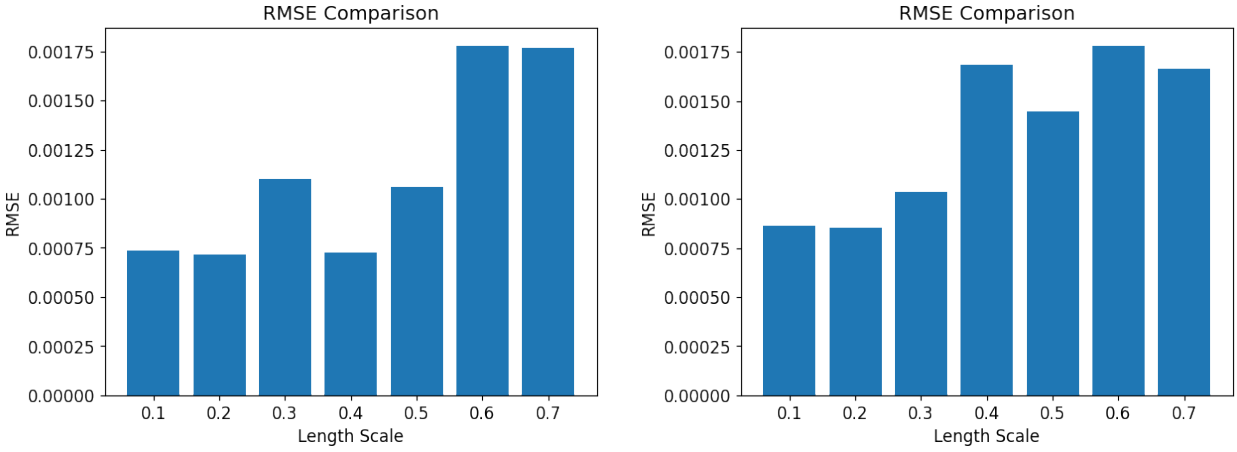


Figure 3.15: Prediction results of the PI-DeepONet with a sinus activation function in the trunk net and being trained at $l = 0.1$ for $l \in [0.2, 0.7]$ (extrapolation)



(a) Prediction score capabilities of PI-DeepONet with sinus activation layer trained with $l = 0.1$ for various length scales

(b) Figure 3.13 for comparison

Figure 3.16: Model generalizability comparison for trained length scale $l = 0.1$

The outcomes presented herein demonstrate comparability to those of 3.6.1. Notably, the sinusoidal function does not appear to confer any discernable benefits in this instance.

3.7 Comparison between hybrid, data, and only physics will be displayed

This section entails a comparative analysis among models trained solely on physics, solely on data, and a hybrid-trained DeepONet. The comparison will focus on training with a length scale of $l = 0.1$, as it has previously demonstrated superior performance.

In the hybrid training approach, the coefficients of one will be employed for both the physical loss (comprising initial conditions and the ODE) and the data loss.

A summary of these comparisons is provided in Table 3.1.

Table 3.1: RMSE comparison for physics only, data only, and hybrid trained with $l = 0.1$ models with and without sinusoidal pre-activation layer [7]

	without sinus		with sinus	
	$l = 0.1$ *(interpolation)*	$l \in [0.2, 0.7]$ *(extrapolation)*	$l = 0.1$ *(interpolation)*	$l \in [0.2, 0.7]$ *(extrapolation)*
hybrid	0.623e-03	1.0783e-03	0.5143e-03	0.699e-03
data only	1.526e-03	2.799e-03	1.4969e-03	2.2238e-03
physics only	0.823e-03	1.1835e-03	0.6856e-03	0.9471e-03

As anticipated, the hybrid approach model exhibits the lowest Root Mean Square Error (RMSE) due to its amalgamation of both physical and data training, rendering it more robust. Several noteworthy observations emerge:

- The hybrid approach achieves the best performance across both interpolation and extrapolation scenarios, with and without the sinus preactivation layer
- The physics-only approach exhibits lower errors compared to the data-only approach
- The data-only approach performs the worst across all scenarios
- The inclusion of the sinus preactivation layer consistently yields lower RMSE values across all cases
- The physics-only approach yields a lower prediction discrepancy error between the interpolation and extrapolation compared to the data-only case, thus showcasing the enhanced generalizability that accompanies a physics-trained solution

The aforementioned observations can likely be attributed to several factors. Firstly, the data-only approach is trained faster, yet it often reaches a plateau, resulting in stagnation. Conversely, the physics-only training paradigm is inherently more intricate to learn, requiring typically numerous epochs to be trained, as it has been noted in the literature and our case, and often necessitating specialized approaches(which has prompted the rapid development of PINNs in recent years). Yet, the integration of physics offer stronger generalizability.

[7] In the extrapolation column, 100 profiles were generated and evaluated for each of the length scales [0.2, 0.3, …, 0.7], which were then averaged. Similarly, for the interpolation column with $l = 0.1$, 100 profiles were assessed

Data-driven approaches need large datasets to effectively capture the underlying physics of a problem; otherwise, they may underperform compared to physics-based approaches. In this study, the data-driven approach was provided with 2000 input profiles, each discretized into 100 points (sensors). This dataset size may contribute to the poor performance of the data-only approach compared to the hybrid and physics-only approaches. Another cause that could potential result in lower performance might be the use of not adequate sensors per profile to capture the intensified fluctuations created from the high frequency inputs.

In the hybrid approach, the model benefits from the ability to learn and converge more rapidly due to the inclusion of data-driven training, while simultaneously leveraging the physics-based constraints to further refine its performance over time. Meanwhile, the superior extrapolation capabilities observed in the physics-informed training stem from the training weights that are getting meaningful representations due to the physical equations integration, which contrasts with the data-only approach.

Lastly, the incorporation of a sinusoidal preactivation layer adds further value to the model. This enhancement may stem from two primary reasons: firstly, it facilitates the modeling of oscillatory behavior inherent in physical systems, which often manifests as wave shape behavior; secondly, the sinusoidal layer aids inputs with near values to expand across a wider spectrum, thereby addressing non-linearities potentially induced by high-frequency values.

The foregoing discussion underscores the considerable significance of incorporating physics into engineering problems that are handled with machine learning. This inclusion proves to be pivotal, not only for enhancing the interpretability of the model but also for improving its performance and generalizability.

4. PDE CASE STUDY (BATTERY CASE)

In this chapter, an examination of a realistic PDE problem will be conducted for estimating battery response. Despite the increasing academic and industrial interest in battery modeling with neural networks, and the ongoing and completed research efforts, there remains a scarcity of applications from the operator learning networks perspective. Particularly those employing solely physics-informed learning into the operator network there hasn't been conducted so far to the best of the author's knowledge.

4.1 Introduction

The rapidly evolving landscape of electrification has brought a surge in energy storage systems that power a variety of modern society applications from small devices (cordless phones) to transportation systems (electric cars) to large-scale systems (renewable energy). At the core of this upsurge lie lithium-ion batteries (LIBs), which play a central role because of their high energy density, efficiency and their relatively low cost compared to other conventional batteries.

As batteries have become a keystone for so many modern technological advancements and the demand for them continues to grow, researchers and manufacturers are keenly aware that further development and improvement of advanced battery technologies become increasingly critical. Ongoing research endeavors have aimed so far to enhance battery performance, cost-effectiveness, and also to prolong their lifespan.

In all the above domains of improvement, the key is to monitor the battery which is essential for predicting its state. This capability comes from the battery management systems (BMSs) that utilize a set of equations either empirical or analytical. BMSs are becoming viable by using a set of equations, either empirical or physics-based, which express the battery behavior. These systems are ubiquitous in most of the batteries whichever their scale (from Wh to MWh).

Empirical models are based on adjusting equations incrementally and fitting accordingly equations and parameters to experimental observable data, while physics-based models are based on the equations that govern the batteries and can shed light into the internal mechanisms of the battery [18]. In general, the development of a comprehensive model capable of describing all dynamic processes across multiple scales proves to be unattainable, necessitating the customization of models to suit specific applications. Physics-based models have demonstrated considerable efficacy in characterizing battery cell behavior; however, their inherent complexity and computational demands render their integration into real-time applications unfeasible.

Henceforth, an array of models has been formulated to accommodate diverse complexities, ranging from microscale models to pseudo-two-dimensional models (P2D) such as Doyle–Fuller–Newman (DFN) and Single Particle Models (SPM). Notwithstanding the range of complexities among existing models, the inherent CPU-intensive nature persists, constraining their integration within Battery Management Systems (BMSs). As a prospective alternative for BMSs, empirical models may offer a lighter computational burden; however, their reliability is compromised by the absence of governing physical equations, rendering them less reliable and incapable of accurately representing intermittent battery variables such as concentration.

As a consequence, neural networks that are designed for modeling physical problems are rising as a promising alternative to combat all the prior weaknesses, while delivering fast results. Therefore, corresponding neural networks will be examined for dealing with batteries.

4.2 Single Particle Model

Physics-based models for batteries were initially developed in the 1960s. Among these, the most widely used category are the P2D models, which require a plethora of parameters, but when properly calibrated they can provide accurate results with the most computationally expensive yet accurate being the DFN model. The development of various simplified models that stem from reductions of the more complicated models has led to the emergence of a P2D category of models known as SPMs.

SPMs posit that the entirety of particles within an electrode can be represented by a singular spherical particle[1], thus substantially streamlining the model's intricacy [18]. The fundamental concept underlying SPMs is that the behavior exhibited by particles within each electrode is highly homogeneous. This comes under the assumption that all electrode particles in anode and cathode, delithiate and lithiate respectively, with the same rate irrespective of their position in the positive and negative electrode. Consequently, they can be collectively approximated by a singular representative (or average) particle, hence the nomenclature. Moreover, these physics-based electrochemical models afford insight into the behavior of multiple internal variables which are not easily observable through *in operando* experimentation.

Throughout the last years, many reduced order models have been derived which are similar but distinct, hence having made the navigation in the literature intimidating. Single Particle models are divided into two subcategories, SPMe which are models that include electrolyte dynamics, and SPM which are not.

An illustration of an SPM model is shown and a brief explanation of it is following:

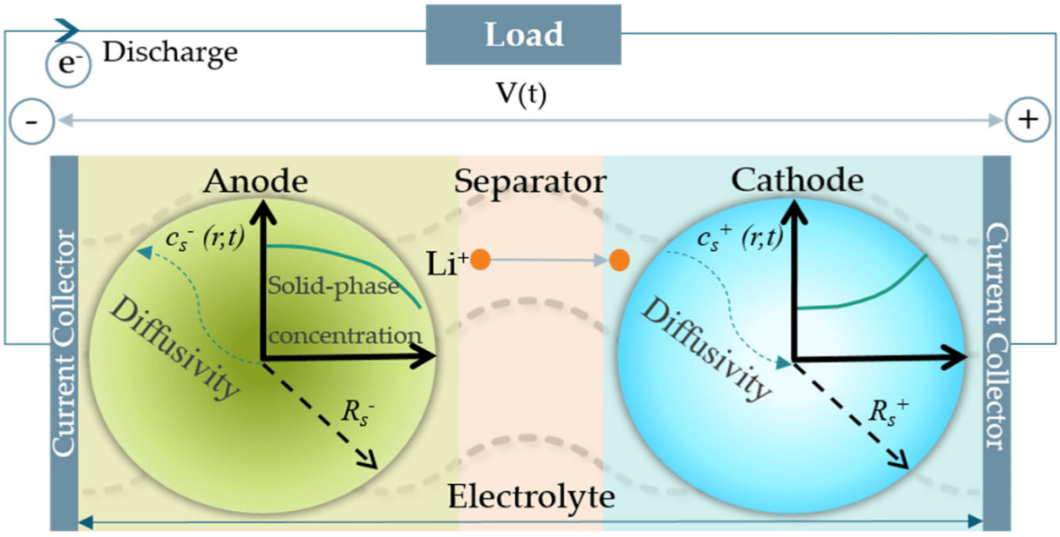


Figure 4.1: Illustration of an SPM model from [22]

[1]Opposed to the microscale models which consider the microstructure of the battery and accounts for the individual shapes of the electrodes or for the DFN that consider the microstructure of the battery homogenized and each electrode as shperical

In the figure 4.1 is shown the SPM battery structure. Anode (negative electrode) and cathode (positive electrode) are represented with spherical particles and the intercalation is modeled over spherical coordinates. This results in a PDE that has only one dimension over the spacial coordinates (contrary to DFN which has two) and one temporal coordinate. Over a charging or discharging session lithium ions (Li-ions) are transferred from one electrode to the other through the electrolyte. Specifically, during discharging Li-ions from the negative electrode are moving towards the positive electrode and conversely over the the charging phase. The behavior of electrode particles are governed by the diffusion law of Fick, resulting in concentration gradients of Li-ions between the particle's center and surface.

The equations that characterize the above system are presented below:

$$\frac{dc_{s,k}(r,t)}{dt} = \frac{D_{s,k}}{r_k^2}\frac{\partial}{\partial r_k}\left(r_k\frac{\partial c_{s,k}(r,t)}{\partial r_k}\right) \tag{4.1a}$$

$$\left.\frac{\partial c_{s,k}(r,t)}{\partial r}\right|_{r_{\mathsf{k}}=0} = 0, \quad \mathsf{k} \in \mathsf{n}, \mathsf{p} \tag{4.1b}$$

$$D_{s,k}\left.\frac{\partial c_{s,k}(r,t)}{\partial r}\right|_{r_{\mathsf{k}}=R_k} = \begin{cases} -\frac{I(t)}{Fa_{\mathsf{n}}L_{\mathsf{n}}}, & \mathsf{k}=\mathsf{n}, \\ \frac{I(t)}{Fa_{\mathsf{p}}L_{\mathsf{p}}}, & \mathsf{k}=\mathsf{p} \end{cases} \tag{4.1c}$$

$$c_{\mathsf{s,k}}(r_{\mathsf{k}},0) = c_{\mathsf{s,k,0}}, \quad \mathsf{k} \in \mathsf{n}, \mathsf{p} \tag{4.1d}$$

Where r is the radius of the electrodes which lies in the interval $r_k \in [0, R_k]$, R_k is the maximum radius of each electrode, t is the time, $c(r,t)$ is the Li-ions concentration, $D_{s,k}$ is the diffusion coefficient, F is the Faraday constant, L is the thickness, a is the specific interfacial area and $I(t)$ the current density[2].
The subscripts s and k are symbolizing the solid phase[3] and the corresponding electrode respectively (n refers to negative and p to positive electrode). In our case, $D_{s,k}$ will be considered constant.

The input in the model is the current $I(t)$. As for the voltage output, many methods have been proposed in the literature to be computed. These methods make use of the already computed variables.

In the equations (4.1), (4.1d) define the initial condition, (4.1b) is boundary condition for the center of the particles and (4.1c) is the boundary condition for the surface of the particles. (4.1a) describe the domain of the problem where $t > 0$ and $r_k \in (0, R_k)$.

The aforementioned variables within the equations, particularly the concentration, can take high numerical values.
Therefore, as neural networks tend to perform optimally in proximity to zero, the equations (4.1) will be converted to a non-dimensional form. This transformation will follow the approach outlined in [24].

To convert the (4.1) to dimensionless we set:

[2] current that passes from surface
[3] in the SPMe model concentration for the electrolyte is also accounted

$$\bar{c} = \frac{c}{c_{k0}}; \bar{r} = \frac{r}{R_k}; \tau = \frac{D}{R_k^2} t \tag{4.2}$$

Where c_{k0} is the initial concentration, R_k is the maximum radius of each electrode, and D is the diffusion coefficient.

Hence, the equation (4.1) can be rewritten as:

$$\frac{\partial \bar{c_k}}{\partial \tau} = \frac{2}{\bar{r_k}} \frac{\partial \bar{c_k}}{\partial \bar{r}} + \frac{\partial^2 \bar{c_k}}{\partial \bar{r_k}^2}, \quad \mathsf{k} \in \mathsf{n}, \mathsf{p} \tag{4.3a}$$

$$\left. \frac{\partial \bar{c}_{s,k}(\bar{r_k}, \tau)}{\partial \bar{r}} \right|_{\bar{r}_\mathsf{k}=0} = 0, \quad \mathsf{k} \in \mathsf{n}, \mathsf{p} \tag{4.3b}$$

$$\left. \frac{\partial \bar{c}_{s,k}(\bar{r}, \tau)}{\partial \bar{r}} \right|_{\bar{r}_\mathsf{k}=1} = \begin{cases} -\frac{I(\tau) R_p}{F L_n D_{s,k} c_{k0} a_k}, & \mathsf{k} = \mathsf{n}, \\ \frac{I(\tau) R_p}{F L_n D_{s,k} c_{k0} a_k}, & \mathsf{k} = \mathsf{p} \end{cases} \tag{4.3c}$$

$$\bar{c}_{\mathsf{s,k}}(\bar{r_\mathsf{k}}, 0) = 1, \quad \mathsf{k} \in \mathsf{n}, \mathsf{p} \tag{4.3d}$$

The battery parameters configuration utilized is derived from Chen et al. [2]. These parameters will be presented only for the negative particle as it is the only particle modeled in the physics-informed scenario.

Table 4.1: Parameters used in the SPM model for the physics-informed case

Parameters	**Value**	**Units**
Farraday constant	96485.33	$\mathsf{C} \times \mathsf{mol}^{-1}$
Electrode height	0.065	m
Electrode width	1.58	m
Negative electrode active material volume fraction	0.75	-
Negative electrode diffusion coefficient	3.3×10^{-14}	$\mathsf{m}^2 \times \mathsf{s}^{-1}$
Negative particle radius	5.86×10^{-6}	m
Negative electrode thickness	8.52×10^{-6}	m
Initial concentration in negative electrode	29866	$\mathsf{mol} \times \mathsf{m}^{-3}$

In this dissertation, an initial attempt will take place to simulate an SPM model through a DeepONet trained exclusively with data and a DeepONet trained solely with physics (PI-DeepONet).
In the latter case, a simplified approach will be taken as a proof of concept focusing solely on modeling the concentration of Li-ions in the electrodes by solving the equations (4.3) for the negative electrode only. Subsequently, we could potentially deduce the voltage as well as obtain some additional insights for the battery, such as state of health (SOH) and state of charge (SOC). However, this is out of the scope of the thesis.

4.3 Relevant Literature

Despite the burgeoning production of batteries in recent years, the volume of Scientific Machine Learning (SciML) research dedicated to batteries remains relatively limited. Initial efforts within the data science research community to model battery behavior predominantly centered on the application of machine learning techniques, notably Support Vector Machines (SVMs), as demonstrated by Alvarez [38], which were utilized for state-of-charge (SOC) prediction. Subsequently, with the advent and blooming of deep learning, neural networks emerged as a versatile tool for harnessing battery dynamics. These methodologies, distinguished by their varied approaches from the groundbreaking work on learning bias through the loss function that stems from Raissi [21], who catalyze a surge in scientific machine learning across diverse engineering disciplines, often referred to as physics-informed neural networks.

These techniques utilize neural networks to integrate physical principles through diverse mechanisms. For instance in Ref. [17] MLPs were employed in a reduced-order model based on Nernst and Butler-Volmer equations to accelerate the predictive capability by reducing the gap between predictions and observations. The MLP network that was used was placed in a strategic position within the empirical equations and their goal was to model battery aging. In Ref. [26] physics informed data were derived from simulation models and fed up to an LSTM with a dynamic sliding window for battery time-variant health prognoses on Li-ions batteries. In Ref. [7] they used experimental physics data from batteries to train an MLP and they proposed a physics-constrained training for each variable of the physics model individually to enhance the model expressibility. In another case, 2-dimensional grid LSTM was used to predict the internal states of a battery through training with experimental data [12].

While these contributions are often categorized as physics-informed neural networks (PINNs), they deviate from Raissi's seminal work by incorporating physics into the learning process through the loss function. Subsequently, a selection of studies will be presented below that utilize this methodology of training neural networks through physics in the loss function. However, it is notable that the extent of such research remains even more limited.

In the work by Cho [6, 5], physics-informed neural networks (PINNs) were employed to estimate the temperature of Li-ion batteries. Cho investigated the influence of various activation functions, including exponential and sinusoidal functions, as prelayers[4] within the network architecture, exploring different combinations and modes of incorporation through multiplication or concatenation. Subsequently, in a following study, Cho augmented a PINN with a Long Short-Term Memory (LSTM) network to enhance temperature prediction accuracy. Additionally, Singh [22] endeavored to estimate electrode's concentrations using a hybrid PINN model, integrating experimental data and physics equations within the loss function, where the physics component consisted of non-dimensional equations. Furthermore, research focusing on operator learning in the context of batteries is even more limited.

Specifically, only three studies have been conducted thus far, with two employing the DeepONet architecture and one utilizing Fourier Neural Operator (FNO) networks. The first study centered on constructing a hybrid DeepONet framework for battery behavior prediction, involving the development of a surrogate model comprising three distinct DeepONets, each modeling individual electrodes, and a top layer amalgamating information from the

[4] In the first layer of the network

electrode-specific DeepONets alongside diffusion coefficients [36]. Later, a second investigation from the same author introduced a multi-input hybrid DeepONet for predicting battery behavior, capable of providing predictions based on arbitrary initial electrode concentrations [37]. Lastly, an FNO model was trained exclusively using experimental data to predict SOC for batteries [10].

4.4 Dataset

Two distinct datasets were generated for the thesis: one for the basic DeepONet and another for the PI-DeepONet. The primary objective of the DeepONet was to establish an operator facilitating the correlation between input current and voltage. Similarly, the goal for the PI-DeepONet remained consistent; however, due to its increased complexity, the approach necessitated disaggregating the model to encompass individual electrodes to incorporate physical principles. Consequently, given the heightened complexity of the PI-DeepONet, the investigation focused solely on the concentration dynamics of the negative electrode (anode).
As a result, the PI-DeepONet study can be regarded as a ***pilot study***.

4.4.1 PyBaMM

To fulfill the requirements of this dissertation, it was essential to construct a dataset. To this end, PyBaMM (Python Battery Mathematical Modelling) was utilized, as documented by Sulzer et al. [25]. PyBaMM is an open-source Python package designed for battery computational analysis, employing numerical methodologies for this purpose. Notably, PyBaMM offers significant flexibility and maintenance, featuring a range of predefined algorithms, solvers, and libraries encompassing diverse experimental configurations, such as [2] which was used in this dissertation. Also, unlike other widely utilized software within the battery research community, such as COMSOL, PyBaMM is free of charge.

As previously indicated, simulations using the Single Particle Model (SPM) with parameters proposed by Chen [2] were executed via PyBaMM.

4.4.2 DeepONet dataset

For the DeepONet, a dataset of triplets was generated using PyBaMM, consisting of the current, temporal coordinates, and voltage, as outlined in Section 2.1.2. Consequently, the input current and voltage were discretized across time coordinates, indicated as $(I^n, t_m^n, G(I^n, t_m^n))$, enabling the DeepONet to forecast voltage within the temporal domain.

Below is the mathematical formulation for the current profiles.

$$I = \begin{cases} 5 \times \left(\frac{-t}{1800} \times b + 1\right), & \text{t} < 1800, \\ 5 \times \left(\left(\frac{t}{1800} - 2\right) \times b + 1\right), & \text{t} \geq 1800 \end{cases} \quad \text{where } b \in (0.05, 0.95) \tag{4.4}$$

b defines the discharge rate and is the variable that varies the profiles.

Below are the current and voltage profiles used for training. These profiles draw inspiration from the work of Zheng et al. [36], which stands as the sole attempt at applying DeepONet to batteries.

Figure 4.2: Input current to the DeepONet

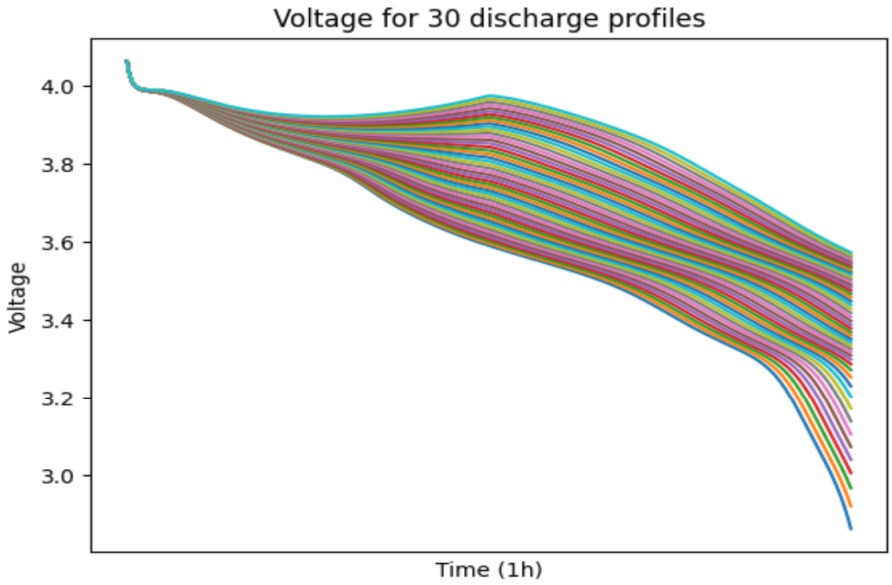


Figure 4.3: Output voltage of the DeepONet

4.4.3 PI-DeepONet dataset

For the PI-DeepONet, the triplet cannot consist of the voltage as output as shown in DeepONet and discussed in Section 4.2; instead, the third element of the triplet represents concentration. Also, contrary to section 4.2, the second element of the triplet consists of two coordinates, radius and time. To simplify the problem and expedite computational processes, only one profile is considered, corresponding to $\tau = 1$, which corresponds to approximately 1040 seconds. While this approach may be considered an extreme case of a PI-DeepONet (or more elaborate neural network architecture of PINN cause of only one input profile), it nonetheless establishes the groundwork for future expansion of the operator network without reliance on experimental data.

The mathematical formulation for the current profile is presented below:

$$I = 5 \times b \times \frac{t}{1800}, \qquad t \leq 1040\ ,\ b = 2 \tag{4.5}$$

Subsequently, the current and concentration profiles of the negative electrode employed for training are presented below.

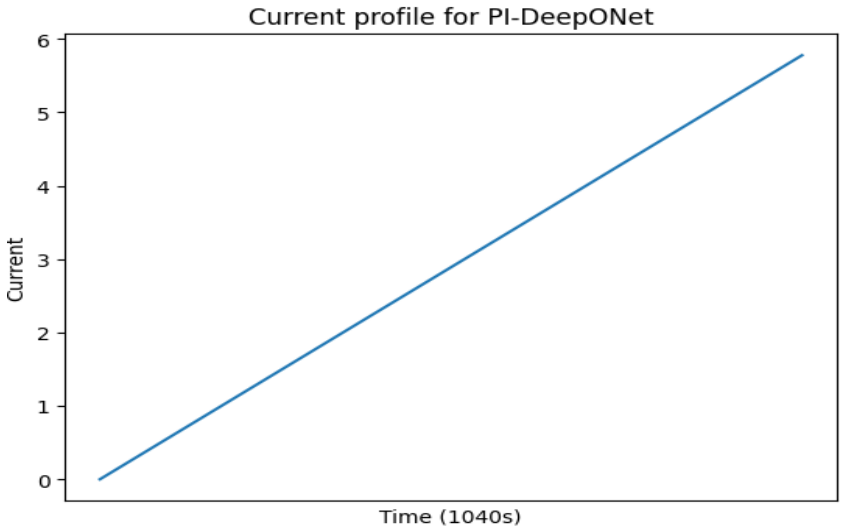


Figure 4.4: Input current [A] to the PI-DeepONet for $\tau = 1$

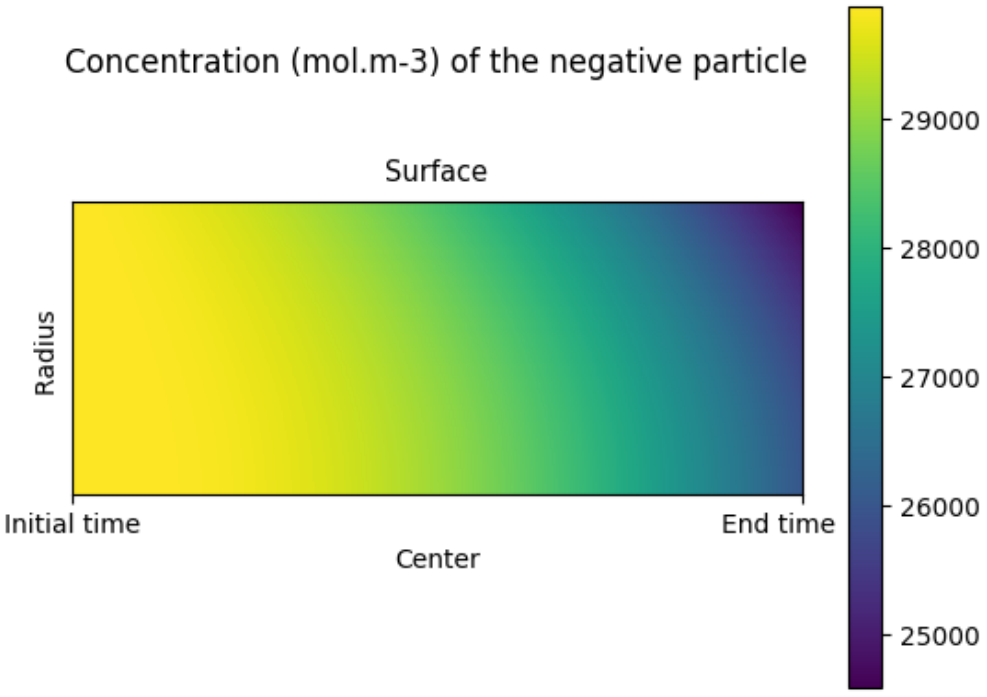


Figure 4.5: Concentration $\left[\frac{\text{mol}}{\text{m}^3}\right]$ for the negative particle for $\tau = 1$

4.5 Neural network architecture

The subsequent section introduces the architectural framework utilized for both DeepONet and PI-DeepONet methodologies. In both instances, Multilayer Perceptrons (MLPs) were

deployed for both trunk and branch networks, with a global application of the hyperbolic tangent function. Furthermore, normalization was applied to the current, spatial, and temporal coordinates. For DeepONet, the branch consisted of four layers, each comprising fifty neurons, while the trunk comprised three layers, each also consisting of fifty neurons. Conversely, the architecture of PI-DeepONet varied across implementations, with a lack of a uniform case. Consequently, detailed descriptions of the architecture utilized in each PI-DeepONet subsection are provided within the respective sections.

All experiments were executed on an Nvidia 2080 Ti GPU with 16GB RAM, utilizing an Ubuntu operating system. Visual representations of both the DeepONet and PI-DeepONet architectures are depicted in the accompanying figures (Fig. 4.6 and 4.7).

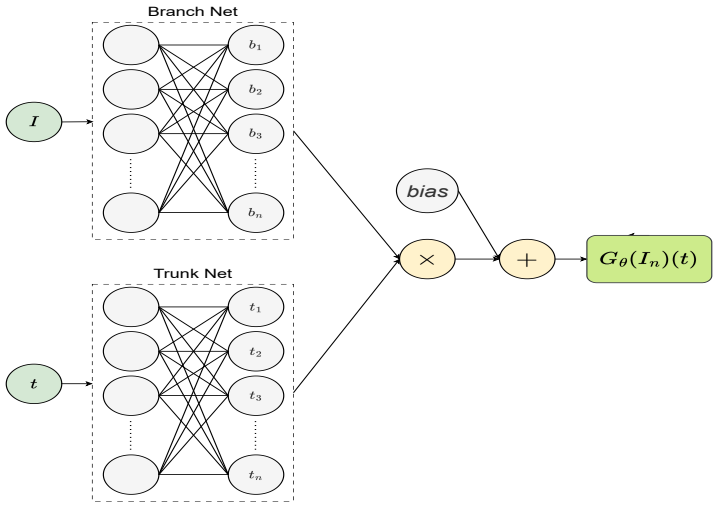


Figure 4.6: DeepOnet illustration for battery

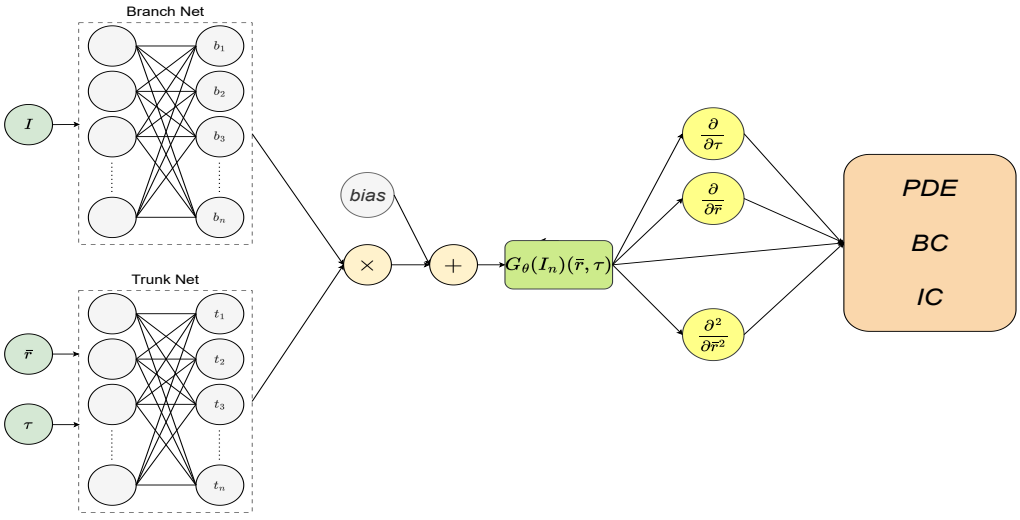


Figure 4.7: Physics informed DeepOnet illustration for battery

4.6 DeepONet Results

In this section, we present the results and observations derived from the implementation of the DeepONet, as previously described. The DeepONet was tested under three distinct

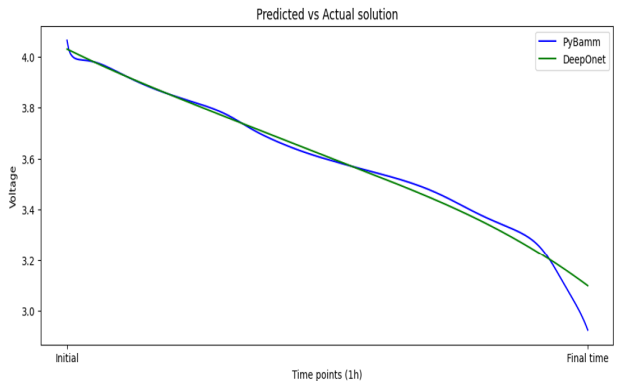
settings for discretizing equispaced the time domain, specifically 300, 100, and 2500 data points[5][6].

Subsequently, a concise summary of the results is provided, accompanied by the corresponding insights.

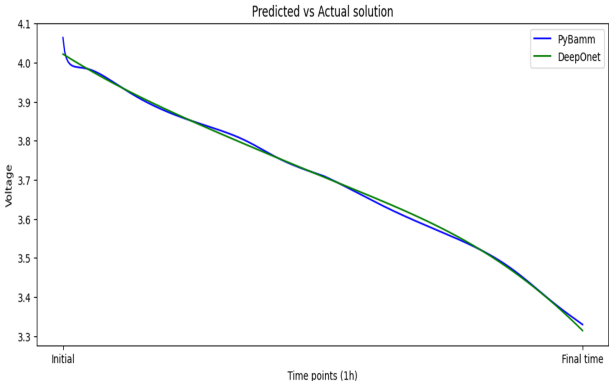
Table 4.2: RMSE for DeepONet testing data

Time Points	300	1000	2500
RMSE	0.1184	0.07713	0.076

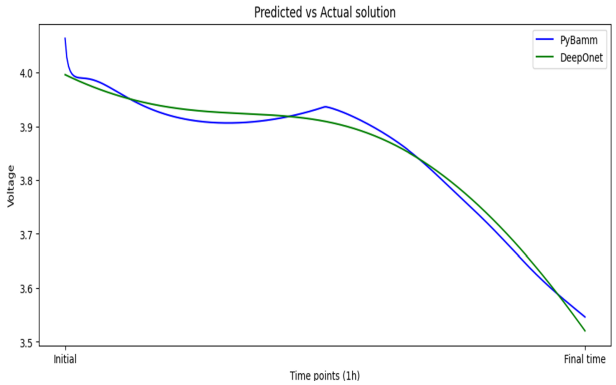
Observations drawn from Table 4.2 indicate that the performance of the DeepONet deteriorates with fewer data points, while improvement is evident as the number of points increases, reaching a threshold beyond which marginal enhancements are observed. Subsequently, an analysis will be conducted to identify instances where DeepONet underperforms under conditions with fewer data points.



(a) Discharge rate $b = 0.1$



(b) Discharge rate $b = 0.4$



(c) Discharge rate $b = 0.9$

Figure 4.8: DeepONet predictions for 300 points with the corresponding discharge rates

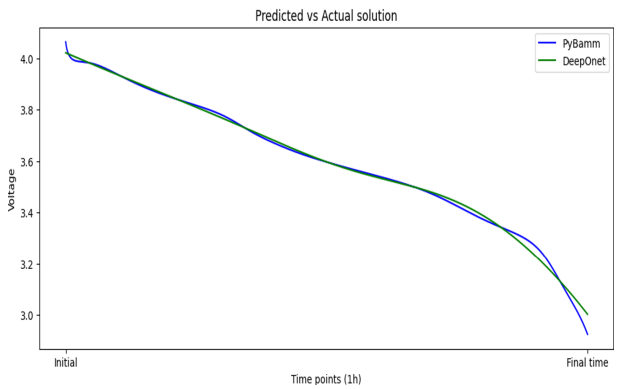
As evidenced by the plots depicted in images 4.8a and 4.8c, DeepONet encounters challenges in regions characterized by rapid variations or non-linear behavior. This phenomenon can be attributed to the abrupt transitions observed, particularly evident towards the end of the time domain in image 4.8a, which may indicate insufficient data capture leading to inadequate representation of dynamic behaviors. However, even in cases where

[5]the forcing function (current) was also discretized with these data points

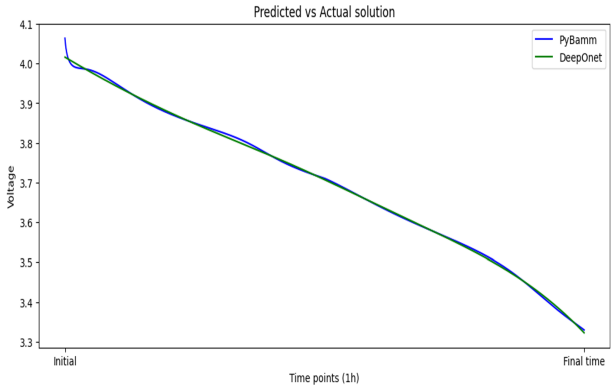
[6]DeepONet was tested also with sinusoidal activation function in the first layer of the trunk with approximately similar results

sufficient data points are ensured, as seen in image 4.8c, the model continues to exhibit limitations in accurately representing the underlying plot dynamics, which can be attributed to an inability to capture the forcing function. Remarkably, DeepONet demonstrates proficiency in capturing relatively linear behaviors, as exemplified by image 4.8b.

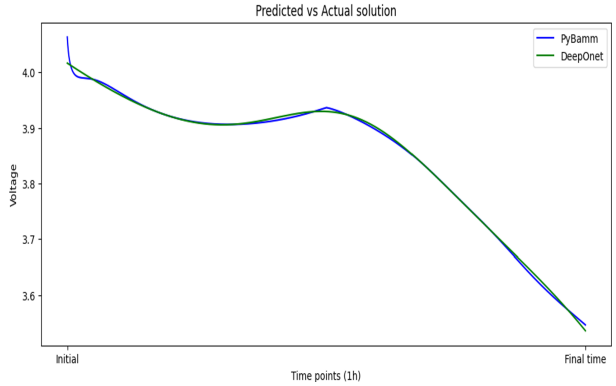
Consequently, to address these shortcomings, an increased number of time points is deemed necessary to effectively capture sudden temporal transitions and abrupt variations in the forcing function.



(a) Discharge rate $b = 0.1$



(b) Discharge rate $b = 0.4$



(c) Discharge rate $b = 0.9$

Figure 4.9: DeepONet predictions for 1000 points with the corresponding discharge rates

Upon examination of Figure 4.9, a marked improvement is evident. Notably, the DeepONet demonstrates proficiency in capturing non-linear dynamics, as illustrated in Image 4.9c, as well as effectively addressing the under-sampled regions evident towards the end of the simulation period, as observed in Image 4.9a.

Consequently, it becomes apparent that augmenting the point density within the time domain yields significant improvements in performance. Having achieved a satisfactory capture of curves with minimal root mean squared error (RMSE), the attention can now be directed toward exploring the capabilities of the PI-DeepONet.

4.7 PI-DeepONet Results

All presented results bear on the negative particle and are obtained at time $\tau = 1$.

Throughout the subsequent analyses, the activation function applied uniformly is the hyperbolic tangent (tanh), unless explicitly stated otherwise.

Additionally, the metrics that were used are: the Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE).

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\text{Actual}_i - \text{Predicted}_i}{\text{Actual}_i}\right| \times 100 \tag{4.6}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\text{Actual}_i - \text{Predicted}_i| \tag{4.7}$$

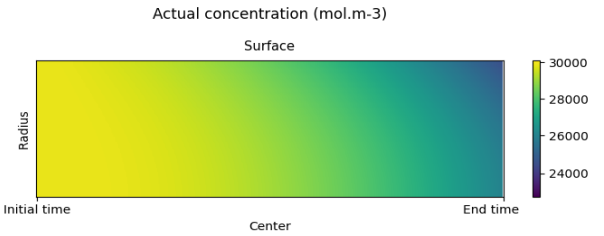
where:

- n is the number of observations,
- Actual_i is the actual value for the i^{th} observation,
- Predicted_i is the predicted value for the i^{th} observation.

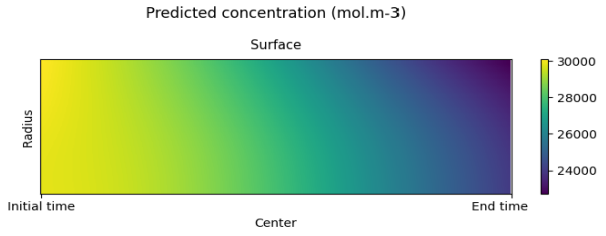
4.7.1 Base architecture

The primary architecture under examination comprises 20 points in the spatial dimension and 300 points in the temporal domain. For the initial condition, 100 points were employed, and for boundary conditions, 300 points were utilized for each side. Regarding the layers employed, two layers, each comprising 100 neurons, were deployed for both the branch and trunk components. Additionally, a layer consisting of 100 neurons was utilized for the branch, and a layer comprising 40 neurons was employed for the trunk as the foundational layers.

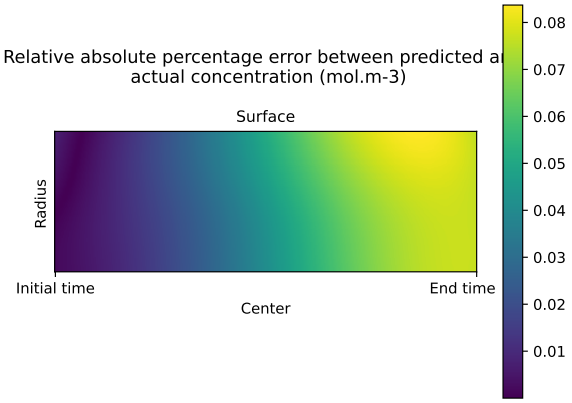
Next, the predicted field is shown alongside the actual field in Figure 4.10, with a detailed examination of the initial and boundary conditions depicted in Figure 4.11, that were described in Equations 4.3.



(a) Actual concentration $\left[\frac{\text{mol}}{\text{m}^3}\right]$

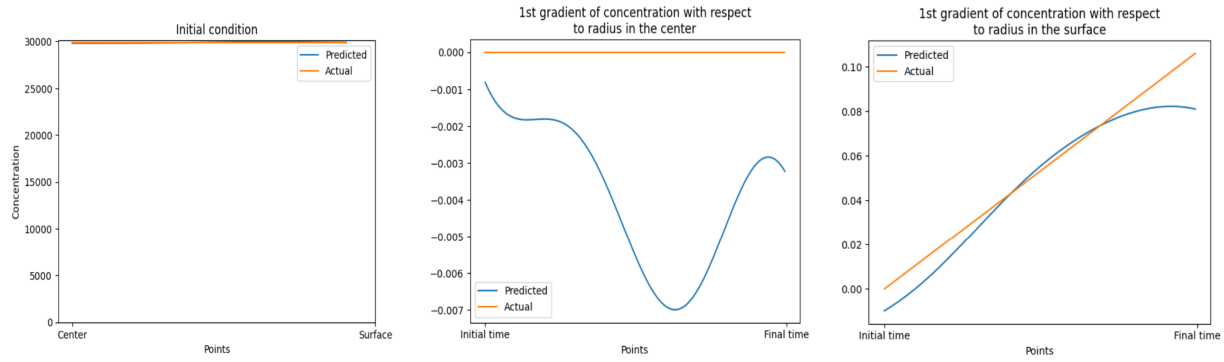


(b) Predicted concentration $\left[\frac{\text{mol}}{\text{m}^3}\right]$



(c) Absolute percentage error

Figure 4.10: (a) Actual concentration obtained from PyBaMM, (b) Prediction of the neural network model, and (c) Absolute percentage error for the whole domain



(a) (b) (c)

Figure 4.11: (a) Initial concentration prediction $\left[\frac{\text{mol}}{\text{m}^3}\right]$, (b) 1^{st} gradient with respect to the radius at $\text{BC}_{\text{center}}$, and (c) 1^{st} gradient with respect to the radius at $\text{BC}_{\text{surface}}$

Following the plots above, a table with the scores, in terms of Mean Absolute Error (MAE), of the loss components will be provided. For the whole domain, the MAPE will be used as a metric and the same approach will be followed also for the initial condition points. This normalization isn't applied in the BCs as the values are already expressed in non-dimensional values and the actual value is zero (hence, no MAPE metric can be obtained).

Table 4.3: Metrics for PI-DeepONet

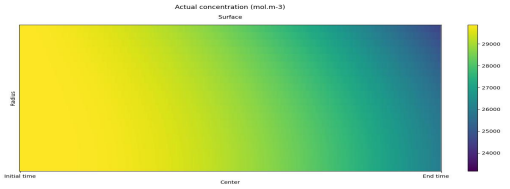
MAPE whole domain	MAPE IC	MAE BC_{center}	MAE $BC_{surface}$
4.40×10^{-2}	3.18×10^{-3}	3.8×10^{-3}	6.36×10^{-3}

Having achieved at least one order of magnitude less in both the BCs and IC, it becomes apparent that a potential area of concern lies within the collocation points (domain points) where PDE is applied.

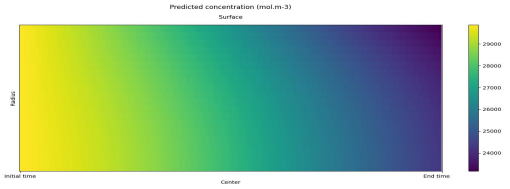
Henceforth, an approach will be undertaken to augment the collocation points within the domain by a factor of four, entailing a doubling of both spatial and temporal points.

4.7.2 Enhanced Base Architecture Utilizing Quadruple Collocation Points

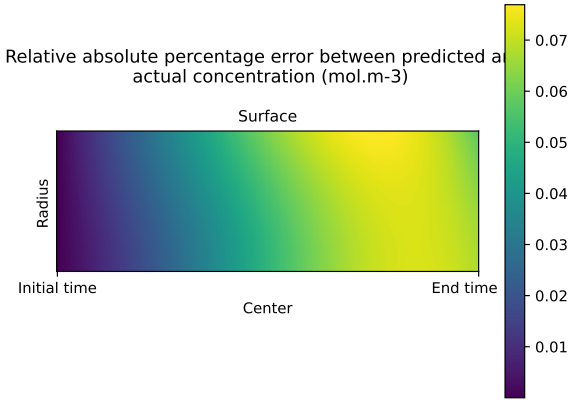
In this context, all parameters and configurations remain consistent with those detailed in the preceding subchapter (see 4.7.1). The sole modification implemented involves the utilization of four times the number of collocation points, comprising a doubling of points in both spatial and temporal coordinates.



(a) Actual concentration $\left[\frac{\text{mol}}{\text{m}^3}\right]$



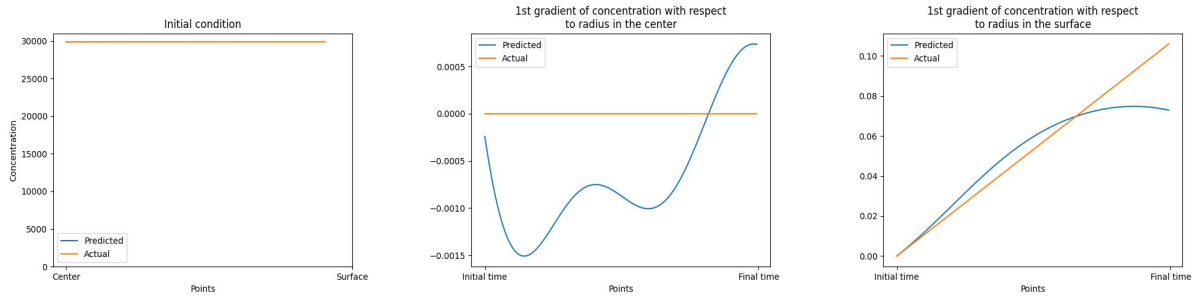
(b) Predicted concentration $\left[\frac{\text{mol}}{\text{m}^3}\right]$



(c) Absolute percentage error

Figure 4.12: (a) Actual concentration obtained from PyBaMM, (b) Prediction of the neural network model, and (c) Absolute percentage error across the entire domain for the more collocation points

Upon initial inspection, the results appeared to exhibit a minor degradation. Subsequently, an examination of the BCs and IC will be conducted, followed by the provision of a table presenting MAPE and MAE metrics as in the section 4.7.1.



(a) (b) (c)

Figure 4.13: (a) Initial concentration prediction $\left[\frac{\text{mol}}{\text{m}^3}\right]$, (b) 1^{st} gradient with respect to the radius at $\text{BC}_{\text{center}}$, and (c) 1^{st} gradient with respect to the radius at $\text{BC}_{\text{surface}}$ for the more collocation points

Table 4.4: Metrics for PI-DeepONet with for 4 times more collocation points

MAPE whole domain	MAPE IC	MAE $\text{BC}_{\text{center}}$	MAE $\text{BC}_{\text{surface}}$
4.7606×10^{-2}	1.48×10^{-4}	8.5×10^{-4}	7.71×10^{-3}

From the foregoing analysis, one notable observation is the reappearance of similar patterns as observed in 4.7.1. Notably, the results exhibit a subtle decline for the whole domain, which is unexpected given our anticipation of reduced errors from the more data points that were provided. A small deterioration also is followed in the $\text{BC}_{\text{surface}}$ while an improvement occurs in the $\text{BC}_{\text{center}}$ and IC, likely attributed to the increased collocation points within the domain and the enhanced continuity of information exchange within.

The discrepancy in the whole domain may stem from the potentially increased training time that should have been required or from instabilities introduced in the loss function by the inclusion of additional collocation points for the PDE. It is conceivable that the PDE now demands greater attention for learning, thereby resulting in a slight deterioration observed in the boundary condition (BC) at the surface, which subsequently impacts the entirety of the problem. This phenomenon can also be attributed to the competing nature of the components within the loss function, which attempts to reconcile four distinct losses simultaneously.

Nevertheless, one important conclusion that can be derived, is that for this case, there isn't a great need to go higher in the number of data points.

In light of these considerations, an alternative approach will be explored henceforth.

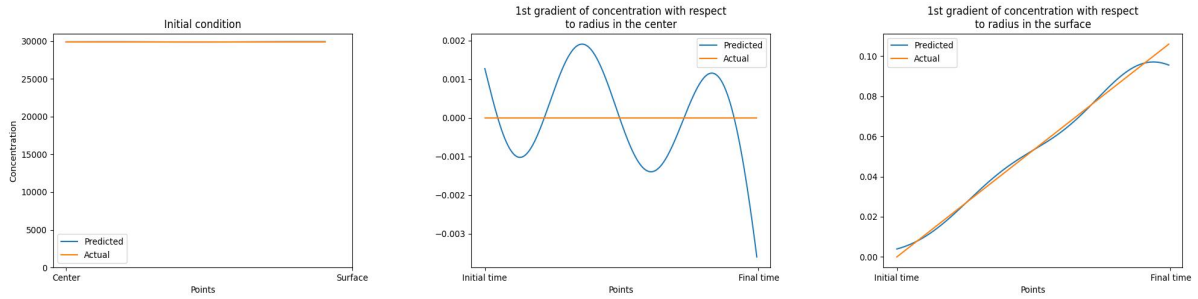
4.7.3 Adjusted loss function

In this section, an attempt to adjust the loss function will be taken into consideration. Also, for reasons of avoiding repetition, the concentration field of predicted/actual won't be displayed as the patterns of them remain the same as in the sections (4.7.1, 4.7.2).

4.7.3.1 1st epoch division

One potential point of concern that might affect the quality of the learning is the discrepancy in the order of magnitudes in the components of the loss function. This issue isn't new and it has already been highlighted by researchers, as in the work of Wang et al. [29].

To bring the loss components on the same scale and not have components dominating one over the other, in this section, we will apply a loss readjustment. This will be done by running the model first for one epoch and then keeping the loss component for each, as coefficients) and dividing them with these coefficients in the subsequent epochs respectively. In this way, every loss component will be readjusted to have close numbers.



(a) (b) (c)

Figure 4.14: (a) Initial concentration prediction $\left[\frac{\text{mol}}{\text{m}^3}\right]$, (b) prediction of BC in the center, and (c) prediction of BC in the surface for the readjusted loss function

Table 4.5: Metrics for PI-DeepONet with a readjusted loss function

MAPE whole domain	MAPE IC	MAE BC_{center}	MAE $BC_{surface}$
5.39×10^{-2}	8.15×10^{-4}	9.63×10^{-4}	1.61×10^{-3}

From the table 4.5, we observe that the $BC_{surface}$ has reduced by a factor of five. This marked improvement holds a notable significance considering that this region serves as the point of entry for information into the system (refer to equations 4.3). Nevertheless, contrary to the improvement of the information flowing in the system, there is a degradation in the results for the entire domain and the IC.

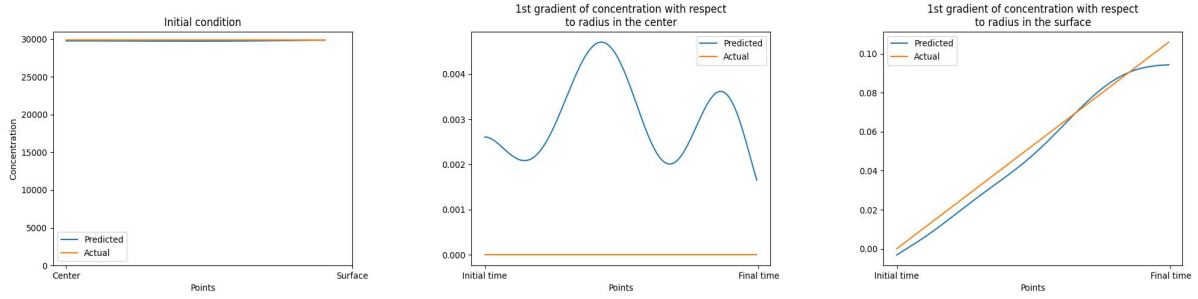
The cause for that degradation may be attributed to the fact that the PDE loss component had the largest coefficient from the first epoch. Consequently, the prominence of the PDE loss component diminished, resulting in insufficient training PDE respetively. Furthermore, as evidenced in Section 4.7.2, where a correlation is observed between the augmented PDE loss (stemming from four times more points) and the IC, brought the subsequent worsening in the IC.

4.7.3.2 50^{th} epoch division & 10 times PDE loss

In this section, since a diminished influence of the PDE was induced previously and subsequently affected adversely the MAPE for the whole domain, while concurrently yielding positive impacts on the $BC_{surface}$, a strategy akin to the one previously is employed in an effort to amalgamate the benefits of both approaches.

Here, rather than assigning coefficients of losses in the initial epoch, we do that in the 50th epoch, so that the loss components in the cost function aren't randomly distributed and have started to slightly converge.[7] Then, after taking each loss at the 50th epoch and retaining it as a coefficient for division in the subsequent epochs, we apply a multiplication by 10 of the PDE loss exclusively.

The intuition behind this approach lies in the fact that the PDE assumes a prominent role in the training process, as it is also in the case of $BC_{surface}$, that can impact the concentration dynamics within the domain if inadequately learned. However, maintaining its original value could potentially lead to the neglect of the other loss components.



(a) (b) (c)

Figure 4.15: (a) Initial concentration prediction $\left[\frac{mol}{m^3}\right]$, (b) prediction of BC in the center, and (c) prediction of BC in the surface for the readjusted loss function with 10 times the PDE loss

Table 4.6: Metrics for PI-DeepONet for the readjusted loss function with 10 times the PDE loss

MAPE whole domain	MAPE IC	MAE BC_{center}	MAE $BC_{surface}$
4.83×10^{-2}	3.75×10^{-3}	3.03×10^{-3}	3.98×10^{-3}

The analysis of Table 4.6 reveals an enhanced MAE for the entire domain compared to that discussed in subsection 4.7.3.1, albeit remaining inferior to the results obtained in section 4.7.1. This underscores the significant influence exerted by the PDE loss within the system. However, it is noteworthy that a trade-off exists between the MAE associated with the BCs and IC and that of the entire domain, wherein the reduction of one loss component corresponds to an increase in others, and vice versa. This observation underscores the importance and inherent difficulty of minimizing all loss components simultaneously.

In a subsequent endeavor, greater emphasis was placed on addressing the PDE loss by scaling up the coefficient, yet regrettably, this approach did not yield improved outcomes for the negative particle.

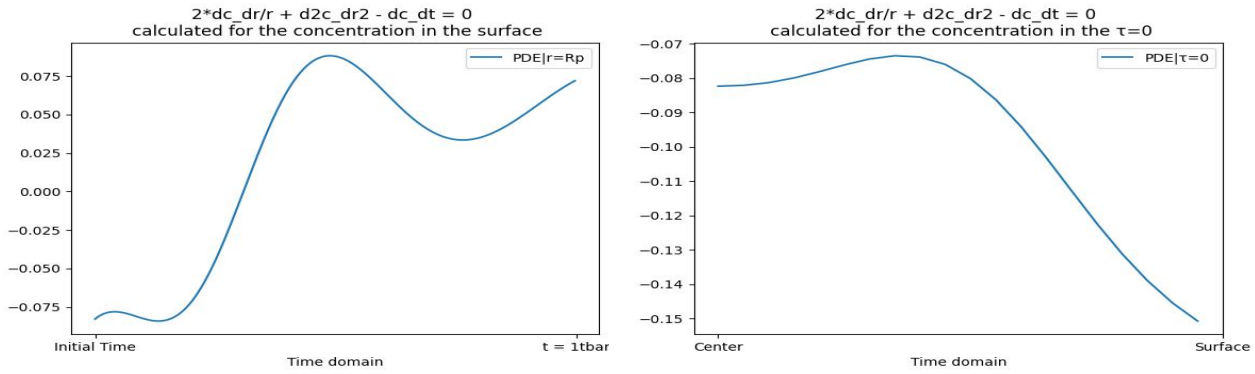
4.7.4 Enforce PDE constraints on BC & IC

Based on the insights collected from the preceding chapters, it has become apparent that the PDE plays a crucial role in achieving improved outcomes within the domain while

[7] at the first epochs the PDE loss is three orders higher than the rest loss components, while it remains higher than the others for the whole training duration if not an action is taken

affecting the training in boundaries/initial conditions. With this understanding in mind, a subsequent inquiry arises as to whether the PDE has been appropriately captured for the model in the boundaries and initial conditions.

Upon scrutinizing the PDE residuals presented for the case in Section 4.7.3.2, it becomes evident that the PDE has not been adequately fulfilled as can be seen from the image 4.16.



(a) (b)

Figure 4.16: (a) the PDE applied to the boundary in the surface and (b) the PDE applied at the initial condition

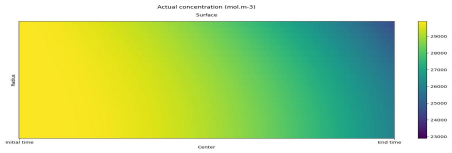
The MAE for the graphs depicted in Figure 4.16 is 5.869×10^{-2} for the $BC_{surface}$ and 9.713×10^{-2} for the IC.

Following the above insight, in this chapter, we will incorporate the PDE loss at the $BC_{surface}$ and IC[8].

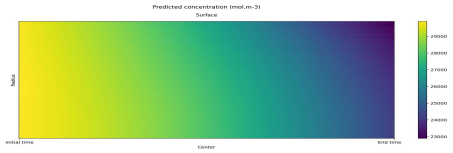
4.7.4.1 PDE at $BC_{surface}$

Initially, the PDE will be applied at the BC on the surface. This prioritization stems from its role serving as the point of entry for the input current into the system. This adjustment is implemented by including an additional term in the loss function.

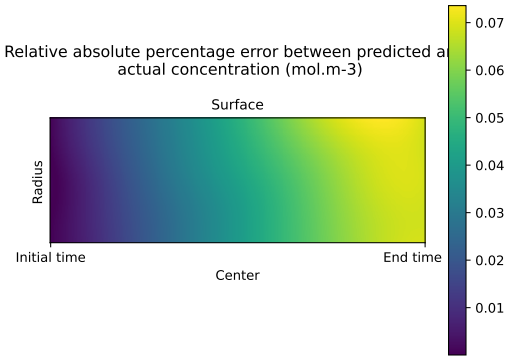
[8]The PDE loss cannot be applied directly at BC_{center} cause of the division with radius in the PDE which is zero at that location



(a) Actual concentration $\left[\frac{\text{mol}}{\text{m}^3}\right]$

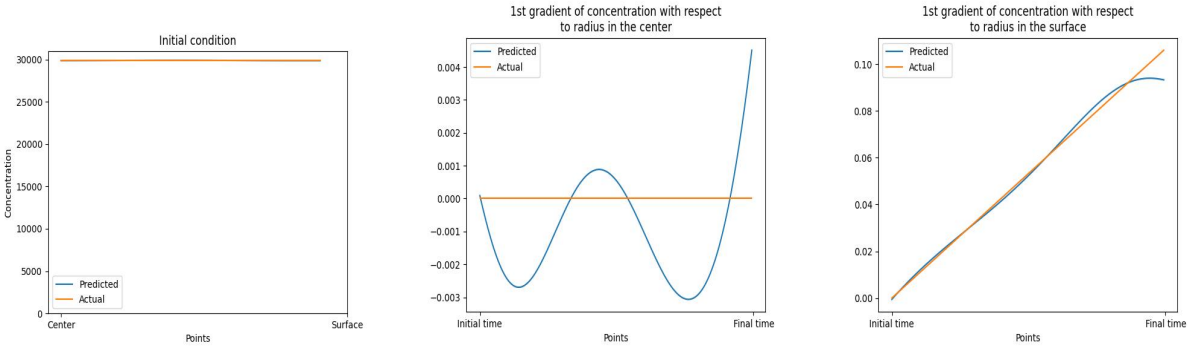


(b) Predicted concentration $\left[\frac{\text{mol}}{\text{m}^3}\right]$



(c) Absolute percentage error

Figure 4.17: (a) Actual concentration obtained from PyBaMM, (b) Prediction of the neural network model, and (c) Absolute percentage error for the whole domain for the PDE loss applied to $\text{BC}_{\text{surface}}$ model



(a) (b) (c)

Figure 4.18: (a) Initial concentration prediction $\left[\frac{\text{mol}}{\text{m}^3}\right]$, (b) 1^{st} gradient with respect to the radius at $\text{BC}_{\text{center}}$, and (c) 1^{st} gradient with respect to the radius at $\text{BC}_{\text{surface}}$ for the PDE loss applied to $\text{BC}_{\text{surface}}$ model

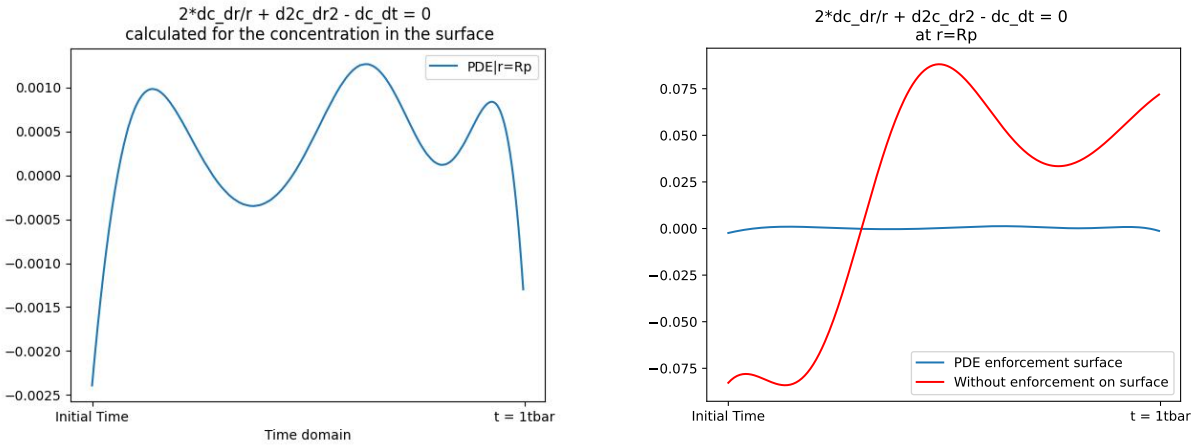
Table 4.7: Metrics for PI-DeepONet losses for the PDE loss applied to $\text{BC}_{\text{surface}}$ model

MAPE whole domain	MAPE IC	MAE $\text{BC}_{\text{center}}$	MAE $\text{BC}_{\text{surface}}$
4.10×10^{-2}	5.5×10^{-4}	1.57×10^{-3}	1.68×10^{-3}

The analysis of Table 4.7 reveals, for the first time, an improvement in the MAPE for the entire domain. Upon closer examination of the factors contributing to this success, it becomes apparent that the model achieves a more precise fitting of the $\text{BC}_{\text{surface}}$ loss while simultaneously reducing the magnitude of the other losses. One plausible explanation for

this improvement probably lies in the improved and smoother transition of the PDE from the surface to the domain, since now it's fitted also in the boundary of the surface, coupled with improved balance among its gradients of concentration, as shown in Equation 4.3a at the surface, thereby facilitating its subsequent propagation throughout the domain.

Below, is illustrated the PDE at the boundary of the surface.



(a) Enforcement of PDE on $BC_{surface}$

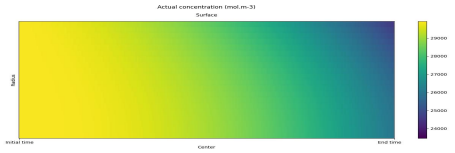
(b) Comparison with Figure 4.16a for improvement performance (red line) with Figure 4.19a (blue line)

Figure 4.19: PDE at $BC_{surface}$

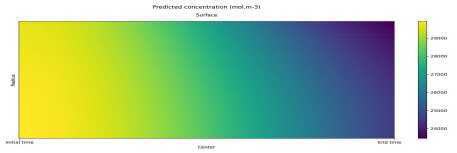
The improvement of the PDE at the $BC_{surface}$ is now evident compared to the image depicted in Figure 4.16a. The MAE now for the PDE at $BC_{surface}$ has decreased to 5.889×10^{-4}, while at the IC, it remains nearly unchanged at 8.891×10^{-2}.

4.7.4.2 PDE at IC&$BC_{surface}$

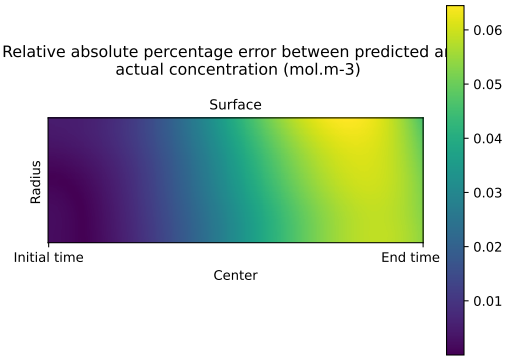
Now, we will attempt to apply on the same time the PDE to both the IC and $BC_{surface}$. The same approach also will be followed and an additional loss term will be introduced in the loss function again.



(a) Actual concentration $\left[\frac{\text{mol}}{\text{m}^3}\right]$

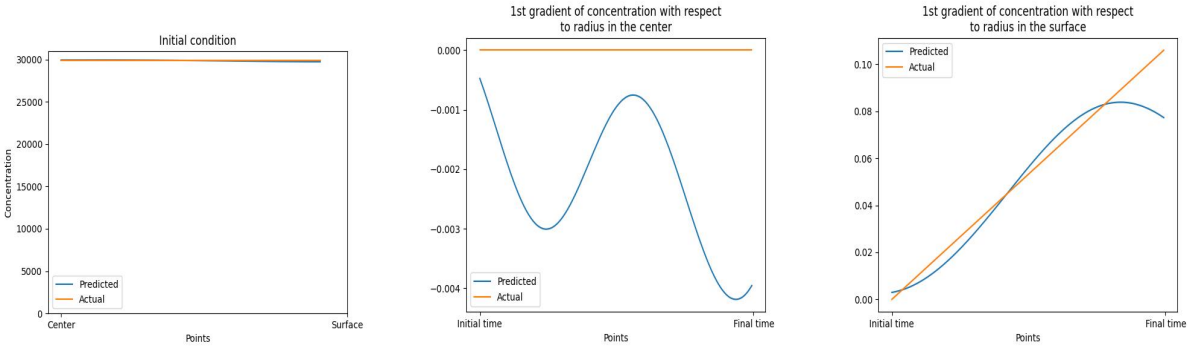


(b) Predicted concentration $\left[\frac{\text{mol}}{\text{m}^3}\right]$



(c) Absolute percentage error

Figure 4.20: (a) Actual concentration obtained from PyBaMM, (b) Prediction of the neural network model, and (c) Absolute percentage error for the whole domain for the PDE applied to IC&BC$_{\text{surface}}$ model



(a) (b) (c)

Figure 4.21: (a) Initial concentration prediction $\left[\frac{\text{mol}}{\text{m}^3}\right]$, (b) 1^{st} gradient with respect to the radius at BC$_{\text{center}}$, and (c) 1^{st} gradient with respect to the radius at BC$_{\text{surface}}$ for the PDE loss applied to IC&BC$_{\text{surface}}$ model

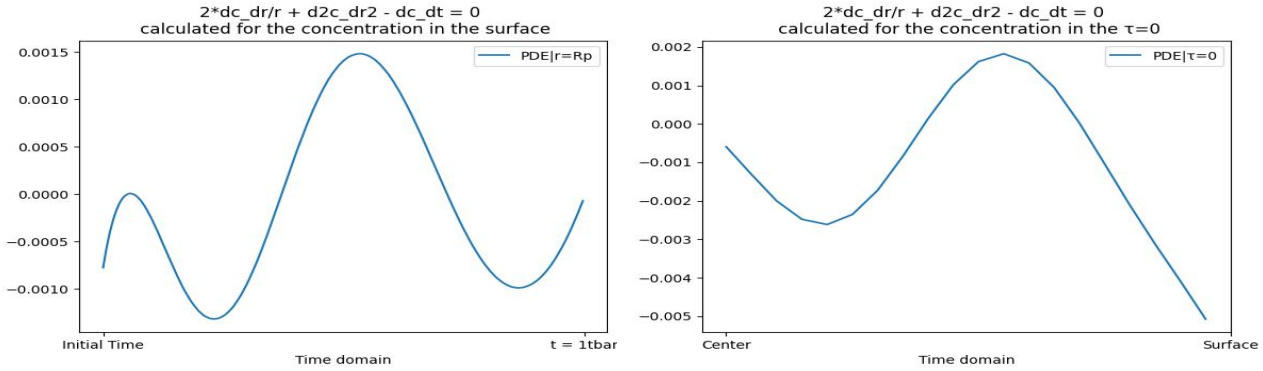
Table 4.8: Metrics for PI-DeepONet losses for the PDE loss applied to IC&BC$_{\text{surface}}$ model

MAPE whole domain	MAPE IC (normalized)	MAE BC$_{\text{center}}$	MAE BC$_{\text{surface}}$
3.31×10^{-2}	2.3×10^{-3}	2.25×10^{-3}	5.36×10^{-3}

The examination of Table 4.9 showcases an even more improved MAE for the whole domain, indicating the efficacy of this approach. This can potentially be attributed to the enhanced resolution of concentration gradients originating from the IC/BC$_{\text{surface}}$.

However, notwithstanding the reduced MAPE for the entire domain, the MAE for the remaining loss components depicted in the table demonstrates a deterioration. This phenomenon arises due to the inclusion of two additional loss terms in the loss function, thereby necessitating a balancing act during the training of all the loss components. Furthermore, the introduction of these additional loss terms induces supplementary effects, such as the observation of diminished gradients at the initial stages of the graph in Figure 4.21c.

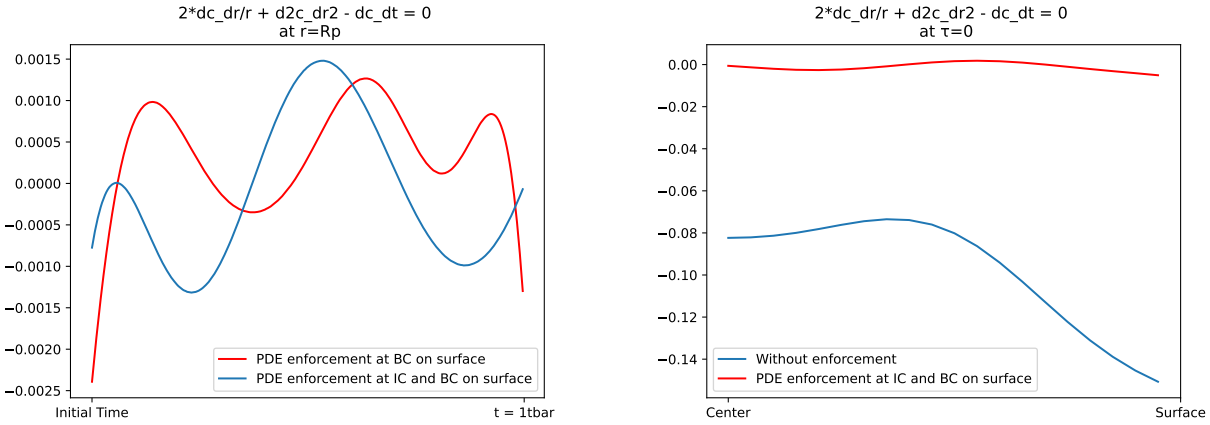
For illustrative purposes, the graphs depicting the added loss components are presented below.



(a) (b)

Figure 4.22: (a) the PDE applied to the boundary in the surface and (b) the PDE applied at the initial condition for the PDE at IC&$BC_{surface}$ model

For comparison purposes, the differences between Figure 4.19a and Figure 4.16 with the Figure 4.22 will be shown to facilitate the evaluation of any improvements.



(a) Red line is from Figure 4.19a, blue line is from Figure 4.22a

(b) Red line is from Figure 4.16b, blue line is from Figure 4.22b

Figure 4.23: Comparison of models with enforcement of PDE at IC and $BC_{surface}$

The improvement observed in these two graphs is significant compared to those without the application of the PDE at the IC and $BC_{surface}$, as illustrated in Figures 4.23b and 4.19a. Moreover, from Figure 4.23a a slight further improvement can be seen in the PDE at $BC_{surface}$ with the enforcement of PDE to IC, resulting in more accurate capture of the initial time value and, subsequently, milder fluctuations of the PDE around zero. In this instance,

the MAE for the BCsurface drops to 7.6×10^{-4}, while for the IC, it shifts to 1.826×10^{-3}. These MAEs have been decreased by roughly two orders of magnitude now.

This confirms the assumption that the lower MAE observed across the entire domain correlates with the improved resolution of concentration gradients.

4.7.5 Best model

In light of the previous results, in this section different setting architectures will be examined while keeping the effective implementation of PDE at IC&$BC_{surface}$. More specifically, four different settings were employed.

Specifically[9]:

First architecture
Branch: 3 layers with 100, 100, 50 neurons
Trunk: 2 layers with 40 and 50 neurons

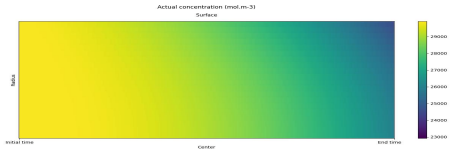
Second architecture
Branch: 3 layers with 100, 100, 100 neurons
Trunk: 3 layers with 50, 100 and 100 neurons

Third architecture
Branch: 4 layers with 100, 100, 200, 200 neurons
Trunk: 4 layers with 100, 100, 100, 200 neurons

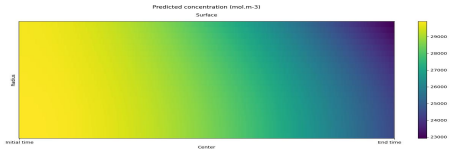
Fourth architecture
Branch: 5 layers with 200, 200, 200, 200, 200 neurons
Trunk: 4 layers with 200, 200, 200, 200 neurons

The fourth architecture is characterized by the smallest MAE for the domain, therefore the results from it will be presented only. The MAPE for the entire domain is 2.37×10^{-2}.

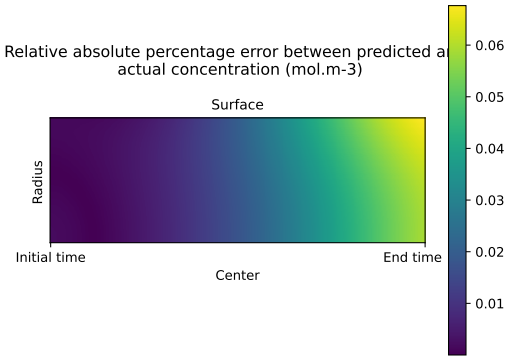
[9]The layers are presented from input to output



(a) Actual concentration $\left[\frac{\text{mol}}{\text{m}^3}\right]$

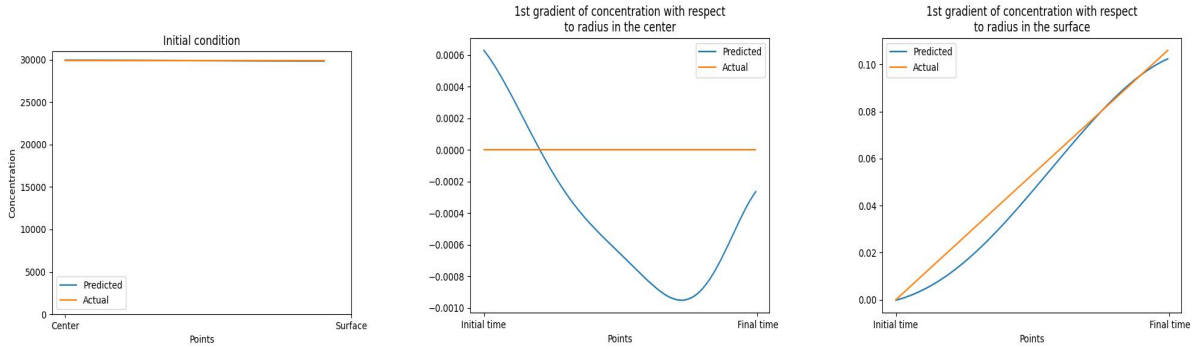


(b) Predicted concentration $\left[\frac{\text{mol}}{\text{m}^3}\right]$



(c) Absolute percentage error

Figure 4.24: (a) Actual concentration obtained from PyBaMM, (b) Prediction of the neural network model, and (c) Absolute percentage error for the whole domain for the best model



(a) (b) (c)

Figure 4.25: (a) Initial concentration prediction $\left[\frac{\text{mol}}{\text{m}^3}\right]$, (b) 1^{st} gradient with respect to the radius at $\text{BC}_{\text{center}}$, and (c) 1^{st} gradient with respect to the radius at $\text{BC}_{\text{surface}}$ for the best model

Table 4.9: Metrics for PI-DeepONet losses for the best model

MAPE whole domain	MAPE IC	MAE $\text{BC}_{\text{center}}$	MAE $\text{BC}_{\text{surface}}$
2.37×10^{-2}	1.16×10^{-3}	5.5×10^{-4}	5.18×10^{-3}

The MAPE for the entire domain has decreased by 46.14% from the original value that had in the base architecture. This improvement could stem from the bigger capacity of the model.

5. CONCLUSIONS AND FUTURE WORK

5.1 Summary

In this work, the intersection of machine learning with physics is explored through the lens of one of the state-of-the-art methods for operator learning, DeepONet. During the study, concepts from PINNs were also used for incorporating physics in DeepONet not only from data. With the aim of assessing the applicability of these methodologies in practical engineering scenarios, this approach is implemented in the analysis of a forced oscillator and a battery discharging scenario, representing ODE and PDE problems, respectively.

The primary objective of this research was to establish the foundational knowledge and enhance the understanding of modeling physics-related problems, particularly in the context of battery systems. It became evident that training models for battery systems presented added complexity compared to the forced oscillator problem.

The forced oscillator was more straightforward to model due to the inherent simplicity of solving ODEs. The study of the forced oscillator demonstrated that a hybrid training approach, combining both data and physics, leads to improved results. Additionally, training the model on more complex inputs enhanced its ability to generalize and extrapolate to less complex inputs. Improved performance can be further bolstered by incorporating an additional initial layer in the trunk net, which proves to yield superior results.

In the case of the battery model, which was based solely on physics, the study highlighted the importance of appropriately weighting the loss components and the significance of integrating the PDE into the boundary and initial conditions. This approach considerably enhanced model performance. However, contrasting results were observed when comparing purely physics-based training to data-driven approaches, with the latter demonstrating greater ease to model and yielding highly satisfactory outcomes.

These results hold a significant value since these findings and methods can be extended to several engineering and science domains. Moreover, it was the first time that a battery case was tried to be addressed using only physics and not a hybrid solution involving data. This study lays the groundwork for the future development of models that rely solely on physics, particularly in scenarios where data availability is limited. Additionally, it paves the way for creating more robust deep-learning solvers tailored to battery applications.

5.2 Future work

Looking ahead, there are several promising avenues for future research in this rapidly evolving field or directions for further exploration and refinement. First of all, in the case of the battery with physics only, a surrogate model can be built comprised of one more PI-DeepONet for modeling the positive particle, thereby facilitating the development of an end-to-end model that will be capable of predicting the voltage output. Likewise, exploring alternate systems of equations beyond the SPM equations could yield enhanced accuracy in battery modeling, albeit at the expense of increased complexity due to the incorporation of additional equations.

Additionally, another interesting topic that can be explored is for the battery to have different initial states moving beyond variations solely in current, thereby aligning more closely with real-world applications.

Another promising avenue for exploration lies in the realm of inverse problems, wherein models endeavor to deduce the parameters or conditions of a system given known inputs and observed outputs. Despite advancements in engineering, the inverse problem remains among the most challenging to address, representing a rich area for future investigation and innovation.

ABBREVIATIONS - ACRONYMS

PINNs	Physics Informed Neural Networks
PIML	Physics Informed Machine Learning
PDEs	Partial Differential Equations
ODEs	Ordinary Differential Equations
BC	Boundary Conditions
IC	Initial Conditions
SciML	Scientific Machine Learning
DeepONet	Deep Operator Network
PI-DeepONet	Physics Informed Deep Operator Network
MSE	Mean Square Error
FEM	Finite Element Methods
FVM	Finite Volume Methods
MLP	Multi-layer Perceptron
SPM	Single Particle Model
LIB	Lithium-Ion Batteries
DFN	Doyle–Fuller–Newman
BMS	Battery Management System
P2D	Pseudo two Dimensional
Li-ions	Lithium ions
FNO	Fourier Neural Operators
MAE	Mean Absolute Error
RMSE	Root Mean Square Error
MAPE	Mean Absolute Percentage Error

REFERENCES

[1] 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 conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, USA, August 2016. USENIX Association.

[2] Chang-Hui Chen, Ferran Brosa Planella, Kieran O'Regan, Dominika Gastol, W. Dhammika Widanage, and Emma Kendrick. Development of Experimental Techniques for Parameterization of Multi-scale Lithium-ion Battery Models. *Journal of The Electrochemical Society*, 167(8):080534, February 2020. Publisher: IOP Publishing.

[3] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, July 1995. Conference Name: IEEE Transactions on Neural Networks.

[4] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. In *Proceedings of the 35th International Conference on Machine Learning*, pages 794–803. PMLR, July 2018. ISSN: 2640-3498.

[5] Gyouho Cho, Mengqi Wang, Youngki Kim, Jaerock Kwon, and Wencong Su. A Physics-Informed Machine Learning Approach for Estimating Lithium-Ion Battery Temperature. *IEEE Access*, 10:88117–88126, 2022. Conference Name: IEEE Access.

[6] Gyouho Cho, Di Zhu, Jeffrey Joseph Campbell, and Mengqi Wang. An LSTM-PINN Hybrid Method to Estimate Lithium-Ion Battery Pack Temperature. *IEEE Access*, 10:100594–100604, 2022. Conference Name: IEEE Access.

[7] QiZhi He, Panos Stinis, and Alexandre M. Tartakovsky. Physics-constrained deep neural network method for estimating parameters in a redox flow battery. *Journal of Power Sources*, 528:231147, April 2022.

[8] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, June 2021. Number: 6 Publisher: Nature Publishing Group.

[9] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, and Andrew Stuart. Neural Operator: Learning Maps Between Function Spaces With Applications to PDEs.

[10] Minkyu Kwak, Hong Sung Jin, Bataa Lkhagvasuren, and Delgermurun Oyunmunkh. A Robust State of Charge Estimator Based on the Fourier Neural Operator for xEV Batteries. *Journal of The Electrochemical Society*, 170(10):100504, July 2023. Publisher: IOP Publishing.

[11] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[12] Weihan Li, Jiawei Zhang, Florian Ringbeck, Dominik Jöst, Lei Zhang, Zhongbao Wei, and Dirk Uwe Sauer. Physics-informed neural networks for electrode-level state estimation in lithium-ion batteries. *Journal of Power Sources*, 506:230034, September 2021.

[13] Zong-Yi Li, Nikola B. Kovachki, K. Azizzadenesheli, Burigede Liu, K. Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. *ArXiv*, October 2020.

[14] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole Graph Neural Operator for Parametric Partial Differential Equations. In *Advances in Neural Information Processing Systems*, volume 33, pages 6755–6766. Curran Associates, Inc., 2020.

[15] 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, March 2021. Publisher: Nature Publishing Group.

[16] Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, April 2022.

[17] Renato G. Nascimento, Matteo Corbetta, Chetan S. Kulkarni, and Felipe A. C. Viana. Li-ion Battery Aging with Hybrid Physics-Informed Neural Networks and Fleet-wide Data. *Annual Conference of the PHM Society*, 13(1), November 2021. Number: 1.

[18] F. Brosa Planella, W. Ai, A. M. Boyce, A. Ghosh, I. Korotkin, S. Sahu, V. Sulzer, R. Timms, T. G. Tranter, M. Zyskin, S. J. Cooper, J. S. Edge, J. M. Foster, M. Marinescu, B. Wu, and G. Richardson. A continuum of physics-based lithium-ion battery models reviewed. *Progress in Energy*, 4(4):042003, April 2022. Publisher: IOP Publishing.

[19] Dimitris C. Psichogios and Lyle H. Ungar. A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 38(10):1499–1511, 1992. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/aic.690381003.

[20] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the Spectral Bias of Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5301–5310. PMLR, May 2019. ISSN: 2640-3498.

[21] 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, February 2019.

[22] Soumya Singh, Yvonne Eboumbou Ebongue, Shahed Rezaei, and Kai Peter Birke. Hybrid Modeling of Lithium-Ion Battery: Physics-Informed Neural Network for Battery State Estimation. *Batteries*, 9(6):301, June 2023. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.

[23] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. In *Advances in Neural Information Processing Systems*, volume 33, pages 7462–7473. Curran Associates, Inc., 2020.

[24] Venkat R. Subramanian, James A. Ritter, and Ralph E. White. Approximate Solutions for Galvanostatic Discharge of Spherical Particles I. Constant Diffusion Coefficient. *Journal of The Electrochemical Society*, 148(11):E444, October 2001. Publisher: IOP Publishing.

[25] Valentin Sulzer, Scott G. Marquis, Robert Timms, Martin Robinson, and S. Jon Chapman. Python Battery Mathematical Modelling (PyBaMM). 9(1):14, June 2021. Number: 1 Publisher: Ubiquity Press.

[26] Bo Sun, Junlin Pan, Zeyu Wu, Quan Xia, Zili Wang, Yi Ren, Dezhen Yang, Xing Guo, and Qiang Feng. Adaptive evolution enhanced physics-informed neural networks for time-variant health prognosis of lithium-ion batteries. *Journal of Power Sources*, 556:232432, February 2023.

[27] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *Advances in Neural Information Processing Systems*, volume 33, pages 7537–7547. Curran Associates, Inc., 2020.

[28] Kejun Tang, Xiaoliang Wan, and Chao Yang. DAS-PINNs: A deep adaptive sampling method for solving high-dimensional partial differential equations. *Journal of Computational Physics*, 476:111868, March 2023.

[29] Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality for training physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421:116813, March 2024.

[30] 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):A3055–A3081, January 2021. Publisher: Society for Industrial and Applied Mathematics.

[31] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Science Advances*, 7(40):eabi8605, September 2021. Publisher: American Association for the Advancement of Science.

[32] 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, January 2022.

[33] Zhi-Qin John Xu. Frequency Principle: Fourier Analysis Sheds Light on Deep Neural Networks. *Communications in Computational Physics*, 28(5):1746–1767, June 2020.

[34] Ge Yang, Anurag Ajay, and Pulkit Agrawal. OVERCOMING THE SPECTRAL BIAS OF NEURAL VALUE APPROXIMATION. 2022.

[35] Colby L. Wight & Jia Zhao. Solving Allen-Cahn and Cahn-Hilliard Equations using the Adaptive Physics Informed Neural Networks. *Communications in Computational Physics*, 29(3):930–954, June 2021.

[36] Qiang Zheng, Xiaoguang Yin, and Dongxiao Zhang. Inferring electrochemical performance and parameters of Li-ion batteries based on deep operator networks. *Journal of Energy Storage*, 65:107176, August 2023.

[37] Qiang Zheng, Xiaoguang Yin, and Dongxiao Zhang. State-space modeling for electrochemical performance of Li-ion batteries with physics-informed deep operator networks. *Journal of Energy Storage*, 73:109244, December 2023.

[38] J. C. Álvarez Antón, P. J. García Nieto, F. J. de Cos Juez, F. Sánchez Lasheras, M. González Vega, and M. N. Roqueñí Gutiérrez. Battery state-of-charge estimator using the SVM technique. *Applied Mathematical Modelling*, 37(9):6244–6253, May 2013.