NOVEL ARCHITECTURES AND TRAINING ALGORITHMS FOR DEEP NEURAL NETWORKS

A Dissertation

by

CAIO CESAR MEDEIROS DAVI

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Ulisses Braga-Neto
Committee Members,	Yoonsuck Choe
	Nicholas Duffield
	Yang Shen
Head of Department,	Miroslav Begovic

December 2022

Major Subject: Electrical Engineering

ABSTRACT

Nowadays, machine learning and deep learning are present in the most diverse types of applications. From such diversity, many particular designs, architectures, training methods were created, given the variety of applications of many different areas. Rather than trying to find a multi-purpose scheme to generate and train deep neural networks, which would be an impracticable challenge, this work aims to deliver well suited training techniques for well defined problems in specific fields. Every single domain has its own necessities and specificities, thus it is essential to have individual solutions for each case.

In the bio-informatics domain we propose the gGAN, a novel approach to train GANs, which is capable of generating labeled genetic datasets using a small labeled dataset and a larger unlabeled dataset, exploiting concepts of semi-supervised learning and data augmentation to create a new approach to deal with the limited labeled data available to researchers. This method may also be used as a self-aware classifier, a classifier with a second level of confidence. Since it is only based on genetic profiles, it can be applied at any stage of the disease (or even before infection). This allows the usage as a triage tool, able to prognose early-infected patients, avoiding the exposure of healthcare professionals who are sensitive to the disease.

This work also addresses a Scientific Machine Learning technique, the Physically Informed Neural Networks (PINNs). Evidence shows that PINN training by gradient descent displays pathologies that often prevent convergence when solving PDEs with irregular solutions. In this work, we propose the use of a Particle Swarm Optimization (PSO) approach to train PINNs. The resulting PSO-PINN algorithm not only mitigates the undesired behaviors of PINNs trained with standard gradient descent but also presents an ensemble approach to PINN that affords the possibility of robust predictions with quantified uncertainty. Comprehensive experimental results show that PSO-PINN, using a modified PSO algorithm with a behavioral coefficient schedule, outperforms other PSO variants for training PINNs, as well as PINN ensembles trained with standard ADAM.

Furthermore, we propose two distinct extensions for this method, namely Multi-Objective

PSO-PINN and Multi-Modal PSO-PINN. The first one acknowledges the PINN as a multi-objective problem and handles the PSO-PINN training as such. This approach unleashes a new paradigm to deal with PINNs, allowing the analysis of the problem and finding out if the model-driven and data-driven portions of the PINN are in agreement. The latter promotes a desirable characteristic of the PSO-PINN, the diversity of the solutions. The Multi-Modal approach enforces multiple local optima during the training, guaranteeing a solution composed of a diverse ensemble.

DEDICATION

To my wife, Luiza.

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my advisor Dr. Braga-Neto for all his guidance, motivation, and support. And especially for believing in me during my Ph.D. journey. I also would like to thank my "pre-doc" advisor, Dr. Fernando Buarque, I had my first class with him about 17 years ago, and I keep learning from him these days.

I also need to thank all my friends at Texas A&M, along with the good fellows I met at the Texas A&M Rural and Community Health Institute. Also, thanks to all my friends from the Brazilian community at College Station. Out of fear of being unfair, I do not dare to name each and every one of you, but be sure I could not have asked for better company for the last four years.

I will forever be thankful for the family I have. For their support since very early in my career and for their understanding, especially regarding my absence. My parents are most certainly the biggest inspiration for the individual who I am.

And I am especially in debt with my wife, Luiza. For all her support, patience, care, and partnership during all the years we have been together. And above everything else, for all the love she endlessly devotes to me. Nothing of this would be possible without you. Thank you.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professors Ulisses Braga-Neto (Advisor), Nicholas Duffield and Yang Shen of the Department of Electrical & Computer Engineering, and Professor Yoonsuck Choe of the Department of Computer Science & Engineering.

The data analyzed for Chapter 2 was provided by Professor Bartolomeu Acioli-Santos of the Oswaldo Cruz Foundation (FIOCRUZ) and the International Genome Sample Resource (IGSR), from the 1000 Genomes Project.

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by a graduate merit scholarship from the Department of Electrical and Computer Engineering.

NOMENCLATURE

ADAM	Adaptive Moment Optimization
AFD	Allelic Frequency Distance
API	Application Programming Interface
BP	Back Propagation
COVID	Coronavirus Disease
DENV	Dengue Virus
DF	Dengue Fever
GAN	Generative Adversarial Network
gGAN	Genetic Generative Adversarial Networks
GPU	Graphics Processing Unit
GWAS	Genome-Wide Association Studies
ML	Machine Learning
MMO	Multi-Modal Optimization
MNIST	Modified National Institute of Standards and Technology
MOO	Multi-Objective Optimization
MSE	Mean Square Error
MTL	Multi-Task Learning
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
PHI	Protected Health Information
PINN	Physically-Informed Neural Network
PINN	Physically-Informed Neural Networks

PSO	Particle Swarm Optimization
PSO	Particle Swarm Optimization
PSO-BP	Particle Swarm Optimization with Backpropagation
PSO-BP-CD	Particle Swarm Optimization with Backpropagation and Coefficient Decay
PSO-PINN	Physically-Informed Neural Networks using Particle Swarm Optimization
ReLU	Rectified Linear Unit
ReLU	Rectified Linear Unit
RFE	Recursive Feature Elimination
SD	Severe Dengue
SNP	Single-Nucleotide Polymorphism
SVM	Support Vector Machine
tanh	Hyperbolic Tangent Function
TPU	Tensor Processing Unit
WHO	World Health Organization

TABLE OF CONTENTS

Page

ABSTRACT ii

DEDICATION iv

ACKNOWLEDGMENTS v

CONTRIBUTORS AND FUNDING SOURCES vi

NOMENCLATURE vii

TABLE OF CONTENTS ix

LIST OF FIGURES xi

LIST OF TABLES xiii

1. INTRODUCTION 1

    1.1 Motivation 2
    1.2 Contributions 2
    1.3 Organization 4

2. A SEMI-SUPERVISED GENERATIVE ADVERSARIAL NETWORK FOR PREDICTION OF GENETIC DISEASE OUTCOMES 5

    2.1 Abstract 5
    2.2 Introduction 5
    2.3 Background 6
    2.4 Methods 8
        2.4.1 Data Preprocessing 8
        2.4.2 gGAN Architecture 10
    2.5 Results 12
    2.6 Conclusion 14

3. PSO-PINN: PHYSICS-INFORMED NEURAL NETWORKS TRAINED WITH PARTICLE SWARM OPTIMIZATION 16

    3.1 Abstract 16
    3.2 Introduction 16
    3.3 Background 18

3.3.1 Deep Neural Networks 18
3.3.2 Physics-Informed Neural Networks 19
3.3.3 Particle Swarm Optimization 20
3.4 PSO-PINN Algorithm 21
3.4.1 The PSO variants 21
3.4.2 The Algorithm 22
3.4.3 The Ensemble Solution 22
3.5 Experimental Results 24
3.5.1 A Simple PSO-PINN Example 24
3.5.2 PSO-PINN Performance using Different PSO Variants 26
3.5.2.1 Advection Equation 26
3.5.2.2 Burgers Equation 27
3.5.2.3 Heat Equation 29
3.5.3 Comparison with Traditional Ensembles of PINNs 30
3.5.3.1 Forced Heat Equation 31
3.5.3.2 The Allen-Cahn Equation 33
3.5.4 Discussion 35
3.6 Conclusion 35

4. EXTENDING THE PSO-PINN: THE MULTI-OBJECTIVE AND MULTI-MODAL APPROACHES 37

4.1 Abstract 37
4.2 Introduction 37
4.3 Related Work 39
4.4 Preliminaries 40
4.4.1 Deep Neural Networks 40
4.4.2 Physics-Informed Neural Networks 41
4.4.3 Multi-Objective Optimization 43
4.4.4 Multi-Modal Optimization 45
4.5 Multi-Objective PSO-PINN 46
4.5.1 The Poisson Equation 48
4.5.2 The Heat Equation 51
4.6 Multi-Modal PSO-PINN 54
4.6.1 The Helmholtz Equation 55
4.6.2 The Allen-Cahn Equation 59
4.7 Conclusion 62

5. CONCLUSION 63

REFERENCES 65

LIST OF FIGURES

FIGURE Page

1.1 Parallel between biological and artificial neurons. 1

2.1 Data preprocessing procedure. Each line in each dataset represents a genetic profile of a patient. The SNPs are selected according to an Allelic Frequency Distance (AFD) threshold and the SVM-RFE algorithm. In this figure, AFD thresholds of 0.07, 0.10, 0.21 are depicted. 7

2.2 Model architecture. The discriminator network is fed two datasets (labeled dataset (n=102), unlabeled dataset (n=2504)) and the synthetic genetic profiles from the generator network. The discriminator training consists of two phases, the first using real unlabeled data and synthetic data, and the second using labeled data. 9

3.1 The PSO-BP update. 21

3.2 Poisson Equation - First row: solutions in the PSO-PINN ensemble as training progresses, using the PSO-BP-CD method. Second row: Best individual, mean and variance of PSO-PINN ensemble. Third row: PSO-PINN mean loss error. The grey band is the two-sided 1-standard deviation region. 25

3.3 Advection Equation - Evolution of the PSO-PINN ensemble using the PSO-BP-CD method, showing the mean and two-sided 1-standard deviation of the ensemble. 27

3.4 Burgers Equation - Evolution of the PSO-PINN ensemble using the PSO-BP-CD method, showing the mean and two-sided 1-standard deviation of the ensemble. 29

3.5 Heat Equation - Evolution of the PSO-PINN ensemble using the PSO-BP-CD method, showing the mean and two-sided 1-standard deviation of the ensemble. 30

3.6 Forced Heat Equation - Results obtained by the PSO-PINN and conventional (ADAM) PINN ensembles at the end of training. 32

3.7 Allen-Cahn Equation - Results obtained by the PSO-PINN and conventional (ADAM) PINN ensembles at the end of training. 34

4.1 Multi-Objective PSO-PINN solutions for Poisson Equation – The results depicted are for two independent training. On the left column, a Multi-Objective PSO-PINN was trained with good data but noisy physics, i.e., a bad choice of coefficients for the PDE. On the right column, the Multi-Objective PSO-PINN was trained with a noisy dataset. On the top row, the Pareto fronts for each scenario. Each point on the Pareto front represents a neural network, and in the second and third row, we have the best solution to each dimension. This solution is also marked in red on the Pareto front. 50

4.2 Multi-Objective PSO-PINN solutions for the Heat Equation – Each row represents one of the solutions found in the Pareto Front (Figure 4.3). On top, the solution that minimizes the first loss from the Loss vector. In the middle, we have the solution most close to zero, one that minimizes the sum of all in (4.18). In the last row is the best solution for the boundary and data losses. 53

4.3 The Pareto front of the Multi-Objective PSO-PINN solutions for the Heat Equation. 54

4.4 Multi-Modal PSO-PINN solutions for the Helmholtz Equation – On top, the PSO-PINN solution. Bellow, the Multi-Modal PSO-PINN solution. 57

4.5 Population Euclidean Distance for Helmholtz Equation. 59

4.6 Results of the Allen-Cahn Equation – On top, the solution from an ensemble composed by conventional PINNs using ADAM optimization. Followed by the PSO-PINN solution and the Multi-Modal PSO-PINN solution. 61

LIST OF TABLES

TABLE Page

2.1 Architecture of the discriminator networks. 11

2.2 Architecture of the generator networks. 11

2.3 Accuracy rates according to testing procedures T_1 and T_2. 13

3.1 PSO-PINN results comparing the PSO variants, displaying the mean, standard deviation, and minimum (in parenthesis) of the L2 error of the PSO-PINN combined prediction across 10 independent repetitions of the experiment. 31

4.1 Allen-Cahn equation results. Errors and variance for each optimization strategy adopted in this experiment. 60

1. INTRODUCTION

Artificial neural networks are a well-known machine learning technique that simulates the mechanics of learning in neural systems of biological organisms. Simply put, the neurons are connected to each other by their axons and dendrites, and these connections are referred to as synapses. Throughout their lives, living organisms are able to learn by changing the thresholds to activate those synapses. A artificial neuron is composed by inputs and outputs in order to mimic the biological behavior of the dendrites and axons, respectively. Each input have a related weight and a bias, which will guarantee the required plasticity for the learning process. The figure 1.1 illustrates those structures in parallel.

The most basic neural network would be a network composed by just one neuron. This arrangement is often called perceptron and it is illustrated on Figure 1.1 (b). The perceptron was proposed by Rosenblatt in 1958[1] and since then it has evolved in different architectures[2][3], training methods [4][5][6], and optimizations[7][8].

However, there is not a single training method or architecture suitable for every possible problem. Hence, exploring multiple alternatives and creating new approaches are necessary and recurrent tasks. In this context, this work proposes new methods for training deep neural networks.

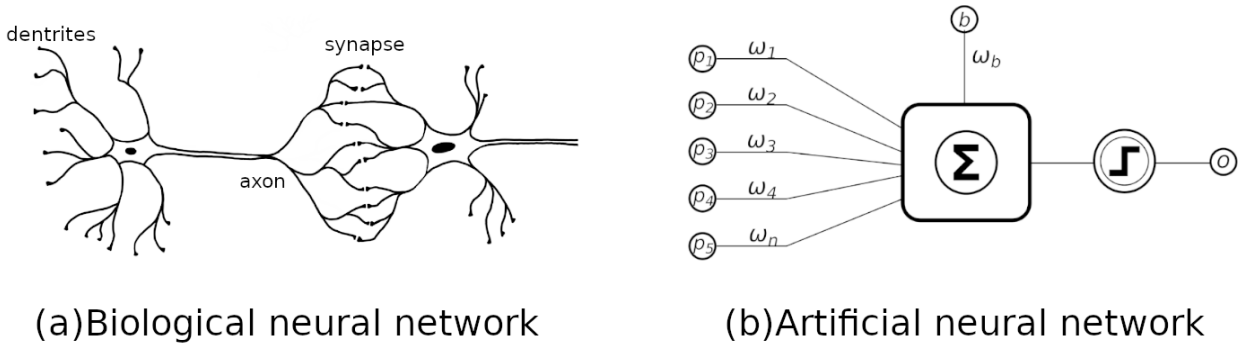

Figure 1.1: Parallel between biological and artificial neurons.

1.1 Motivation

Nowadays, machine learning and deep learning are present in the most diverse types of applications. From health care to analytical finances and going through wide-ranging fields such education, security, and even entertainment[9][10][11][12][13][14]. From such diversity, many particular designs, architectures, training methods were created, given the variety of applications of many different areas.

Rather than trying to find a multi-purpose scheme to generate and train deep neural networks, which would be an impracticable challenge, this work aims to deliver well suited training techniques for well defined problems in specific fields. Every single domain has its own necessities and specificities, thus it is essential to have individual solutions for each case. Taking for example the Chapter 2, which is centered on bio-informatics and specifically on Machine Learning for genetic models, the main challenge there is the scarcity of large enough datasets to train data-driven models. Obtaining a patient cohort to fill such dataset would be a hard an expensive task. Therefore, the ideal solution would not only require a smaller dataset, but it could be able to somehow expand the knowledge acquired from it.

Physically-informed neural networks (PINNs) are the main topic of Chapter 3 and 4. Although it came from an ingenious idea, aiding the predictions for physical problems using differential equations, it exhibits some pathological behavior during training, especially when the differential equation high-frequency solutions[15, 16]. This behavior is related to stiffness in the gradient flow dynamics of the PINNs loss functions, which leads to unstable convergence for gradient-based optimization algorithms [15, 16].

1.2 Contributions

Chapter 2 describes a novel approach to train GANs, which is capable of generating labeled genetic datasets using a small labeled dataset and a larger unlabeled dataset, exploiting concepts of semi-supervised learning and data augmentation to create a new approach to deal with the limited labeled data available to researchers. This approach may benefit studies with limited datasets.

Additionally, it also provides a self-aware classifier, capable of determine the prediction with a level of confidence.

In chapter 3, we present PSO-PINN, this method consists of a swarm optimization, which opens many possibilities for training PINNs. Firstly, the swarm is intrinsically an ensemble. Thus, it can take advantage of the many ensemble methods for building predictors and quantifying uncertainty. A more considerable degree of agreement among the PINNs leads to minor variance and a higher degree of confidence in the predicted values. Second, by design, the proposed algorithm is highly parallelizable, allowing massive computation and distributed processing. Also, the proposed method does not rely exclusively on the backpropagation of the loss gradients, avoiding the pathological behavior of the gradient descent methods when training PINNs.

In the next chapter 4, we expand the possibilities of PSO-PINN proposing two novel training strategies for swarm training. First, we propose the Multi-Objective PSO-PINN, as a tool for assess the problem and define how the prior information and the available data would best benefit the training. Although the PINN method offers a certain level of flexibility to choose where the model would be in a range starting as a strict mathematical PDE solver to a more yielding data-driven approach, it is remarkably difficulty to find this particular configuration. Until now, this property was partially explored, most often relying on the weighting of the total loss function, usually done in a trial-and-failure basis. The approach described here yields a set of Pareto non-dominated solutions, where the better one can be chosen based on how well-defined the terms are.

In chapter 4, we also propose the Multi-Modal PSO-PINN algorithm. One of the advantages of using a swarm-based optimization algorithm like PSO-PINN is the possibility of using ensemble methods to produce better solutions also quantify the uncertainty related to such solution. But for the specific case of Machine Learning, full convergence might be a bad after-training situation. For most of the cases the training does not happen through the entire possible cases, but through a limited dataset. In such a scenario the fully-converged population would be nothing but an overfitted model. Thus, the idea behind the Multi-Modal PSO-PINN is to enforce the optimization process to increase the diversity of the solution. This is done by splitting the swarm into multiple sub swarms

and performing parallel searches for local optima. The improved diversity in the solution leads to better generalization.

1.3 Organization

This document consists of five major chapters. This first chapter gives the reader an introductory overview of the subject, contextualization, motivations, and contributions. Followed by three self-contained papers reporting the advances and contributions achieved so far, and lastly, the final chapter discusses the work done and the prospective investigations.

Chapter 2, titled *A Semi-Supervised Generative Adversarial Network for Prediction of Genetic Disease Outcomes*, addresses an innovative method to train adversarial networks based on a semi-supervised approach. In addition to being a reliable classifier, this technique may be used to generate large synthetic genetic data sets based on a small amount of labeled data and a large amount of unlabeled data (such configuration is not uncommon, especially in the genetic field). This work was published in the 2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (IEEE-MLSP)

Chapter 3 is titled *PSO-PINN: Physics-Informed Neural Networks Trained with Particle Swarm Optimization*. This work was submitted to the PINNs special issue of the IEEE Transactions on Artificial Intelligence, and it is currently under review. This chapter describes a novel optimization scheme for PINNs using swarm optimization, gradient descent, and ensembles techniques. This novel technique affords the possibility of robust predictions with quantified uncertainty.

Chapter 4, *Extending the PSO-PINN: The Multi-Objective and Multi-Modal approaches*, introduces two new approaches for PSO-PINN training, namely, The Multi-Objective and the Multi-Modal PSO-PINN algorithms. These two techniques expanded the possibilities for the PSO-PINN while consistently increasing the accuracy and reducing the uncertainty of the solutions. Also, they open new implementation possibilities, specially for distributed training strategies.

Finally, Chapter 5 contains a brief reflection concerning the proposed contributions, and the impacts and limitations of the proposed techniques. It also discusses future work and ramifications for these techniques.

2. A SEMI-SUPERVISED GENERATIVE ADVERSARIAL NETWORK FOR PREDICTION OF GENETIC DISEASE OUTCOMES

2.1 Abstract

For most diseases, building large databases of labeled genetic data is an expensive and time-demanding task. To address this, we introduce genetic Generative Adversarial Networks (gGAN), a semi-supervised approach based on an innovative GAN architecture to create large synthetic genetic data sets starting with a small amount of labeled data and a large amount of unlabeled data. Our goal is to create a mechanism able to increase the sample size of the labeled data and generalize learning over different populations while keeping awareness of the quality of its own predictions. The proposed model achieved satisfactory results using real genetic data from different datasets and populations, in which the test populations may not have the same genetic profiles. The proposed model is self-aware and capable of determining whether a new genetic profile has enough compatibility with the data on which the network was trained and is thus suitable for prediction. The code and datasets used can be found at `https://github.com/caio-davi/gGAN`.

2.2 Introduction

Creating large genetic datasets is an expensive and time-demanding task in the case of most diseases, as it involves patient recruitment, laboratory work, sequencing equipment, and bioinformatics expertise. In many instances, data is classified as Protected Health Information (PHI)[17], making it difficult to obtain and to handle. There is also a range of intrinsic characteristics within this domain that increase the complexity of archiving and handling this data. Firstly, it may be difficult to obtain patients from the right population to extract the genetic material from. Secondly, maintaining the quality of the analysis is not a trivial process due to biological and technical noise, which varies between laboratories and/or between batches.

*Reprinted with permission from "A Semi-Supervised Generative Adversarial Network for Prediction of Genetic Disease Outcomes." by Caio Davi and Ulisses Braga-Neto. 2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP). IEEE, 2021.

With current advancements in Machine Learning algorithms applied to human genetics [18, 19, 20], large datasets are increasingly needed in order to support the use and analysis of a variety of data-intensive algorithms. Therefore, new mechanisms that are capable of generating coherent synthetic genetic data would be a remarkable advancement within this field.

However, genetic data on neglected diseases is notoriously scarce. This is the case of Dengue disease, a viral disease most commonly found in tropical areas of under-developed and developing countries. Dengue is a global public health concern with an annual global incidence of 390 million cases[21]. There is evidence supporting that the disease progression is genetically influenced[22, 23, 24, 25], but, as mentioned previously, genetic data on Dengue is scarce.

Recently, we employed machine learning algorithms to find associations between profiles of single-nucleotide polymorphisms (SNPs) and disease severity using a small data set from a cohort of patients in Recife, Brazil [26]. In this paper, we propose gGAN, an innovative Generative Adversarial Network (GAN) architecture to create large synthetic genetic data starting with the small amount of labeled data used in the previous study and a large amount of unlabeled data from the human 1000 genome project. This leads to a semi-supervised classifier that is able to predict the genetic predisposition to severe dengue of a patient, even if their genetic profile is from a different population than the one originally used to train the classifier (in this case, a Brazilian population). The classifier is self-aware in that one of the outputs of the proposed GAN is able to discriminate whether a given test genetic profile is sufficiently compatible with the genetic profiles in the training population. Even though we use dengue to demonstrate our method, it is entirely general and can be applied to other diseases.

2.3 Background

Genome-wide association studies (GWAS) are a powerful tool for identifying genetic influences in diseases, particularly for conditions influenced by a variety of genes and environmental factors at once [27]. These techniques are historically used for finding genetic differences in case-controlled studies for particular diseases.

One of the biggest concerns related to genetic studies is the size of the patient cohort used in

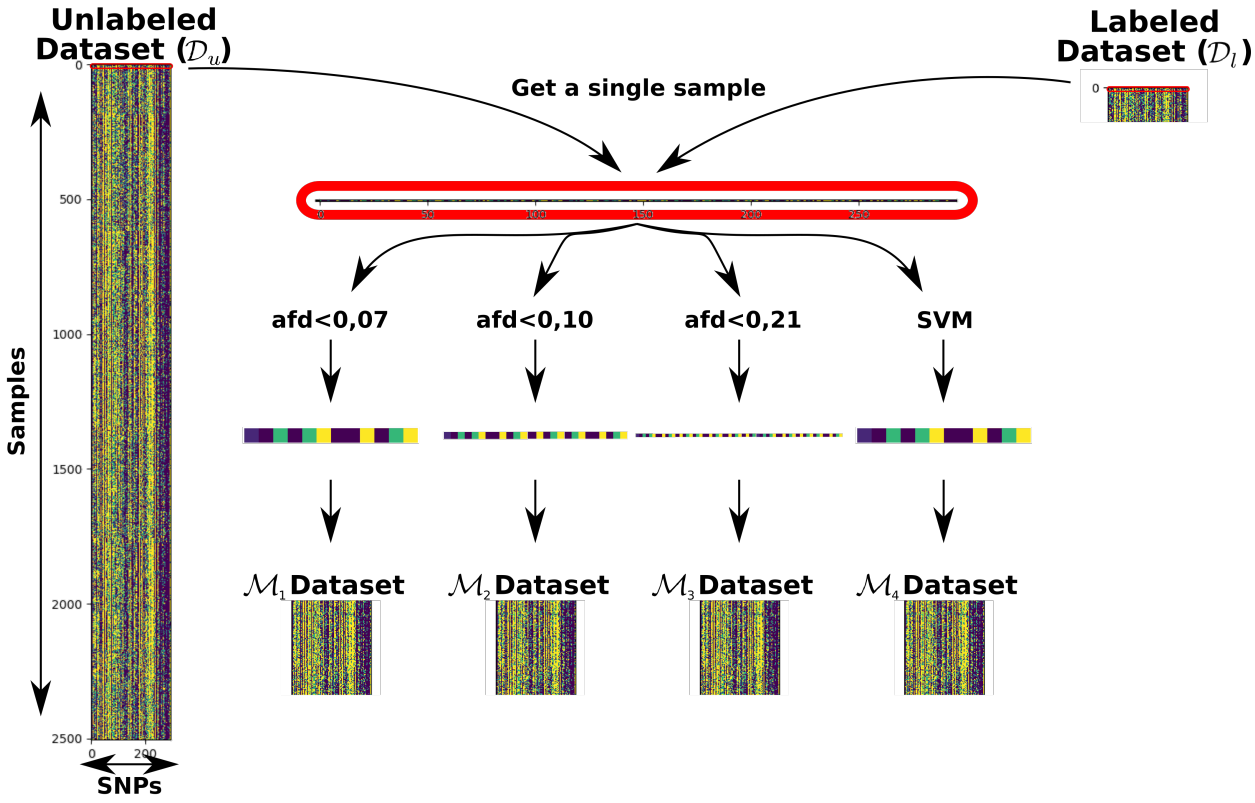

Figure 2.1: Data preprocessing procedure. Each line in each dataset represents a genetic profile of a patient. The SNPs are selected according to an Allelic Frequency Distance (AFD) threshold and the SVM-RFE algorithm. In this figure, AFD thresholds of 0.07, 0.10, 0.21 are depicted.

the experiments. As mentioned previously, obtaining a patient cohort is a hard and expensive task. Moreover, the discrepancy between the size of the cohort and the number of features is a frequent burden. This occurs due to the large number of Single Nucleotide Polymorphisms (SNPs) in the human genome, which, on average, occur almost once in every 1,000 nucleotides. This means there are roughly 4 to 5 million SNPs in a person's genome. This factor can easily lead to overfitting due to the curse of dimensionality [28]. To overcome this, feature selection becomes necessary.

Dengue disease is a global public health concern caused by the Dengue Virus (DENV), a positive-sense RNA virus from the *Flaviviridae* family. According to the 2009 World Health Organization (WHO) Dengue classification guideline[29], symptomatic infection by DENV can range from a mild disease know as Dengue Fever (DF), to a more severe disease known as Severe

Dengue (SD), which displays hemorrhagic complications that may eventually culminate in the death of the host individual.

There is evidence that some individuals may be genetically predisposed to some prognostic of the Dengue disease [22, 23]. More specifically, studies have found associations between the genetic expression, mostly in the innate and in the adaptive response pathways, and the Dengue infection phenotype [30, 31, 32, 33, 34]. Also, there are efforts to use Machine Learning techniques to find associations between static genetic profiles and dengue severity [35, 26].

Generative Adversarial Networks (GAN) [36] are a promising ML approach to generate synthetic data. GANs are commonly used to generate synthetic images from a known training set. However, there are applications of GANs in other domains[37], including genetics[38]. There is also a report of a semi-supervised approach to GANs [39], for classification of the MNIST dataset [40].

2.4 Methods

This section describes the steps performed to create the model. We begin with pre-processing of our datasets, then describe the model itself, and finally how it was trained. The tests and results are described in the next section.

2.4.1 Data Preprocessing

We used two primary datasets from real genetic data available to the research community. The first is a dataset labeled with the Dengue Infection phenotype in human subjects. Patients with dengue-related symptoms were screened from three hospitals in the city of Recife, Brazil; this population constitutes the domain $\mathcal{D}_l$. The first dataset, denoted as $\mathcal{X}_l$, was extracted from this population, and it consists of 102 patient genotypes measured at 322 loci polymorphisms. The methods of extraction and genotyping and cohort description for the labeled dataset can be found in [41].

The second dataset contains unlabeled genetic data from Phase 3 of the 1000 Genomes Project[42] (which consists of 2504 individuals genotyped for more than 84 million variants). The 1000 Genomes Project maps genetic information from populations around the world; we call

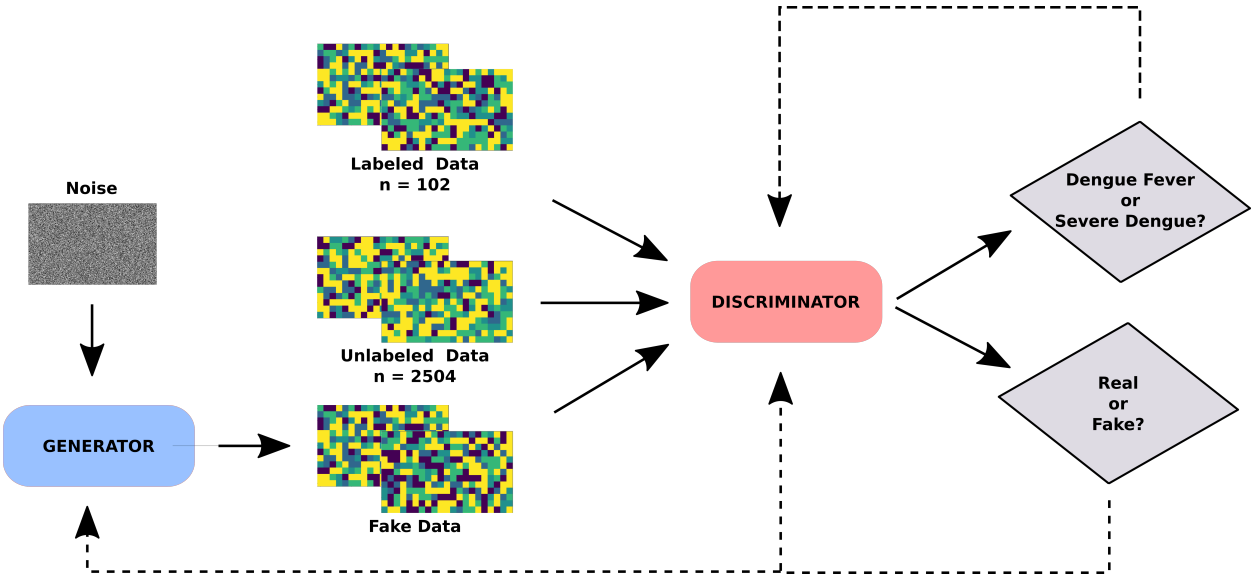

Figure 2.2: Model architecture. The discriminator network is fed two datasets (labeled dataset (n=102), unlabeled dataset (n=2504)) and the synthetic genetic profiles from the generator network. The discriminator training consists of two phases, the first using real unlabeled data and synthetic data, and the second using labeled data.

this population $\mathcal{D}_u$. The unlabeled dataset $\mathcal{X}_u$ consists of the same set of 322 loci in the labeled dataset, extracted from the 1000 Genomes Collection. An important step in the manipulation of this kind of data is that we estimated each of the allelic frequencies[43]. Different populations tend to have different frequencies due to their diverse genomic background[44]. These differences may affect the performance of the discriminator network, which would try to learn from two datasets with different frequency distributions. With this in mind, we compared these frequencies between the labeled dataset and the unlabeled dataset. We measured the frequencies for each allele of each variant in both datasets and assume the largest difference between them as their frequency proximity. The Allelic Frequency Distance (AFD) between two domains is defined as

$$\text{AFD} \ = \max(f_i(\mathcal{X}_l) - f_i(\mathcal{X}_u)) \, . \tag{2.1}$$

In equation (1), f_i is the frequency of a specific allele i within the n_i possible values for each nucleotide in population $\mathcal{X}$.

After this, we established certain thresholds to sampling the data and used them to create different INPUT configurations to the network. We used thresholds of 0.7, 0.1, and 0.21 resulting in subsets of 12, 25, and 96 SNPs, as shown in Figure 2.1. Those subsets were used to build the models $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$, respectively.

We also used a subset of SNPs chosen in [26]. In that paper, a Support Vector Machine was used to find the best SNPs to discriminate Severe Dengue. As a result, 13 SNPs were found as the most adequate ones to be used to classify the severe form of the disease. One of those SNPs was not genotyped in the 1000 Genome Project, so it was removed from this set, resulting in 12 SNPs for model $\mathcal{M}_4$.

2.4.2 gGAN Architecture

We modified the traditional GAN architecture to work with both labeled and unlabeled data. In the traditional GAN model, the discriminator has only one output, which classifies the input as real or fake (synthetic). In our case, we want to predict the label of a test genetic profile ("Dengue Fever" or "Severe Dengue") in addition to classifying between real or fake. In order to archive this, we added a new output to the discriminator network. This is illustrated in Figure 2.2. Tables 2.1 and 2.2 give a detailed overview of the layers of the discriminator and generator networks in the implementation of the proposed paradigm. All layers and operators in these tables are described in Tensorflow API notation. Notice that all generators have inputs of the same length. The generators are always fed with random noise, and in this case, we opted to keep the same size for all the models. The Upsample layer simply increases the size of the layer by duplicating it.

During training, the discriminator network weights are updated using the labeled and unlabeled data. This update occurs in turns (labeled and then unlabeled) for each epoch of training. First the network $\mathcal{M}_{m,disc}$, for $m = 1, \ldots, 4$, is exposed to the labeled samples in $\mathcal{D}_l$, and a loss $\mathcal{L}(\mathcal{D}_l)$ is calculated using the binary cross-entropy [45]:

$$\mathcal{L}(\mathcal{D}_l) \;=\; -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=0}^{1} y_{ij}\log\hat{y}_{ij}\,, \tag{2.2}$$

Table 2.1: Architecture of the discriminator networks.

$\mathcal{M}_{1,disc}$	$\mathcal{M}_{2,disc}$	$\mathcal{M}_{3,disc}$	$\mathcal{M}_{4,disc}$
input(12,1)	input(25,1)	input(96,1)	input(12,1)
ReLu(12,24)	ReLu(25,50)	ReLu(96,180)	ReLu(12,24)
ReLu(12,48)	ReLu(25,100)	ReLu(96,360)	ReLu(12,48)
ReLu(12,24)	ReLu(25,50)	ReLu(96,180)	ReLu(12,24)
Flatten	Flatten	Flatten	Flatten
Dropout	Dropout	Dropout	Dropout
Output$_l(2,1)$	Output$_l(2,1)$	Output$_l(2,1)$	Output$_l(2,1)$
Output$_u(2,1)$	Output$_u(2,1)$	Output$_u(2,1)$	Output$_u(2,1)$

Table 2.2: Architecture of the generator networks.

$\mathcal{M}_{1,gen}$	$\mathcal{M}_{2,gen}$	$\mathcal{M}_{3,gen}$	$\mathcal{M}_{4,gen}$
input(100)	input(100)	input(100)	input(100)
ReLU(24)	ReLU(50)	ReLU(90)	ReLU(24)
UpSample(48)	UpSample(100)	UpSample(180)	UpSample(48)
ReLU(48)	ReLU(100)	ReLU(180)	ReLU(48)
UpSample(96)	UpSample(200)	UpSample(360)	UpSample(96)
ReLu(12)	ReLu(25)	ReLu(96)	ReLu(12)
Reshape(12,1)	Reshape(25,1)	Reshape(96,1)	Reshape(12,1)

where N is the batch size, (y_{i0}, y_{i1}) is a one-hot encoding for the training label ("Dengue Fever" or "Severe Dengue"), and $(\hat{y}_{i0}, \hat{y}_{i1})$ is the softmax output of the discriminator corresponding to training profile $\mathbf{x}_i$, for $i = 1, \ldots, N$. Next, the weights are updated using profiles in $\mathcal{D}_u$ using the loss function:

$$\mathcal{L}(\mathcal{D}_u, \mathcal{D}_g) \;=\; -\frac{1}{M}\sum_{i=1}^{M}\sum_{j=0}^{1} y_{ij}\log\hat{y}_{ij} \tag{2.3}$$

where M is the batch size, (y_{i0}, y_{i1}) is a one-hot encoding for the training label ("fake" or "real"), and $(\hat{y}_{i0}, \hat{y}_{i1})$ is the softmax output of the discriminator corresponding to training profile $\mathbf{x}_i$, for $i = 1, \ldots, M$. The training batch is composed by unlabeled real samples ($\mathcal{D}_u$) and synthetic samples ($\mathcal{D}_g$) created by the generator model.

On the other hand, the weights of the generator network are updated using random noise as the input data set, denoted by $\mathcal{D}_r$. A synthetic sample will be produced using this random input and then the discriminator will classify it as real or fake data. Based on this answer, the binary cross-entropy is used to calculate the loss:

$$\mathcal{L}(\mathcal{D}_r) = -\frac{1}{P}\sum_{i=1}^{P}\sum_{j=0}^{1} y_{ij}\log\hat{y}_{ij} \tag{2.4}$$

where P is the batch size, $(y_{i0}, y_{i1}) = (0, 1)$ is a one-hot encoding for the training label (all profiles are "fake"), and $(\hat{y}_{i0}, \hat{y}_{i1})$ is the softmax output of the discriminator corresponding to the random genetic profile $\mathbf{x}_i$, for $i = 1, \ldots, P$.

2.5 Results

The dataset from the labeled domain $\mathcal{D}_l$ was split into training and testing following a proportion of 80% for training and the remaining 20% for testing. This resulted in a training dataset $\mathcal{X}_{l_{train}}$ containing 72 labeled genetic profiles, and testing dataset $\mathcal{X}_{l_{test}}$ with the remaining 20 labeled profiles. We used the whole unlabeled dataset to train during the unlabeled step and selected 20% as a testing dataset $\mathcal{X}_{u_{test}}$, for a total of 502 unlabeled testing profiles.

In our implementation, the batch size for training the supervised discriminator was $N = 10$, selected randomly from the labeled dataset each time. Also, 50 randomly selected profiles from the unlabeled dataset with an additional 50 profiles created by the generator network were used to train the unsupervised discriminator, for a batch size $M = 100$. Finally, $P = 100$ random genetic profiles were used to train the generator model.

We followed two testing procedures to perform accuracy estimation. The first, denoted by T_1, simply measures the accuracy of each of the discriminator outputs, independently. The second, denoted by T_2, aggregates both of the discriminator outputs. Testing procedure T_1 calculates the labeled accuracy $Acc_{(\mathcal{D}_l)}$ and unlabeled accuracy $Acc_{(\mathcal{D}_u)}$ on the testing data sets $\mathcal{X}_{l_{test}}$ and $\mathcal{X}_{u_{test}}$, respectively. Testing procedure T_2 uses the testing dataset $\mathcal{X}_{l_{test}}$ to simulate a deployment scenario. It is a two-step procedure: first, it checks if the discriminator is able to recognize the test profile

as a real profile, and calculates the accuracy $Acc_{1(\mathcal{D}_l)}$ on the testing dataset. After that, it verifies if the labeled output matches the actual data label, producing an accuracy estimate $Acc_{2(\mathcal{D}_l)}$ using only the correctly predicted profiles in the first step. The accuracy rates obtained in our experiments according to testing procedures T_1 and T_2 are displayed in Table 2.3.

Table 2.3: Accuracy rates according to testing procedures T_1 and T_2.

Models	T_1		T_2	
	$Acc_{(\mathcal{D}_l)}$	$Acc_{(\mathcal{D}_u)}$	$Acc_{1(\mathcal{D}_l)}$	$Acc_{2(\mathcal{D}_l)}$
$\mathcal{M}_1$(AFD <0.07)	0.6111	0.8719	0.9444	0.6470
$\mathcal{M}_2$(AFD <0.10)	0.6999	0.9918	1.0	0.6999
$\mathcal{M}_3$(AFD <0.21)	0.5263	0.9919	0.9473	0.5555
$\mathcal{M}_4$(SVM-RFE)	0.9375	0.9430	1.0	0.9375

The first two columns in Table 2.3 from testing procedure T_1 show good results for the unlabeled output for all models, meaning that gGAN has good performance when differentiating between real and synthetic data. Models $\mathcal{M}_2$, and $\mathcal{M}_3$ correctly classified almost all of the test profiles in the $\mathcal{X}_{u_{test}}$ dataset (n=500). Model $\mathcal{M}_4$, using the SNPs obtained by the SVM-RFE method, showed superior results, obtaining 94.3% accuracy when discriminating test profiles in $\mathcal{X}_{u_{test}}$ as real or synthetic, and nearly 93.75% accuracy on the labeled testing profiles in $\mathcal{X}_{l_{test}}$ (these results are similar to the ones obtained in [26]).

The next two columns in Table 2.3 from testing procedure T_2 show that all models built with AFD SNPs display high accuracy on testing profiles in $\mathcal{X}_{l_{test}}$, which reinforces the AFD hypothesis for selecting the best SNPs. On the other hand, AFD ignores the relationship between the SNPs and the dengue phenotype, as it only considers the similarity between SNPs. This can be noticed in the results for $Acc_{2(\mathcal{D}_l)}$. Model $\mathcal{M}_4$ is far more accurate in this respect, which confirms that the SNPs used by $\mathcal{M}_4$ have information on the Dengue phenotype.

2.6 Conclusion

In this paper, we introduced a novel approach to train GANs, which is capable of generating labeled genetic datasets using a small labeled dataset and a larger unlabeled dataset, exploiting concepts of semi-supervised learning and data augmentation to create a new approach to deal with the limited labeled data available to researchers.

The proposed gGAN architecture modifies the usual perspective on GANs: instead of a tool solely designed to generate labeled genetic profiles, it also provides a classifier with a level of confidence. Previous ML methods to classify diseases based on genetic information are not aware of data coming from new populations. With gGAN, one may see the unlabeled output as a self-aware output. While the labeled output of the discriminator predicts whether the individual with the corresponding genetic profile is likely to develop Severe Dengue, the unlabeled output predicts whether the genetic profile is real or synthetic. Given a real genetic profile, if the gGAN classifies it as not real (i.e., synthetic), we can infer that it was most likely never seen before and thus is not appropriate for prediction by the model.

The differences within the allelic frequencies between populations is a challenge when one is using multiples datasets from different populations. Indeed, even the 1000 Genome Dataset[42] is composed of different populations and is therefore subject to a variety of allelic frequencies itself. Due to limitations on the size of our datasets, we opted to filter the datasets using the AFD, but other approaches would have also been possible if a good sample of both labeled and unlabeled data were available for a population. Since we utilized the labeled dataset from Recife, Brazil [41], and there is no large unlabeled dataset for this specific population, we had to construct our own AFD thresholds. Another possible approach to this issue would be to use Style Transfer to somehow make the synthetic data with frequencies between those of the two datasets.

Nevertheless, our methods produced good results using the SNPs selected by [26] using the same labeled dataset we are using here. In their work, they chose the 13 most representative SNPs to classify Severe Dengue. Indeed, we had good results using those SNPs in the labeled phase of the training and test. However, we also had surprisingly good results using the unlabeled output of

the discriminator network, which lead us to believe that the discriminator was able to generalize well, even with the allelic frequencies discrepancies.

As expected, the results of the model $\mathcal{M}_4$ in the tests using $\mathcal{X}_{l_{test}}$ were far better than the models trained using the AFD method to select SNPs. Also, the model $\mathcal{M}_2$ had a fair accuracy (almost 70%) if we think of it as a triage tool for prognosis. Those results are even more acceptable during disease outbreaks, when more specific tests are commonly scarce and expensive. Another important aspect is that it can be applied at any stage of the disease, or even before infection, and can use a broad choice of human sample tissue, since it is only based on genetic profiles.

The COVID-19 pandemic is a good example of a scenario where the method described here could bring great benefits to society. Our approach could be used to help triage patients with a level of confidence. It could also be used to avoid exposure of healthcare professionals who are sensitive to the disease. The proposed method could be a powerful tool to help public health efforts against a new outbreak.

3. PSO-PINN: PHYSICS-INFORMED NEURAL NETWORKS TRAINED WITH PARTICLE SWARM OPTIMIZATION

3.1 Abstract

Physics-informed neural networks (PINN) have recently emerged as a promising application of deep learning in a wide range of engineering and scientific problems based on partial differential equation (PDE) models. However, evidence shows that PINN training by gradient descent displays pathologies that often prevent convergence when solving PDEs with irregular solutions. In this paper, we propose the use of a particle swarm optimization (PSO) approach to train PINNs. The resulting PSO-PINN algorithm not only mitigates the undesired behaviors of PINNs trained with standard gradient descent but also presents an ensemble approach to PINN that affords the possibility of robust predictions with quantified uncertainty. Comprehensive experimental results show that PSO-PINN, using a modified PSO algorithm with a behavioral coefficient schedule, outperforms other PSO variants for training PINNs, as well as PINN ensembles trained with standard ADAM.

3.2 Introduction

Physics-informed machine learning is an emerging area that promises to have a lasting impact in science and engineering. Physics-informed neural networks (PINNs) [46], in particular, have shown remarkable success in a variety of problems modeled by partial differential equations [47, 48]. PINNs exploit the universal approximation capabilities of neural networks [49]. Unlike traditional neural networks, PINNs employ a multi-part loss function containing data-fitting and PDE residual components. And differently from traditional numerical methods, PINNs are meshless, producing a solution over an entire, possibly irregular, domain, rather than on a pre-specified grid of points.

Standard gradient descent tools already in widespread use for training deep neural networks, such as stochastic gradient [50] and ADAM [8], were promptly adopted as the methods of choice to train PINN. However, an accumulating body of evidence shows that gradient descent exhibits pathological behavior when training PINNs, especially when the differential equation solution

contains irregular and fast varying features [15, 16]. Numerous approaches have been proposed to mitigate this undesirable behavior, including weighting the loss function [16, 51, 52], domain decomposition[53, 54], and changes to the neural network architecture [15]. Recent work attributed this behavior to stiffness in the gradient flow dynamics of the PINNs loss functions, which leads to unstable convergence for gradient-based optimization algorithms [15, 16].

In this work, we propose to address this issue by moving away from pure gradient descent by employing particle swarm optimization (PSO) [55], a metaheuristic algorithm used in numerous applications, including neural network training [56, 57, 58, 59, 60]. Indeed, the original paper on PSO [55] was motivated in part by neural network training. The resulting PSO-PINN algorithm not only mitigates the pathologies associated with grandient-based optimization, but also produces a diverse ensemble of neural networks [61], which enables the application of ensemble methods to PINNs, such as variance reduction[62] and uncertainty quantification[63]. Indeed, ensemble methods have become popular in deep learning as they combine the outputs of a diverse set of neural networks, thus reducing prediction variance and producing uncertainty estimates [64, 65, 66, 67]. To our knowledge, PSO-PINN is the first algorithm for training PINNs based on swarm optimization. However, PSO-PINN is not the only ensemble approach to PINNs; the ensemble approach was used to progressively train PINNs forward in time in a recent publication [68].

In this paper, we consider and contrast three variants of the PSO algorithm to train PINNs: the PSO method in its original form [55], a variant called PSO-BP (PSO with Back-Propagation), which adds a gradient descent component to the particle velocity vectors [69], and PSO-BP-CD (PSO-BP with Coefficient Decay), a proposed modification of PSO-BP that includes a decreasing schedule for the behavioral coefficients. PSO-BP-CD puts more weight on the gradient descent component near the end of training, when the swarm should have already converged on a good local optimum. Experimental results with several ODE and PDE benchmarks show that PSO-PINN, using the proposed PSO-BP-CD algorithm, consistently outperforms the other PSO variants for training PINNs, as well as PINN ensembles trained with standard ADAM [8], in diverse scenarios under various parameter settings. These results demonstrate the potential of PSO-PINN in providing

accurate prediction with quantified uncertainty.

This paper is organized as follows. Section 3.3 provides a brief overview of deep neural networks, physics-informed neural networks, and particle swarm optimization. Section 3.4 introduces the PSO-PINN algorithm based on three variants of PSO, plain, PSO-BP, and PSO-BP-CD, whereas Section 3.5 studies PSO-PINN through experimental results using various ODE and PDE benchmarks. Section 3.6 provides concluding remarks and future directions.

3.3 Background

In this section, we define PINNs and describe the PSO-BP optimization algorithm.

3.3.1 Deep Neural Networks

The feed-forward fully-connected neural network is the basic architecture used in deep learning algorithms[70]. A fully-connected neural network with L layers is a function $f_\theta : \mathbb{R}^d \to \mathbb{R}^k$ described by

$$f_\theta(\mathbf{x}) = W^{[L-1]}\sigma \circ (\cdots \sigma \circ (W^{[0]}\mathbf{x} + b^{[0]}) + \cdots) + b^{[L-1]} \tag{3.1}$$

where σ is an entry-wise activation function, $W^{[l]}$ and $b^{[l]}$ are respectively the weight matrices and the bias corresponding to each layer l, and θ is the set of weights and biases:

$$\theta = (W^{[0]}, \cdots, W^{[L-1]}, b^{[0]}, \cdots, b^{[L-1]}). \tag{3.2}$$

Among popular choices for the activation function are the sigmoid function, the hyperbolic tangent function (tanh), and the rectified linear unit (ReLU) [71]. To fit the neural network $f_\theta(\mathbf{x})$ to data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, one minimizes a suitable loss function. A popular choice in traditional deep learning is the mean square error (MSE):

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} ||f_\theta(\mathbf{x}_i) - \mathbf{y}_i||^2. \tag{3.3}$$

3.3.2 Physics-Informed Neural Networks

Consider a non-linear differential equation of the general form:

$$\begin{aligned} \mathcal{N}[u(\mathbf{x})] &= g(\mathbf{x}), \quad \mathbf{x} \in \Omega, \\ u(\mathbf{x}) &= h(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \end{aligned} \tag{3.4}$$

where $\Omega \subset \mathbb{R}^d$, $u : \Omega \to \mathbb{R}^k$, $\mathcal{N}[\cdot]$ is a differential operator, and $g, h : \Omega \to \mathbb{R}^k$ specify forcing and boundary conditions, respectively. We employ a neural network $f_\theta(\mathbf{x})$ to approximate the unknown function $u(\mathbf{x})$. The PDE is enforced through the loss function

$$\mathcal{L}_r = \frac{1}{n_r} \sum_{i=1}^{n_r} ||\mathcal{N}[f_\theta(\mathbf{x}_i^r)] - g(\mathbf{x}_i^r)||^2, \tag{3.5}$$

where $\{\mathbf{x}_i^r\}_{i=1}^{n_r} \subset \Omega$ are collocation points, and $\mathcal{N}[f_\theta(\mathbf{x})]$ is computed accurately using automatic differentiation methods [72]. The neural network is fitted to the initial and boundary condition $h(\mathbf{x})$ using a traditional data-fitting loss function:

$$\mathcal{L}_b = \frac{1}{n_b} \sum_{i=1}^{n_b} ||f_\theta(\mathbf{x}_i^b) - h(\mathbf{x}_i^b)||^2 \tag{3.6}$$

where $\{\mathbf{x}_i^b\}_{i=1}^{n_b} \subset \partial\Omega$ are initial and boundary points. If there are experimental or simulation data $\{(\mathbf{x}_i^d, \mathbf{y}_i^d)\}_{i=1}^{n_d}$ available, they may be included in the usual way through the loss:

$$\mathcal{L}_d = \frac{1}{n_d} \sum_{i=1}^{n_d} ||f_\theta(\mathbf{x}_i^d) - \mathbf{y}_i^d||^2. \tag{3.7}$$

The previous multi-objective optimization problem is typically solved by simple linear scalarization, in which case the PINN is trained by minimizing the total loss function $\mathcal{L}$:

$$\mathcal{L} = \mathcal{L}_r + \mathcal{L}_b + \mathcal{L}_d. \tag{3.8}$$

This framework is easily extended to more complex initial and boundary conditions involving

derivatives of $u(\mathbf{x})$.

3.3.3 Particle Swarm Optimization

The Particle Swarm Optimization (PSO) algorithm [73, 74] is a population-based stochastic optimization algorithm that emulates the swarm behavior of particles distributed in a n-dimensional search space [75]. Each individual in this swarm represents a candidate solution. At each iteration, the particles in the swarm exchange information and use it to update their positions. Particle θ^t at iteration t is guided by a velocity determined by three factors: its own velocity inertia βv^t, its best-known position p_{best} in the search-space, as well as the entire swarm's best-known position g_{best}:

$$v^{t+1} \;=\; \beta v^t + c_1 r_1 (p_{best} - \theta^t) + c_2 r_2 (g_{best} - \theta^t)\,, \tag{3.9}$$

where c_1 and c_2 are the cognitive and social coefficients, respectively, referred to jointly as the behavioral coefficients, and r_1 and r_2 are uniformly distributed random numbers in range $[0, 1)$. Then the particle position is updated as

$$\theta^{t+1} \;=\; \theta^t + v^{t+1}\,. \tag{3.10}$$

Many variations of the PSO algorithm were proposed over time. PSO can be combined with gradient descent to train neural networks, with the gradients computed efficiently by automatic differentiation (backpropagation), as was done in [69]. In this algorithm, which was called PSO-BP by the authors, the particle velocity has an additional component, namely, the gradient vector of the loss function$\nabla \mathcal{L}(\theta)$:

$$v^{t+1} \;=\; \beta v^t + c_1 r_1 (p_{best} - \theta^t) + c_2 r_2 (g_{best} - \theta^t) - \alpha \nabla \mathcal{L}(\theta^t)\,, \tag{3.11}$$

where α is a learning rate. Therefore, the gradient participates in the velocity magnitude and direction, by an amount that is specified by the learning rate, which helps guide the swarm to a good local optimum. See Figure 3.1 for an illustration of the PSO-BP update.

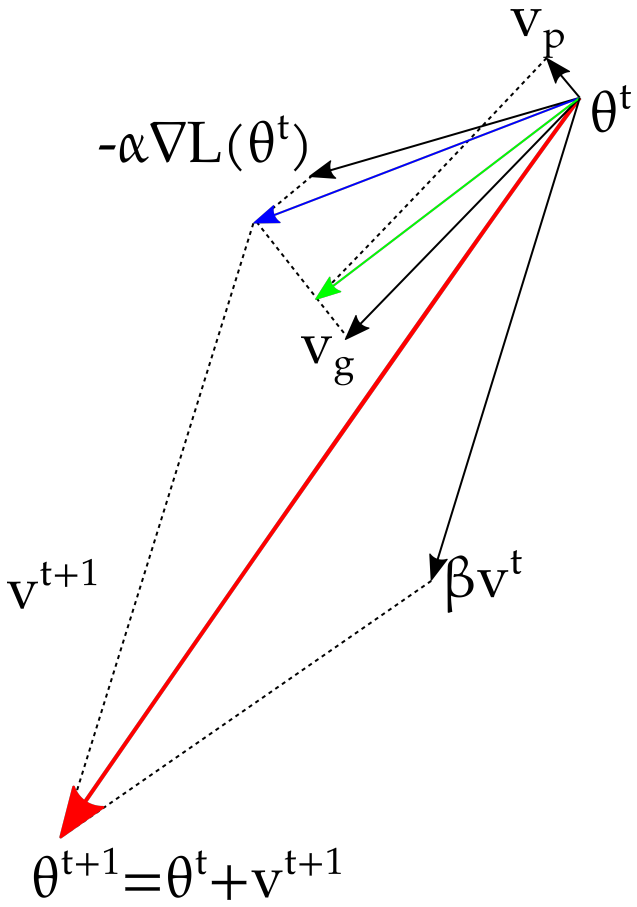

Figure 3.1: The PSO-BP update.

3.4 PSO-PINN Algorithm

In this section, we describe three variants of the PSO algorithm that we use to train PINNs, how to initialize them, and a few possible ways to summarize the ensemble results.

3.4.1 The PSO variants

We consider the following three variants of the PSO algorithm:

1. **PSO** - The algorithm in its original form, as described in (3.9) and (3.10).

2. **PSO-BP** - This version adds a component based on the gradient of the loss function, as described in (3.11).

3. **PSO-BP-CD** In this variant, we introduce a behavioral coefficient schedule. The behavioral components c_1 and c_2 in PSO-BP-CD decay linearly according to the iteration number as follows:

$$c_k^{t+1} = c_k^t - \frac{2c_k^t}{n} \tag{3.12}$$

for $k = 1, 2$ and $t = 1, 2, \ldots, n$, where n is the total number of iterations.

We propose PSO-BP-CD as a variant of PSO-BP, which puts more weight on the gradient vector near the end of training, when the swarm should already have zeroed in on a good local optimum, and less communication between the particles is needed. This allows PSO-BP-CD to be more effective than the PSO-BP and pure PSO methods in training PINN ensembles, as will be seen in the experimental results section.

3.4.2 The Algorithm

The PSO-PINN swarm is an ensemble of PINN candidates. Each particle in the swarm is a vector θ containing the weights and bias of the corresponding PINN. The population Θ of all particles is randomly initialized, as will be detailed later in Section 3.5.2. The objective function $f_\theta(\cdot)$ is the total loss function in (3.8). The p_{best} vector should be initialized to the values in the original population Θ, as the particles know nothing but their initial point at this time. The g_{best} parameter is equal to the value of $\theta \in \Theta$ that achieves the minimum value of $f(\theta)$. After the initialization step, the updates to the swarm attempt to minimize the expected value of the objective function. Each particle velocity is updated following (3.11). Despite the use of the gradient update $\alpha\nabla\mathcal{L}(\theta)$ in the previous equation, in practice, the update is made using ADAM optimization [8]. After the velocity update, the particle locations (i.e., the network weights) are updated according to (3.10). The values of p_{best} and g_{best} are updated accordingly. The training loop runs until a maximum number of iterations is reached. The procedure is summarized in Algorithm 1.

3.4.3 The Ensemble Solution

Besides the fact that swarms are a well-established optimization method, they also provide an ensemble of solutions. Thus, we can take advantage of the various properties provided by ensembles.

Algorithm 1: PSO-PINN

Require: α: step size;
Require: β: inertia;
Require: c_1, c_2: behavioral coefficients;
Require: $\mathcal{L}$: total loss function for the PINN;
Initialize population Θ;
$p_{best}(i) \leftarrow \theta_i$, for $i = 1, \ldots, |\Theta|$;
$g_{best} \leftarrow \arg\min_{\theta \in p_{best}} \mathcal{L}(\theta)$;
for $t = 1, 2, \ldots,$ MAX **do**:
  for $i = 1, \ldots, |\Theta|$ **do**:
    $r_1, r_2 \leftarrow U(0, 1]$;
    $V = \beta V + c_1 r_1 (p_{best}(i) - \theta_i) + c_2 r_2 (g_{best} - \theta_i)$
    if BP:
      $V = V + \alpha \nabla \mathcal{L}(\theta_i)$
    $\theta_i = \theta_i + V$;
    if $\mathcal{L}(\theta_i) < \mathcal{L}(p_{best}(i))$:
      $p_{best}(i) \leftarrow \theta_i$
  end
  $g_{best} \leftarrow \arg\min_{\theta \in p_{best}} \mathcal{L}(\theta)$;
  if *coefficient_decay*:
    $c_1 = c_1 - \frac{2c_1}{t}$;
    $c_2 = c_2 - \frac{c_2}{t}$;
end

This category of learning methods combines several individual models to create a "collective" solution, which is expected to display improved performance and stability with respect to any of the individual models. There are different methods to seek the consensus decision in an ensemble [67]. The most straightforward one perhaps would be simply choosing the best-fitted model according to the loss function. Although this may yield a good model, it could fail due to overfitting. To avoid this, a better approach exploits the ensemble diversity through model averaging:

$$\hat{f}(\mathbf{x}) = \frac{1}{|\Theta|} \sum_{\theta_i \in \Theta} f_{\theta_i}(\mathbf{x}) \,. \tag{3.13}$$

Uncertainty of the prediction at each point $\mathbf{x}$ of the domain can be quantified naturally through the sample variance:

$$\hat{\sigma}^2(\mathbf{x}) = \frac{1}{|\Theta| - 1} \sum_{\theta_i \in \Theta} (f_{\theta_i}(\mathbf{x}) - \hat{f}(\mathbf{x}))^2. \tag{3.14}$$

3.5 Experimental Results

In this section, we study the performance of PSO-PINN empirically, using the three PSO variants discussed earlier, by means of several classical ODE and PDE benchmarks. In addition, PSO-PINN ensembles are compared to traditional ensembles of PINNs trained with ADAM gradient descent. All experiments employ fully-connected architectures, with the hyperbolic tangent activation function and Glorot initialization of the weights [76]. The main figure of merit used is the L_2 error:

$$L_2 \text{ error } = \frac{\sqrt{\sum_{i=1}^{N_U} |\hat{f}(\mathbf{x}_i) - U(\mathbf{x}_i)|^2}}{\sqrt{\sum_{i=1}^{N_U} |U(\mathbf{x}_i)|^2}}. \tag{3.15}$$

where $\hat{f}(\mathbf{x})$ is the prediction and $U(\mathbf{x})$ is the analytical solution or a high-fidelity approximation over a test mesh $\{\mathbf{x}_i\}_{i=1}^{N_U}$. All experiments in this section were performed using Tensorflow 2 [77].

3.5.1 A Simple PSO-PINN Example

Fist, we use the 1D Poisson equation to illustrate the PSO-PINN algorithm. Due to its simplicity, this example allows us to visualize the evolution of training and the results. The 1D Poisson equation considered here is:

$$\begin{aligned} &u_{xx} = g(x)\,, \quad x \in [0, 1]\,, \\ &u(0) = u(1) = 0\,. \end{aligned} \tag{3.16}$$

where $g(x) = -(2\pi)^2 \sin(2\pi x)$ is manufactured so that the solution is $u(x) = \sin(2\pi x)$. For this simple problem, we use a relatively shallow architecture consisting of 3 layers of 10 neurons. The PSO-PINN ensemble was composed by 50 particles and used the proposed PSO-BP-CD optimization method, with hyperparameter values $\alpha = 0.005$, $\beta = 0.9$, $c_1 = 0.08$, and $c_2 = 0.5$.

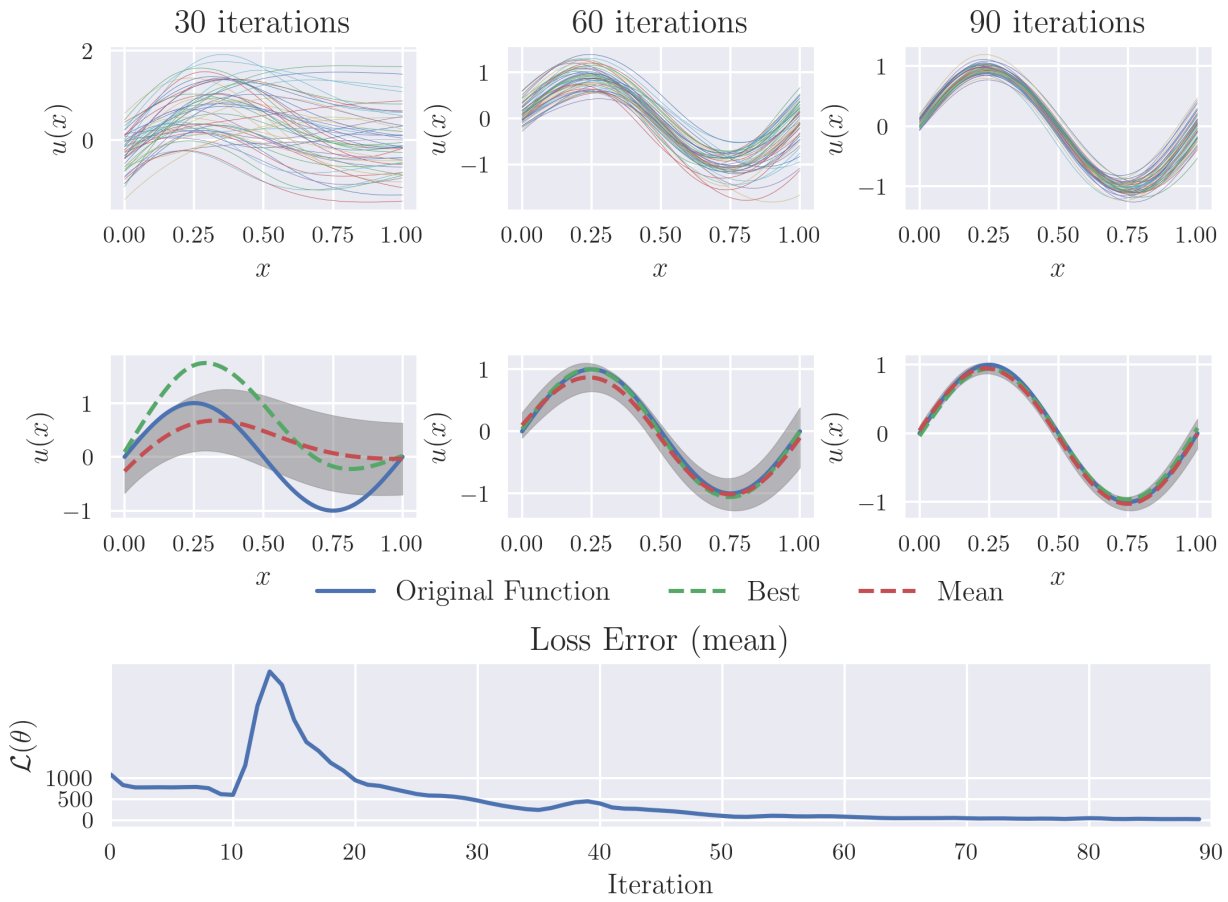

Figure 3.2: Poisson Equation - First row: solutions in the PSO-PINN ensemble as training progresses, using the PSO-BP-CD method. Second row: Best individual, mean and variance of PSO-PINN ensemble. Third row: PSO-PINN mean loss error. The grey band is the two-sided 1-standard deviation region.

We can see in Figure 3.2 the progression of training, and how the PSO-PINN swarm zeros in on the solution, rather quickly, after only 90 iterations. We can observe that, initially, the mean is off and the variance is large, which is the desired behavior, as the variance is supposed to quantify the uncertainty associated with the approximation. As training progresses, the swarm approaches a consensus, the mean converges to the solution, and the variance simultaneously shrinks to zero, indicating more confidence in the final result. The PSO-PINN swarm is thus accurate, with well-calibrated uncertainty quantification.

3.5.2 PSO-PINN Performance using Different PSO Variants

In this section, we use classical PDE benchmarks to study the performance of PSO-PINN using the three PSO variants described in 3.4.1. The number of training iterations was fixed at 2,000, while the architecture of the neural networks was fixed at 5 layers and 8 neurons per layer. The ensemble size was kept fixed at 100 neural networks. We kept the same number of initial condition points, boundary condition points and residual points for all experiments in this section. In all experiments, we used 1024 evenly spaced points for the initial condition, 512 evenly spaced points for the boundary conditions, and 1000 residual points randomly distributed over the solution domain using Latin hypercube sampling.

3.5.2.1 Advection Equation

The advection equation models the transport of a substance by bulk motion of a fluid. In this example, we are assuming a linear univariate advection equation [78]:

$$q_t + uq_x = 0 \tag{3.17}$$

where u is the constant velocity. The Riemman initial condition is

$$q(x,0) = \begin{cases} q_l, & 0 \leq x < x_0\,, \\ q_r, & x_0 < x \leq L\,. \end{cases} \tag{3.18}$$

This simple problem has as solution:

$$q(x,t) = \begin{cases} q_l, & 0 \leq x < x_0 + ut\,, \\ q_r, & x_0 + ut < x \leq L\,, \end{cases} \tag{3.19}$$

for $0 \leq t < (L - x_0)/u$. In other words, the initial discontinuity in concentration is simply advected to the left with constant speed u.

The parametrization used for the PSO algorithm was $\beta = 0.9$, $c_1 = 0.8$ and $c_2 = 0.5$. For

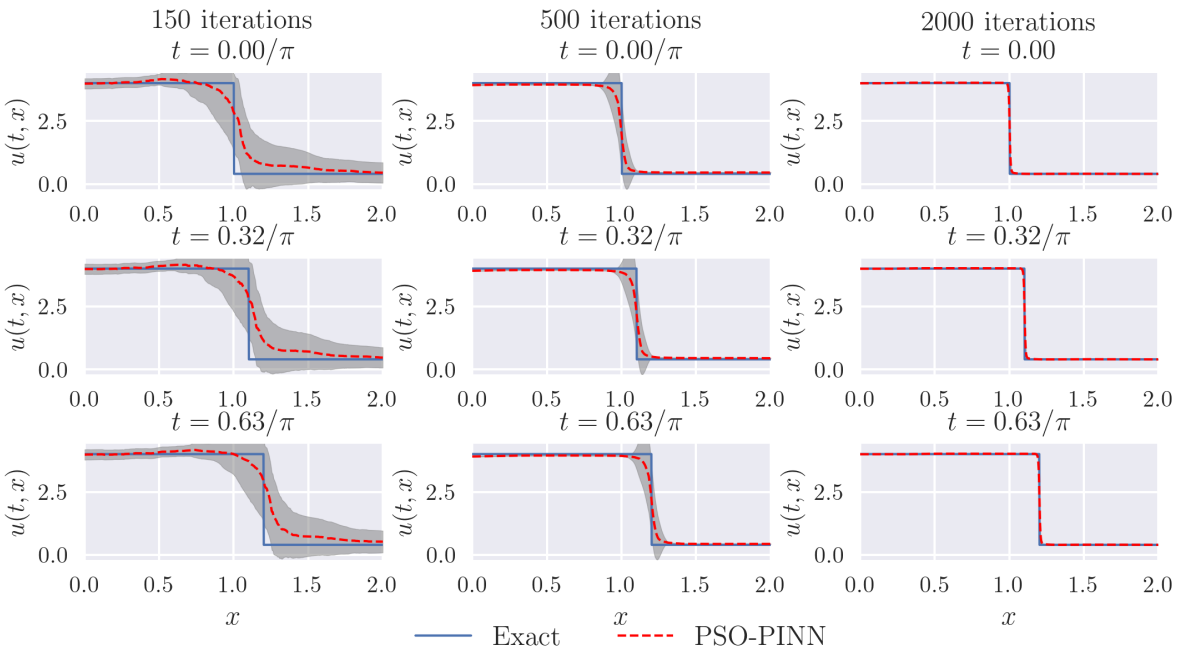

Figure 3.3: Advection Equation - Evolution of the PSO-PINN ensemble using the PSO-BP-CD method, showing the mean and two-sided 1-standard deviation of the ensemble.

both the PSO-BP and the PSO-BP-CD methods, the values $\alpha = 0.005$, $\beta = 0.99$, $c_1 = 0.08$, and $c_2 = 0.5$ were used. Table 3.1 displays the performance of PSO-PINN using the three PSO variants, which shows that the PSO-BP-CD method produced the best average $L - 2$ error over 10 independent repetitions of the experiment. The minimum L_2 error over the 10 repetitions was almost identical between the PSO-BP and PSO-BP-CD variants, with slight superiority to the first. Figure 3.3 displays the evolution of training of the PSO-PINN ensemble using the PSO-BP-CD method, in one of the 10 tests. We can observe that the PSO-PINN ensemble shows a reasonable solution after 500 iterations, and has converged at 2000 iterations. As expected, the variance of the ensemble is largest when the approximation is off and it shrinks to nearly zero upon convergence to the analytical solution.

3.5.2.2 Burgers Equation

.

Here we consider the following version of the viscous Burgers equation:

$$\begin{aligned} &\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \qquad x \in [-1, 1], \quad t \in [0, 1] \\ &u(0, x) = -\sin(\pi x)\,, \\ &u(t, -1) = u(t, 1) = 0\,, \end{aligned} \tag{3.20}$$

with kinematic viscosity $\nu = 0.02/\pi$.

Again, we used the same parametrization for the PSO-BP and the PSO-BP-CD. The values for α, β, c_1 and c_2 were set to 0.001, 0.9, 0.08 and 0.05, respectively. For the PSO algorithm we used 0.9, 0.8 and 0.5 for β, c_1 and c_2, respectively. The results can be see in Table 3.1. It can be seen that the proposed PSO-BP-CD variant is better, by more than an order of magnitude, than the other variants. Figure 3.4 illustrates the evolution of training for the PSO-BP-CD variant in one of the experiments. As in the case of advection, the PSO-PINN ensemble produces, for this much more complex PDE, a reasonable solution after 500 iterations, and has converged at 2,000 iterations. Note how the variance is larger when the approximation is farther from the reference solution.

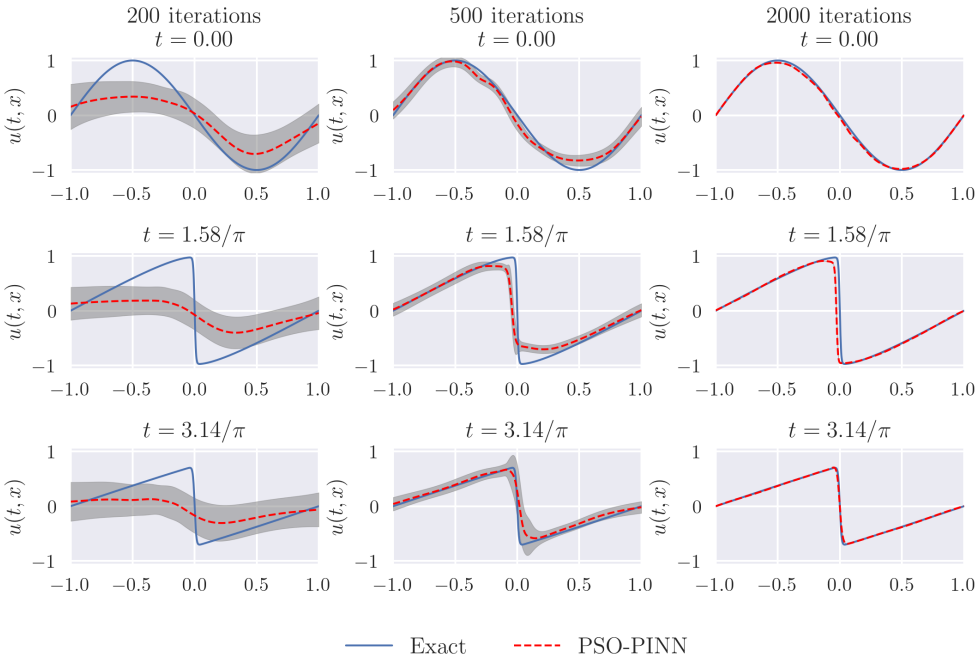

Figure 3.4: Burgers Equation - Evolution of the PSO-PINN ensemble using the PSO-BP-CD method, showing the mean and two-sided 1-standard deviation of the ensemble.

3.5.2.3 Heat Equation

Finally, we consider the classical 1D Heat equation. This PDE models temperature dissipation in a heat-conducting bar:

$$\begin{aligned} &\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}, \qquad x \in [0, L], \quad t \in [0, 1] \\ &u(0, t) = u(L, t) = 0, \\ &u(x, 0) = \sin\left(\frac{\pi x}{L}\right), \qquad 0 < x < L\,, \end{aligned} \tag{3.21}$$

where $L = 1$ is the length of the bar. The reference solution is $u(x, t) = e^{\frac{\pi^2 \alpha t}{L^2}} \sin(\frac{\pi x}{L})$.

For the PSO algorithm we used 0.9, 0.8 and 0.5 for β, c_1 and c_2, respectively. The same parametrization was used for the PSO-BP and the PSO-BP-CD, the values for α, β, c_1 and c_2 were set to 0.005, 0.99, 0.08 and 0.5, respectively. The results are displayed in Table 3.1, where again we

can see that the PSO-BP-CD variant performed the best. Figure 3.5 represents the training of the PSO-PINN for the heat equation. Figure 3.4 illustrates the evolution of training for the PSO-BP-CD variant in one of the experiments. Once again, the PSO-PINN ensemble produces a reasonable solution after 500 iterations, has converged at 2,000 iterations, and displays larger variance when the approximation is farther from the reference solution.

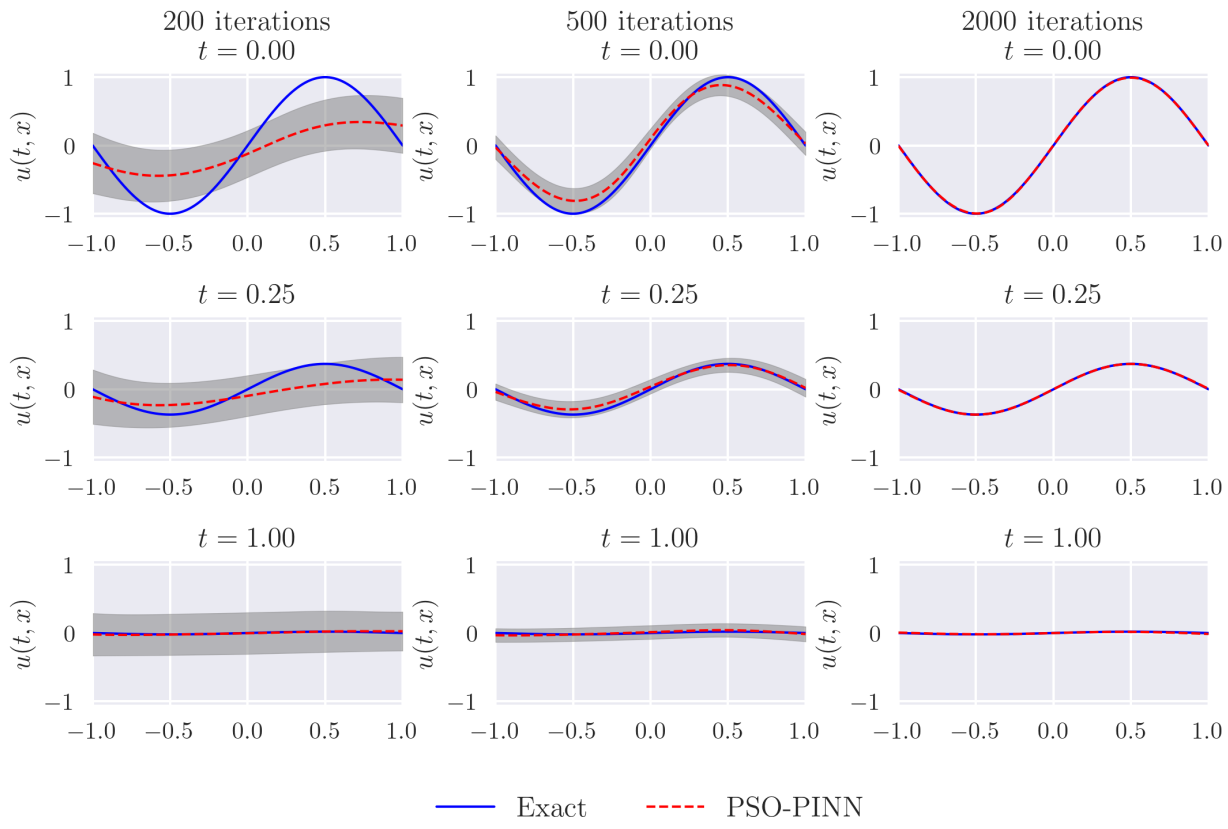

Figure 3.5: Heat Equation - Evolution of the PSO-PINN ensemble using the PSO-BP-CD method, showing the mean and two-sided 1-standard deviation of the ensemble.

3.5.3 Comparison with Traditional Ensembles of PINNs

In this section we show that PSO-PINN ensembles produce better results than traditional ensembles of PINNs, trained individually with ADAM gradient descent. For this, we selected two

Table 3.1: PSO-PINN results comparing the PSO variants, displaying the mean, standard deviation, and minimum (in parenthesis) of the L2 error of the PSO-PINN combined prediction across 10 independent repetitions of the experiment.

Problem	PSO variant	L_2 error	
Advection	PSO	0.1177 ± 0.0714	(0.0391)
	PSO-BP	0.0234 ± 0.0017	(**0.0212**)
	PSO-BP-CD	**0.0232** ± 0.0016	(0.0213)
Burgers	PSO	0.4197 ± 0.0711	(0.2819)
	PSO-BP	0.1470 ± 0.0826	(0.0579)
	PSO-BP-CD	**0.0125** ± 0.0502	(**0.0057**)
Heat	PSO	0.3974 ± 0.0834	(0.2617)
	PSO-BP	0.0068 ± 0.0020	(0.0027)
	PSO-BP-CD	**0.0051** ± 0.0026	(**0.0023**)

particularly hard examples, in order to fully appreciate the improvement over traditional ensembles afforded by the communication among particles in the PSO-PINN ensemble (the members of traditional ensembles do not communicate). In both benchmarks, the PSO-BP-CD variant is used with the PSO-PINN.

3.5.3.1 Forced Heat Equation

This benchmark features a 1D forced heat equation:

$$\begin{aligned} &\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 1 + x\cos(t) \quad x \in [-1,1], \quad t \in [0,1] \\ &\left.\frac{\partial u}{\partial x}\right|_{x=0} = \left.\frac{\partial u}{\partial x}\right|_{x=1} = \sin(t), \\ &u(x,0) = 1 + \cos(2\pi x)\,. \end{aligned} \tag{3.22}$$

This problem has an exact solution, given by

$$u(x,t) = 1 + t + e^{-4\pi^2 t}\cos(2\pi x) + x\sin(t)\,. \tag{3.23}$$

In this benchmark, we set set the number of initial, boundary, and residual points to 1024, 512, and 512, respectively. We used 20 neural networks of 5 hidden layers of 8 neurons in both

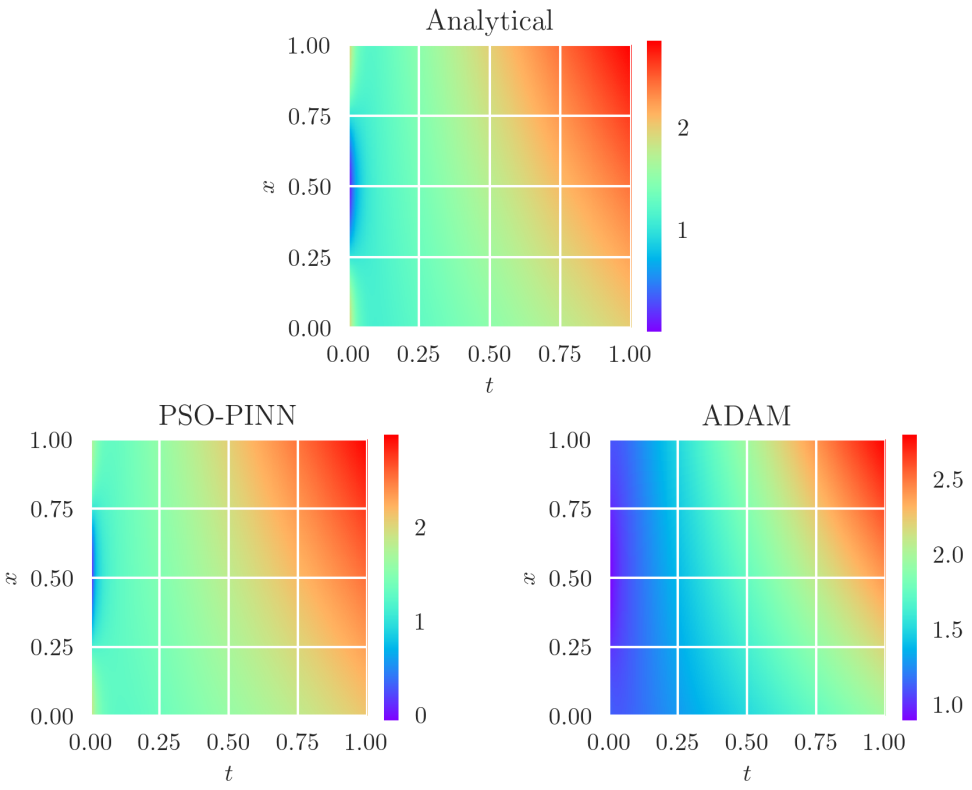

Figure 3.6: Forced Heat Equation - Results obtained by the PSO-PINN and conventional (ADAM) PINN ensembles at the end of training.

PSO-PINN and traditional ADAM ensembles, which are trained for 2,000 iterations. For the PSO-PINN training the values for α, β, c_1 and c_2 were set to 0.005, 0.99, 0.08 and 0.5, respectively. The ADAM parameters were set as follows: $\alpha = 0.001$, $\beta_1 = 0.99$, $\beta_2 = 0.999$.

Figure 3.6 displays the solution obtained by the PSO-PINN and ADAM ensembles at the end of training, where the approximation displayed is the mean of the ensemble, as before. It is clear that the PSO-PINN solution is close to the analytical solution, while the ADAM solution is rather poor. After running over 10 independent repetitions the L_2 error obtained by the PSO-PINN solution was 0.014 ± 0.006, while the error for the ADAM ensemble was 0.102 ± 0.005.

3.5.3.2 The Allen-Cahn Equation

The Allen-Cahn is a nonlinear reaction-diffusion PDE, which is used in phase-field models of the evolution of phase separation in a multi-component metal alloy. The Allen-Cahn equation considered here is:

$$\begin{aligned}
&\frac{\partial u}{\partial t} - D\frac{\partial^2 u}{\partial x^2} - 5(u - u^3) \;=\; 0, \quad x \in [-1, 1], \quad t \in [0, 1] \\
&u(-1, t) \;=\; u(1, t)\,, \\
&\left.\frac{\partial u}{\partial x}\right|_{x=-1} = \; -\left.\frac{\partial u}{\partial x}\right|_{x=1}\,, \\
&u(x, 0) \;=\; x^2 \cos(\pi x)\,,
\end{aligned} \tag{3.24}$$

where $D = 0.0001$ is the diffusivity coefficient. The Allen-Cahn PDE is a challenging benchmark for PINNs due to sharp space and time transitions in its solutions and the periodic boundary condition. In order to deal with the latter, the PINN boundary loss term has to be modified, as specified, e.g., in [51].

In this experiment, we set set the number of initial, boundary, and residual points to 256, 400, and 2000, respectively. In addition to these data, we use 2000 data points randomly sampled throughout the spacial-temporal domain using Latin hypercube method. We used 100 neural networks of 5 hidden layers of 8 neurons in both PSO-PINN and traditional ADAM ensembles, which are trained for 2,000 iterations. We used the same parametrization described in Section 3.5.3.1 for the PSO-PINN and the ADAM training.

Over 10 independent repetitions of the experiment, the PSO-PINN ensemble achieved and L_2 error of 0.093 ± 0.032, while the ADAM ensemble achieved 0.327 ± 0.029. The results at the end of training for one of the repeated experiments are displayed in Figure 3.7. We can observe that the PSO-PINN ensemble had converged to the reference solution, while the ADAM ensemble had not.

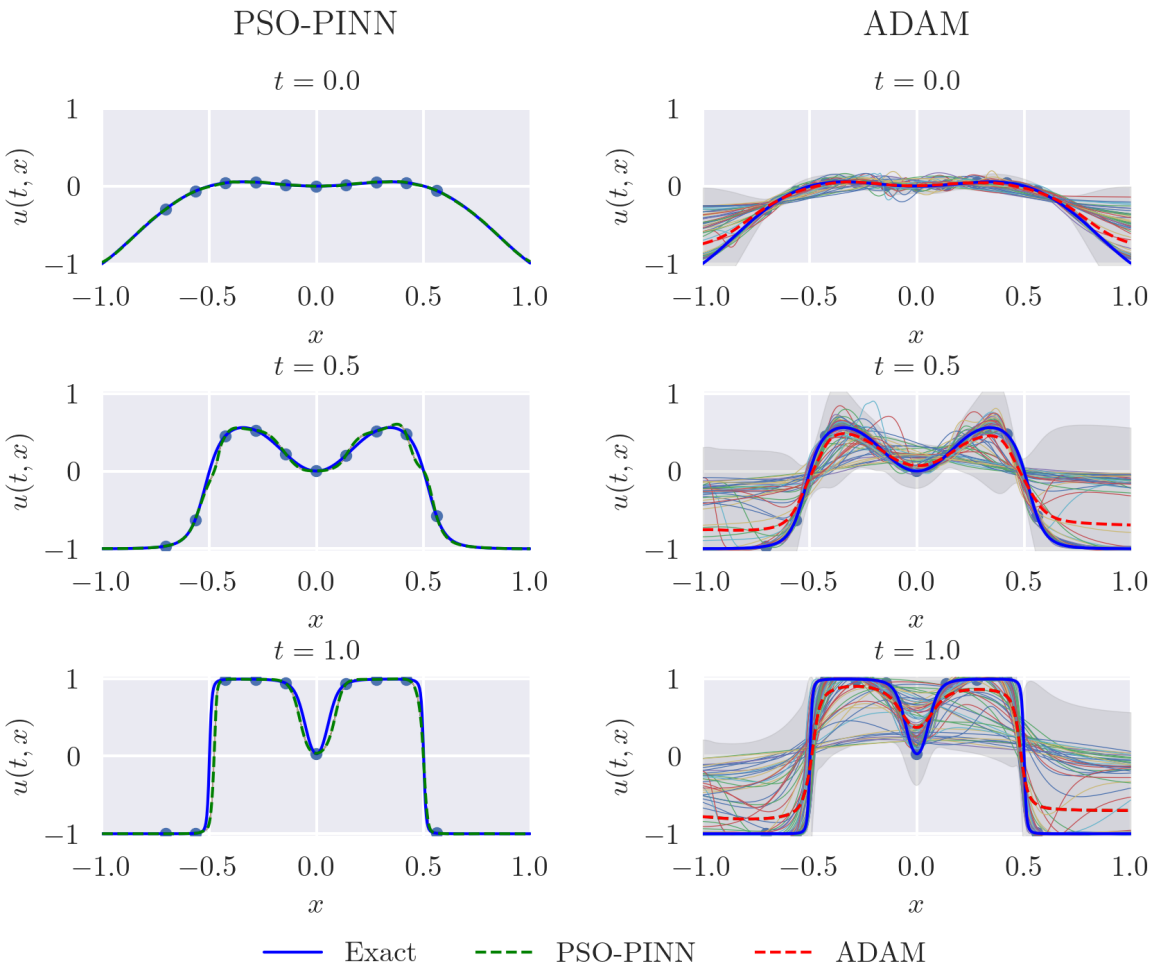

Figure 3.7: Allen-Cahn Equation - Results obtained by the PSO-PINN and conventional (ADAM) PINN ensembles at the end of training.

3.5.4 Discussion

The experimental results displayed promising results for PINN training using swarm intelligence and ensemble models. It was seen in Section 4.2 that both PSO-BP and PSO-BP-CD show superior results to using a basic PSO algorithm. This is no doubt due to the extra information contained in the gradient of the function. We also observed that PSO-BP-CD, the variant we proposed in this paper, performed better than PSO-BP, especially in the case of the nonlinear Burgers benchmark.

PSO-PINN provides the advantages of an ensemble approach, allowing the use of the mean of the ensemble as the predictor and its variance as a measure of the corresponding prediction uncertainty. However, PSO-PINN has clear advantages over a standard ensemble of PINNs trained with ADAM, as seen in the results of Section 4.3. The PSO-PINN ensembles are more precise and converge faster, with well-calibrated variance, leading to a greater confidence in the solution. This no doubt results from the fact that there is communication between the PINNs in a PSO-PINN swarm, while the traditional ensemble the PINNs do not communicate.

Setting the hyperparameters of the PSO-PINN ensemble, i.e., α, β, c_1, and c_2, is a model selection problem comparable to setting the correct learning rates and initialization in training a deep neural neural network. Indeed, the hyper parameter α is directly comparable to the learning rate used in traditional gradient descent. We observed in our experimental results that suitable hyperparameter values were fairly consistent across experiments. Also, there was no need to set a schedule for the learning rate hyperparameter α, as is usually needed in traditional gradient descent algorithms

3.6 Conclusion

In this paper, we proposed PSO-PINN, a particle-swarm optimation methodology for training physics-informed neural networks. PSO-PINN can be employed with any variant of PSO; here, we proposed PSO-BP-CD, a modification of the existing hybrid PSO-gradient descent PSO-BP method, which puts more weight on the gradient descent component as training progresses. PSO-BP-CD was observed in our experiments to be the best-performing variant of PSO for use with PSO-PINN.

PSO-PINN produces an ensemble of PINN solutions, which allows variance estimation for quantifying the uncertainty in the prediction. A more considerable degree of agreement among the PINNs leads to small variance and a higher degree of confidence in the predicted values. Our experimental result indicate a clear advantage of PSO-PINN over traditional ensembles of PINNs. Other ensemble techniques, such as stacking [79] and snapshot [80], could be readily introduced into the PSO-PINN framework to produce even better predictions and uncertainty quantification.

As an ensemble approach, PSO-PINN incurs a larger computational cost than training a single PINN. However, some of the cost could be mitigated by using parallelization. There are out-of-the-shelf implementations in TensorFlow to parallelize tensors calculations, and PSO-PINN is already taking advantage of that. Nevertheless, there is room for improvement, mainly regarding distributed training strategies. Such procedures would allow PSO-PINN to distribute its training across multiple GPUs, clusters, multiple machines, or even Cloud TPUs.

Some other aspects of swarm optimization are yet to be explored, such as multi-objective optimization, which would be quite suitable for PINNs, since its total loss function is nothing more than a scalarization of multiple losses (often minimized as the sum of the losses). Using distributing computing strategies will unleash further improvements in scalability and new approaches for optimization. Future work will explore these and other improvements in the PSO-PINN methodology in order to tackle more challenging problems, including inverse mapping for the determination of equation parameters and solving high-dimensional problems with unobserved boundary conditions.

4. EXTENDING THE PSO-PINN: THE MULTI-OBJECTIVE AND MULTI-MODAL APPROACHES

4.1 Abstract

The PSO-PINN is a swarm strategy for training PINNs, it aims to mitigate the gradient descent difficulties when dealing with PDEs with irregular solutions. Additionally, it takes advantage of the ensemble properties to yield robust predictions with quantified uncertainty. In this paper, we propose two distinct versions for this method, namely Multi-Objective PSO-PINN and Multi-Modal PSO-PINN. The first one acknowledges the PINN as a multi-objective problem and handles the PSO-PINN training as such. This approach unleashes a new paradigm to deal with PINNs, allowing the analysis of the problem and to find out if the model-driven and the data-driven portions of the PINN are in agreement. The latter promotes a desirable characteristic of the PSO-PINN, the diversity of the solutions. The Multi-Modal approach enforces multiple local optima during the training, guaranteeing a solution composed of a diverse ensemble.

4.2 Introduction

The Scientific Machine Learning [81] field has gotten tremendous traction in the last years, especially in the Physics-Informed Neural Networks (PINNs) [82, 46, 47, 83, 84]. Although the idea of Physics-informed Machine Learning has been firstly explored almost three decades ago [85], recent advancements in high processing computational methods [86], and particularly in the automatic differentiation algorithms [87, 88, 89], have attracted great attention to PINNs. The method have emerged as an alternative to the numerical methods solvers for Partial Differential Equations (PDE), introducing the great advantage of including collected data into the model. Thus, unlike typical data-driven techniques, PINNs take into account prior information of the physical understanding in the problem domain.

Nevertheless, adjusting the PINN mechanisms to find an optimal spot in the range between a PDE solver and a fully data-driven approach is not a straightforward task. To date, it is commonly

performed by adjusting the weights on the loss function. A work that can be made manually, based on the user's familiarity with the problem [90], or using adaptive weights [51]. Another reported weakness is how fragile the method might be in facing more complex problems [91, 15, 16], especially when solving "stiff" PDEs [92], with solutions that contain sharp space or transitions [15].

The PSO-PINN [93] is a novel training method for PINNs, proposed to mitigate the often reported fragility against "stiff" PDEs. Using a hybrid optimization method, combining gradient descent and meta-heuristics, this method has proven to successfully solve a range of PDEs, including "stiff" PDEs. Also, it takes advantage of ensemble techniques to generalize better when dealing with noisy data. Like many population-based heuristic optimizations, the Particle Swarm Optimization (PSO) [55] algorithm is expandable enough to contemplate different application scenarios. Indeed, there are a number of proposed variations on the algorithm to contemplate Multi-Objective problems [94, 95] and Multi-Modal optimization [96, 97].

In this paper, we explore the potential of Multi-Objective Optimization for solving PINNs, specifically adapting the PSO-PINN algorithm to deal with the PINN losses independently. We propose a new approach for choosing the extent of the influence of the analytical model and the available data in the neural network model. Instead of determining it before the training, this new approach allows the practitioner to simultaneity train the model over the many possible objectives and select the best suited for the problem after a post-training analysis.

We also attend to a particular deficiency of PSO-PINN. Although the algorithm was designed to use ensemble methods to improve the generalization properties of the solution, it is always liable to full convergence. In this case, even though there is still a population of results, the particles would be so close to each other that it may be seen as a unique solution. To avoid such a scenario, we propose to use Multi-Modal Optimization in order to forcibly find multiple local optima in the PINN loss surface. These local optima, in turn, may yield a better solution when presented to problems with missing physics or noisy data.

The remainder of this paper is organized in the following sections. Section 4.3 brings some related work and advancements on the field. Section 4.4, the preliminaries, describes the theoretical

framework where this work is based upon. Section 4.5 introduces the Multi-Objective PSO-PINN and explore some of its features through a couple of examples. The Section 4.6 introduces the Multi-Modal PSO-PINN and addresses its capabilities. Finally, the Section 4.7 concludes the paper discussing the advantages and limitations of the proposed methods and lists possible future works.

4.3 Related Work

Physics-Informed Neural Networks (PINNs) [46] have shown remarkable success in a variety of problems modeled by partial differential equations [47, 48]. This technique has the appealing property of integrating machine learning and physical laws. Therefore, the final model is not solely data-driven, but it uses partial differential equations as prior information. In the last few years PINNs are widely expanding into various real-world problems, such as fluid mechanics [84, 90, 98], material science [99, 100] and biomedical engineering [101, 102, 103, 104]. There is also an enormous variety of techniques and approaches for building, training and evaluating PINNs [53, 83, 54, 105, 93]. Fundamentally, the PINN framework consists in modeling a PDE into a Deep Learning model, using the PDE constraints as the losses guiding the network training (see more details in section 4.4.2). Usually, all these losses are aggregated together, and the resulting gradient is used to update the weights of the neural network incrementally.

Alternatively, one can deal with the training of multiple losses using **Multi-Objective Optimization** (MOO) [106, 107]. This approach yields a space of possible solutions in the approximated Pareto Front [108]. Dealing with PINNs as a MOO problem also allows new training strategies, where portions of the loss can be optimized in turns, similar to what is done in **Multi-Task Learning** (MTL) [109, 110].To some extent, there are works on PINNs [111, 112], but most of them are restricted to MTL.

There reported attempts to use MOO for Deep Learning [113, 114, 115, 116]. However, deep networks are a challenging domain to MOO, considering how computationally expensive it could be. The two most typical approaches are somewhat costly; they would either train multiple models for populating the Pareto Optimal Front or rely on large parameter dependencies of hyper-networks.

There is also another family of techniques for tackling multiple solutions, the **Multi-Modal**

Optimization(MMO) [117, 118, 119]. Instead of finding multiple objectives in the loss function, this approach picks multiple local and global optima.

Particle Swarm Optimization (PSO) [55, 120] is a well known algorithm for metaheuristic optimization with notably versions for dealing with MOO [121, 94, 95] and MMO [96, 97]. The PSO algorithm is widely used in numerous applications, including deep neural network training [56, 57, 58, 59, 60], for both training [122, 123, 124] and network hyper-parametrization [125, 126]. The PSO-PINN [93] is an implementation of the PSO algorithm particularly designed to reduce the pathological behavior of gradient-based optimization on PINNs. In addition, it is capable of using the ensemble properties of the swarms to promote a solution with good generalization and also quantify the uncertainty related to such a solution.

4.4 Preliminaries

In this section, we provide a basic survey of deep neural networks, PINNs, and optimization methods. The models and the definitions described in this section will contribute as a solid base when explaining the methods, experiments and results ahead.

4.4.1 Deep Neural Networks

Given enough hidden neurons and sufficient training, multi-layer feed forward networks are universal approximators[49]. This outstanding property let this family of algorithms evolve in what is called today's deep learning algorithms[70]. Usually, the topology consists in a feed-forward fully-connected network, the basic architecture for deep learning. A fully-connected neural network with L layers is a function $f_\theta : \mathbb{R}^d \to \mathbb{R}^k$ described in Equation 4.1. This function is composed essentially by three components, the weights, bias and activation functions. Among popular choices for the activation function are the sigmoid function, the hyperbolic tangent function (tanh), and the rectified linear unit (ReLU) [71].

$$f_\theta(x) \;=\; W^{[L-1]}\sigma \circ (\cdots \sigma \circ (W^{[0]}x + b^{[0]}) + \cdots) + b^{[L-1]} \tag{4.1}$$

where σ is an entry-wise activation function, $W^{[l]}$ and $b^{[l]}$ are respectively the weight matrices and the bias corresponding to each layer l, and θ is the set of weights and biases:

$$\theta = (W^{[0]}, \cdots, W^{[L-1]}, b^{[0]}, \cdots, b^{[L-1]}) . \tag{4.2}$$

This framework allows the approximation of $f_\theta(\cdot)$ to any arbitrary function $g : \mathbb{R}^d \to \mathbb{R}^k$. The method for building this approximation is called training and it is typically guided by a loss function $\mathcal{L} : \Theta \times \mathcal{T} \to \mathbb{R}$. Among others, a popular choice in deep learning is the Mean Square Error(MSE):

$$\mathcal{L}(\theta, x) \; = \; |f_\theta(x) - y|^2, \tag{4.3}$$

where $y \in \mathbb{R}^k$ is the target value. Several other loss functions have been proposed in the last decades, where a particular choice may drastically change the behavior of the neural network model [127].

Calculating $f(x)$ and $\mathcal{L}(\theta, x)$ is commonly called the forward propagation. It will provide enough information for the subsequently phase, the back propagation. During the back propagation, the network's weights and bias are updated based on the gradient of the error given by the loss function:

$$\theta^{t+1} = \theta^t - \alpha \frac{\partial \mathcal{L}(\theta^t, x)}{\partial \theta^t} \tag{4.4}$$

where θ^t denotes the learnable parameters of the neural network at iteration t in gradient descent and α is the learning rate. Again, there were numerous gradient-based optimization methods proposed in the last years [7, 8, 128].

4.4.2 Physics-Informed Neural Networks

Consider a non-linear partial differential equation (PDE) of the general form defined on a bounded spatio-temporal domain $\{\Omega \in \mathbb{R}^n, [0, T]\}$:

$$\begin{aligned} \mathcal{N}[u(\mathbf{x},t)] &= f(\mathbf{x},t),\ \mathbf{x}\in\Omega,\ t\in[0,T] \\ u(\mathbf{x},t) &= g(\mathbf{x},t),\ \mathbf{x}\in\partial\Omega,\ t\in[0,T] \\ u(\mathbf{x},0) &= h(\mathbf{x}),\ \mathbf{x}\in\Omega \end{aligned} \tag{4.5}$$

where $\Omega \subset \mathbb{R}^n$, $u : \Omega \to \mathbb{R}^k$, and $\mathcal{N}[\cdot]$ is a spatio-temporal differential operator. The boundary of the domain is restricted by $\partial\Omega$. And the source, boundary condition, and initial condition are defined by $f(\cdot)$, $g(\cdot)$, and $h(\cdot)$, respectively.

The PINN approach to solve a given PDE consists in training a deep neural network $u_\theta(\mathbf{x}, \mathbf{t})$ (with a set of learnable parameters θ) in order to approximate the solution $u(\mathbf{x}, \mathbf{t})$ of the PDE. This task can be accomplished by manufacturing the losses in a way to respect the PDE's definition. Therefore, we would have multiple losses depicting the restrictions of the given PDE. Bellow we have the governing equation, the boundary conditions, and the initial conditions into the training losses for the neural network:

$$\mathcal{L}_r(\theta, \mathbf{x}_r, \mathbf{t}_r) = \frac{1}{N_r}\sum_{n=1}^{N_r} |\mathcal{N}[u_\theta(x_r^n, t_r^n)] - f(x_r^n, t_r^n)|^2 \tag{4.6}$$

$$\mathcal{L}_b(\theta, \mathbf{x}_b, \mathbf{t}_r) = \frac{1}{N_b}\sum_{n=1}^{N_b} |u_\theta(x_b^n, t_b^n)] - g(x_b^n, t_b^n)|^2 \tag{4.7}$$

$$\mathcal{L}_0(\theta, \mathbf{x}) = \frac{1}{N_0}\sum_{n=1}^{N_0} |u_\theta(x_0^n, 0)] - h(x_0^n)|^2 \tag{4.8}$$

where $\mathcal{L}_r$ is the PINN residual loss on collocation training points $\{x_r^n, t_r^n\}_{n=1}^{N_r}$ sampled randomly in the domain $\{\Omega, [0, T]\}$, $\mathcal{L}_b$ is the boundary condition loss on boundary points $\{x_b^n, t_b^n\}_{n=1}^{N_b}$, and $\mathcal{L}_0$ is the initial condition loss on initial points $\{x_0^n, 0\}_{n=1}^{N_0}$. Also, if there is any data-points for the problem, it can be readily incorporated as another loss component:

$$\mathcal{L}_d(\theta, \mathbf{x}_d, \mathbf{t}_d) = \frac{1}{N_d}\sum_{n=1}^{N_d} |u_\theta(x_d^n, t_d^n)] - y^n)|^2 \tag{4.9}$$

where $\mathcal{L}_d$ in the data loss resulted from the dataset $\{[x_b^n, t_b^n], y^n\}_{n=1}^{N_b}$. The resultant loss function used for training the PINN is often defined as the simple sum of the individuals losses:

$$\mathcal{L}_\theta = \mathcal{L}_r(\theta, \mathbf{x}_r, \mathbf{t}_r) + \mathcal{L}_b(\theta, \mathbf{x}_b, \mathbf{t}_b) + \mathcal{L}_0(\theta, \mathbf{x}_0, \mathbf{t}_0) + \mathcal{L}_d(\theta, \mathbf{x}_d, \mathbf{t}_d) \tag{4.10}$$

Some authors, however, proposed diverse strategies for weighted sum approaches [129, 130]. Thus, forcing the PINN to satisfy its restrictions more closely. This approach is particularly suitable for PDEs describing time-irreversible processes, where the solution has to be approximated well early. In this case, a larger weight may be applied to initial condition loss ($\mathcal{L}_0(\theta)$), compelling the PINN to tightly observe this restriction.

$$\mathcal{L}(\mathbf{x}, \mathbf{t}) = \sum_i^U \omega_i \mathcal{L}_i(\theta, \mathbf{x}_i, \mathbf{t}_i) \tag{4.11}$$

Where U is the set of losses for a given problem, following the examples here we have $U = [r, b, 0, d]$. Since we are dealing with PINNs, we have $i \geq 2$ and $\mathcal{L}_i(\theta, \mathbf{x}, \mathbf{t}) : \Theta \times \mathcal{X} \times \mathcal{T} \to \mathbb{R}$.

Lastly, there are adaptive weighted methods [16, 131, 51], where the weights for the individual losses are updated during the training. These weighting methods are somehow related to multi-purpose deep neural network training [132, 133].

4.4.3 Multi-Objective Optimization

The main goal during the PINN's training is to degrade each one of the Losses (defined in the Equations 4.6 to 4.9) to a minimum possible value. Although the PINNs commonly use scalarization methods to aggregate all the losses into a single value, it is a clear example of Multi-Objective Optimization. Instead of using equations 4.10 and 4.11 to guide the optimization of the θ parameters of the PINN[134], one can jointly minimize the set of $\mathcal{L}$ loss functions, following the Equation 4.12, where each objective focuses on minimizing the expectation for all the respective loss functions:

$$\theta^* = argmin_\theta \mathbb{E}_{(\mathbf{x},\mathbf{t})}\{\mathcal{L}_1(\theta, \mathbf{x}_1, \mathbf{t}_1), \mathcal{L}_2(\theta, \mathbf{x}_2, \mathbf{t}_2), \dots, \mathcal{L}_n(\theta, \mathbf{x}_n, \mathbf{t}_n)\} \tag{4.12}$$

Unfortunately, finding the optimal value for θ^* is an arduous task, and often there is no single solution able to optimize all losses $\mathcal{L}(\cdot)$ simultaneously. However, there is a set of θ^* optimal solutions for each individual Loss. These solutions compose the Pareto Front. The following two definitions are used to build the Pareto Front (where n is the given number of loss functions):

Definition 1(Pareto dominance). A solution θ^* dominates another solution θ (denoted as $\theta^* \prec \theta$) if and only if:

- θ^* is not worse than θ in any objective, i.e.: $\mathcal{L}_i(\theta^*, \mathbf{x}, \mathbf{t}) \leq \mathcal{L}_i(\theta, \mathbf{x}, \mathbf{t})\ \forall i \in \{1, 2, 3, ..., n\}$.
- θ^* is better than θ in at least one objective, i.e.: $\exists i \in \{1, 2, 3, ..., n\} : \mathcal{L}_i(\theta^*, \mathbf{x}, \mathbf{t}) < \mathcal{L}_i(\theta, \mathbf{x}, \mathbf{t})$

Definition 2 (Pareto optimality). A solution θ^* is Pareto optimal if it is not dominated by any other solution. Therefore, the set of all Pareto optimal solutions is defined as $\mathcal{P} := \{\theta \in \Theta \mid \nexists \theta' \in \Theta : \theta' \prec \theta\}$. Meanwhile, the Pareto front $\mathcal{F}$ is the n-dimensional manifold of the objective values of all Pareto optimal solutions $\mathcal{F} := \mathcal{L}(\theta) \in \mathbb{L} \mid \theta \in \mathcal{P}$, where $\mathbb{L} = \{\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3 ..., \mathcal{L}_n\}$.

The goals in a MOO problem are often conflicting. Thus the Pareto front may work as a tool to select the best model according to the objectives of main interest. The set of Pareto non-dominated models is composed of solutions that cannot improve any objective without degrading at least one of the remaining objectives. That particular setup comes in handy when it is needed to compromise in order to prioritize between solutions.

However, that might not be the case on most of PINNs. Theoretically, the restrictions applied on the differential operator must respect and adhere to each other, resulting in a concave curve of solutions with respect to the objectives, which allows the user shape the Pareto front according

to the loss weights [135]. In such circumstances, the Pareto may be degradable to zero, thereby the total loss as defined in Equation 4.10 would completely satisfy the objective equation and its restrictions.

Nonetheless, this is an unlikely scenario for real-conditions experiments. On the one hand, it might be hard to precisely determine all the prior information required to mount the differential equation beforehand. Also, when dealing with datasets, one is always subject to noisy measurements and missing data. The loss weighting could be yet another issue, as the pre-defined weights are susceptible to investigator bias [136]. In such cases, a MOO approach may be more suitable than the traditional PINN, there is a further discussion on this topic in the Section 4.5.

4.4.4 Multi-Modal Optimization

The training of a Deep Learning model is a guided search throughout the domain Θ, as it minimizes $\mathcal{L}(\theta, \mathbf{x}), \theta \in \Theta$. Typically, there is a limit to this approximation when the solution is too closely related to a particular set of data, thus losing its generality. This behavior is commonly called overfitting, and it is a well-known weakness in many Machine Learning techniques.

Multi-Modal Optimization (MMO) algorithms can locate multiple global optima in a single run. Since it does not seek a single global optima θ to minimize $\mathcal{L}(\theta, \mathbf{x})$, the idea is to find a set of solution candidates $\theta^* \in \Theta$ that locally minimize the function result in isolated peak regions. When applied to Deep Neural Networks, MMO may be a convenient method to tackle the bias-variance tradeoff [137]. One can use ensemble techniques with high-variance models (such as Deep Neural Networks), to get a better generalization and thus avoiding the overfitting.

Definition 3 (Multi-Modal Minimization Problem). Let v be the local minima $\theta_1^*, ..., \theta_{\mathrm{v}}^*$ of the total loss function $\mathcal{L}$ in Θ. Ordering these optima in $f(\theta_1^*) \leq ... \leq f(\theta_l^*) \leq h \leq ... \leq f(\theta_{\mathrm{v}}^*)$, the multi-modal minimization problem would approximate the set $\bigcup_{i=1}^{l} \{\theta_i^*\}$.

Note the addition of the variable h as a threshold in order to avoid possible poor quality local optima. It gives the flexibility to model the problem statement from find all local optima ($h = \infty$) to find only the global optimum ($h = f(\theta_1^*)$, and thus reducing the problem to a single optimization problem). The choose of h is usually given by the characteristics of one particular problem [138].

Most authors classifies MMO algorithms into two categories, the one-stage MMO and the two-stage MMO. The two-stage algorithms are called this way due to a previous step done before the optimization, they split up the candidate models before hand, usually using clustering algorithms, and then perform sub-clusters searches in the many sub-groups. On the other hand, the one-stage MMO algorithms dynamically arrange the population according a particular metric (such as the euclidean distance [139] or the gravitational force [140]), avoiding any need for previous arrangement of the candidate solutions. Although usually one-step MMO algorithms are easier to understand and implement, they typically are not as efficient as two-step algorithms [141, 142, 143].

4.5 Multi-Objective PSO-PINN

By design, a PINN could behave as a purely PDE-solver as well as adopt fully data-driven comportment. This is an outstanding property from PINNs, capable of putting the method to a whole new level when compared with the traditional ways of modeling physical systems. It leads to a flexible framework where the user can determine their confidence degree in each piece of information on the problem. Furthermore, it allows them to expand the information insights hidden in the data through the physics model, a procedure often known as inverse problems [46, 144, 145].

Despite this, adjusting the PINN mechanisms to find an optimal spot in the range between completely model-driven and fully data-driven is not a straightforward task. Often the data losses and the physics-based losses have notorious disparity in their values, which may lead to a poor approximation to the solution. Commonly, this fine tune is done by adjusting the weights of the loss function. Indeed, this adjustment is essential for setting up a well-working PINN.

The balance between model-driven and data-driven behavior is usually defined before the PINN training. It is generally done by using a trial-and-failure approach to determine appropriate weights [90], or during the training through self-adaptive methods [16, 131, 51]. Whatever the choice, it may imply a problematic situation, as the user generally has precarious conditions to choose these weights. Missing knowledge of the underlying physical system may incur in a poorly defined PDE, and lacking information on the dataset may compromise the data-driven search. At this point, a good setting for the PINN is hardly made with confidence.

Here, we propose using MOO to change this paradigm. Instead of choosing between model or data driven approaches before/during the training, we propose to make this selection after training. We can take advantage of the Pareto front created by most of the MOO algorithms to analyse possible solutions according to their spot on it. Notice that a MOO approach may be employed by a number of optimizations strategies, but it is particularly suitable for the PSO-PINN algorithm. First, its innate decentralized behavior facilitates the search in a multiple objective dimension. Second, it already keep track of a population of solutions, a small change on the algorithm can also keep track on the positions belonging to the Pareto front.

A few modifications are required in order to adapt the PSO-PINN to a MOO approach. First we need to create a detached archive of best positions. This archive should hold the all optimal positions (according to the definitions on section 4.4.3) visited during the training. These positions are the Pareto front, and it will serve as a collection of g_{best}, the global optima. At each iteration, the algorithm randomly chooses the g_{best} among the positions in the archive, since all of them are theoretically equally suitable solutions.

The update of the p_{best} values must follow the rules defined in Section 4.4.3 as well. A particle should move to a new position only if the new position is as good as the old one for all objectives, and better in at least one of the objectives. That is the minimal condition to guarantee the movement is not hurting any of the objectives.

Another major modification is the removal of the gradient-based component. Since now we are dealing with multiples objectives, supposedly holding no distinction in importance, a gradient-based approach would be prohibitive. First, there would be a number of gradient vector, and the size of this vector would be as large as the number of objectives. It is well known how the increase of gradients-vector may harm the optimization process. Also, there would be a considerably computational cost related to compute multiple gradients after each iteration.

There is a caveat when adopting a multi-objective approach to solve PINNs, as solutions to differential equations are usually not unique. If that is the case and the residual loss stands alone in an individual objective dimension, the PINN's solution will probably lie down in the first feasible

solution, not adhering to the physical constraints previously established. Here, we propose to aggregate the residual and boundary losses to avoid such a situation. This way, we reduce the PDE freedom keeping the flexibility to set the multiple objectives according to each problem's needs. The next examples address this issue and make clear how this hack works.

4.5.1 The Poisson Equation

The Poisson Equation is a famous PDE broadly employed in many fields of theoretical physics. Although this problem is usually found in three-dimensional Cartesian coordinates, here we restricted it to a single dimension. This way it would better accomplish its objective as a toy example for the Multi-Objective PSO-PINN analysis. Therefore, for the extent of this example, we have limited it as an elliptic Ordinary Differential Equation defined by:

$$\frac{\partial^2 u}{\partial x^2} = g(x) \tag{4.13}$$

With $x \in [0, 1]$ and the following boundary conditions $u(0) = u(1) = 0$. We manufactured $g(x)$ so that $u(x) = sin(2\pi x)$. Also, we added a few data points, for better illustrate the capabilities of the Multi-Objective approach. There are used six evenly spaced data points over the specified interval $[0, 1]$.

The total loss for the PSO-PINN was split into a vector of two positions, the first one in charge to track the error related to the analytical information and the second one responsible for the data-driven portion of the training:

$$\begin{aligned} \mathcal{L}(\theta, \mathbf{x}) = [\mathcal{L}_r(\theta) + \mathcal{L}_0(\theta, \mathbf{x}_0), \\ \mathcal{L}_d(\theta, \mathbf{x}_d)] \end{aligned} \tag{4.14}$$

This example was intentionally designed to simulate a real case scenario, where the PINN user do not have full knowledge of the equation ruling the physical phenomena or may have noise on his measurements of the experiment. For the first scenario, let's say the user mistakenly considered

$u(x) = sin(2\pi x)/2$ instead of it's true form. For the second one, the noisy data scenario, a random noise uniformly distributed on the range $[0.0, 0.5)$ was inject to the data points. We had purposely introduced such gross errors to have a better contrast on the predictions, thus a easier understanding of the Multi-objective Optimization capabilities.

Both of the Poisson problems cases were solved using the same parametrization. The neural network architecture was composed by one input neuron, ten layers of three neurons and an output neuron. There were a hundred particles in the population and they were trained for ten thousands iterations. The PSO parameters β, c_1, and c_2 were set to 0.99, 0.008 and 0.05, respectively.

The Figure 4.1 illustrates the results for both experiments. On the left, the noisy physics example and on the right the noisy data one. The first row has both the Pareto fronts, with the extreme solutions marked in red. The second and third rows represent the solutions given by these extreme solutions. The second row shows the best solution for the analytical objective and the third row shows the best solution for the data-driven objective.

As expected, the solution with less data loss was better in a scenario with missing physics, and the solution with less analytical loss was better in the scenario with noisy data. But another interesting finding lies on the influence from the other objectives, the best solution for the analytical loss in the noisy physics example does not fit entirely to the modified equation ($u(x) = sin(2\pi x)/2$), it borrows some information from the correct fitted data solution. The same thing happens the other way around, where the best data solution for the set trained with noisy data has a good fit in some extent of the solution.

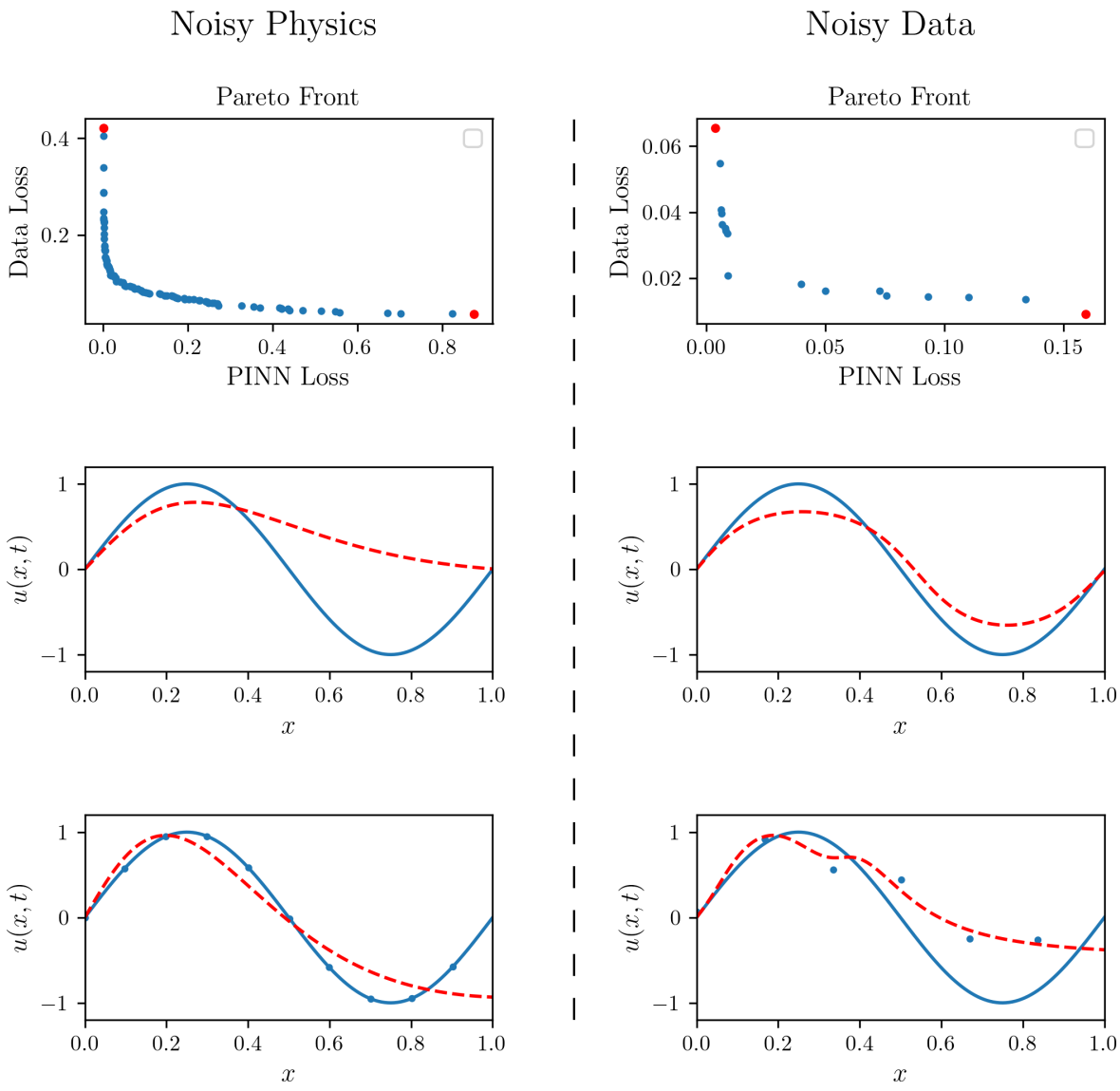

Figure 4.1: Multi-Objective PSO-PINN solutions for Poisson Equation – The results depicted are for two independent training. On the left column, a Multi-Objective PSO-PINN was trained with good data but noisy physics, i.e., a bad choice of coefficients for the PDE. On the right column, the Multi-Objective PSO-PINN was trained with a noisy dataset. On the top row, the Pareto fronts for each scenario. Each point on the Pareto front represents a neural network, and in the second and third row, we have the best solution to each dimension. This solution is also marked in red on the Pareto front.

4.5.2 The Heat Equation

Here, we use one of the most popular forms of the Diffusion equation, the heat equation. It is a famous PDE, with many known variants applied to a variety of fields, such science, applied mathematics and engineering. This particular case consider the cooling of a one-dimensional rod with an initial temperature distribution and Dirichlet boundary conditions on both rod ends. The systems dynamics, initial and boundary conditions are following the specifications found in [146]. The dissipation of the temperature in an iron rod, is given by the following equation:

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}, \qquad x \in [-1, 1], \quad t \in [0, 1] \tag{4.15}$$

The temperature of the rod $u(x, t)$ is a function of the position x over the time t, and α is the parametrization coefficient, commonly known as thermal diffusivity constant. The constraints for this equation, the initial (IC) and boundary (BC) conditions, are given by:

$$\begin{aligned} &u(0, t) = u(1, t) = 0, \\ &u(x, 0) = \sin\Big(\frac{n\pi x}{L}\Big), \qquad 0 < x < L \end{aligned} \tag{4.16}$$

where $L = 1$ is the length of the bar, $n = 1$ is the frequency of the sinusoidal initial conditions. As defined here, this problem is well-posed and has a unique solution:

$$u(x, t) = e^{\frac{-n^2\pi^2\alpha t}{L^2}} \sin(\frac{n\pi x}{L}) \tag{4.17}$$

As said before, the simple accommodation of the losses to the MOO framework (as defined in 4.12) would not suffice. Disconnected to any constraints, the equation 4.15 can assume infinite solutions, in fact, any constant would be a feasible solution.

Therefore, the multi-objective for this experiment are defined by the loss vector (following the

definitions from equations 4.6 to 4.9):

$$\begin{aligned}\mathcal{L}_\theta = \{&[\mathcal{L}_r(\theta, \mathbf{x_r}, \mathbf{t_r}) + \mathcal{L}_0(\theta, \mathbf{x_0})], \\ &[\mathcal{L}_b(\theta, \mathbf{x_b}, \mathbf{t_b}) + \mathcal{L}_d(\theta, \mathbf{x_d}, \mathbf{t_d})]\}\end{aligned} \tag{4.18}$$

Notice that we may use as many objectives as wanted for this problem, but in sake of simplicity we kept the objectives in a two dimensional vector. The first one containing the sum of the residual and original loss. The second is the sum of the boundaries and data losses.

For this experiment, we set the thermal diffusivity constant $\alpha = 0.4$ as the reference solution. Also, we added a uniformly distributed noise in the range $[0.0, 0.3)$ for all data in the problem, this includes original points, boundary points and data points. The neural-net architecture was composed by two input neurons (for x and t), followed by six layers hidden with eight neurons each, and a single output representing $u(x, t)$. The PSO parameters β, c_1, and c_2 were set to 0.99, 0.008 and 0.05, respectively.

In Figure 4.2 are displayed different solutions through the Pareto front. In the first row is the solution that optimize the first objective of $\mathcal{L}_\theta$, i.e. $argmin_\theta[\mathcal{L}_r(\theta, \mathbf{x_r}, \mathbf{t_r}) + \mathcal{L}_0(\theta, \mathbf{x_0})]$. The second row is the solution closest to the absolute zero. Since all the losses must be bigger than zero, we can achieve this using $\sum_i^{\mathcal{L}_\theta} \mathcal{L}_i$. The third and last row is the best optimization for the data loss, $argmin_\theta[\mathcal{L}_b(\theta, \mathbf{x_b}, \mathbf{t_b}) + \mathcal{L}_d(\theta, \mathbf{x_d}, \mathbf{t_d})]$.

The solution in the first row, the one minimal value found for the original and residual losses, had a good fit to the proposed PDE. It does not completely fit to the analytical solution, mostly due to the heavy noise in all data sources. Even though it is the optimal solution for PINN Loss, it suffers influence from the other losses in the Loss vector (4.18). After all, the algorithm has just one population. In the third row, the minimum data loss, although the solution had a good generalization for the data points, it was compromised by the noisy data points. Lastly, there is the second row, which depicts the solution with the minimum sum of all objective losses. This solution is deemed the elbow point for all the Pareto Front, as we can see in Figure 4.3. We couldn't say it is the best solution since, by definition, all solutions in the Pareto Front are equally good, but we might say the

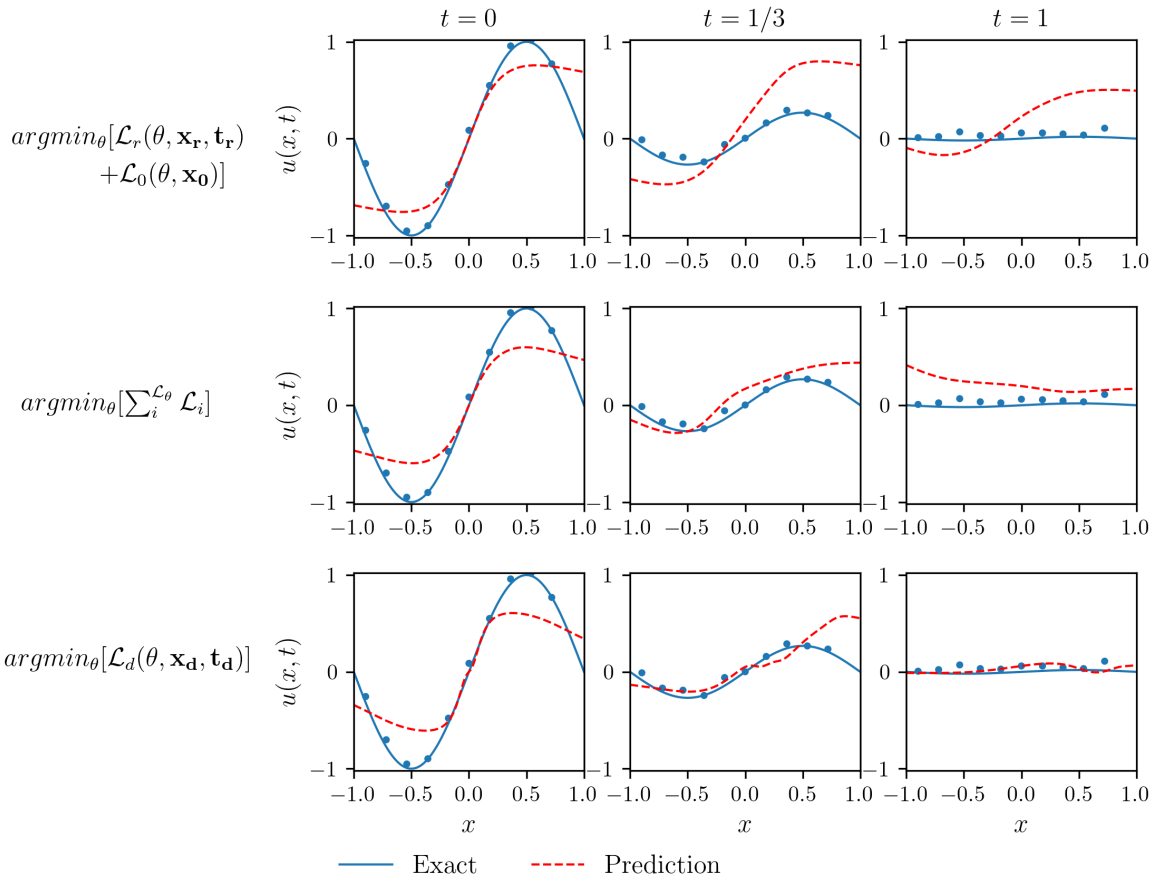

Figure 4.2: Multi-Objective PSO-PINN solutions for the Heat Equation – Each row represents one of the solutions found in the Pareto Front (Figure 4.3). On top, the solution that minimizes the first loss from the Loss vector. In the middle, we have the solution most close to zero, one that minimizes the sum of all in (4.18). In the last row is the best solution for the boundary and data losses.

one particular solution on the elbow point holds the best balance among the objectives.

Notice that in a real case scenario, we couldn't say with one would be the better solution for this problem. At this point, the only thing we have for fact is that the data points are in disagreement with the analytical solution. It could be either caused by sensors out of calibration or by bad coefficients on the PDE. A possible approach for this situation would be using the PINN to solve the inverse form of the problem [99, 145, 147].

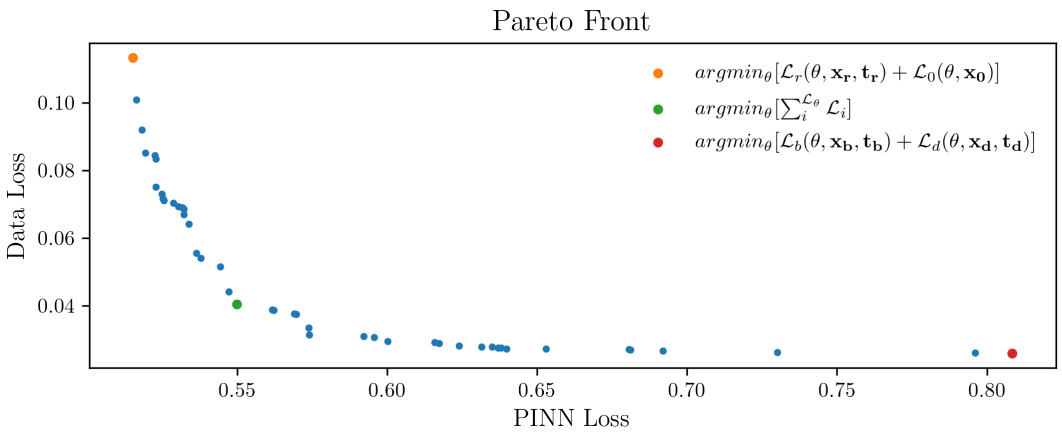

Figure 4.3: The Pareto front of the Multi-Objective PSO-PINN solutions for the Heat Equation.

4.6 Multi-Modal PSO-PINN

One of the biggest challenges in pattern recognition is to train a model avoiding overfitting it. In other words, there are two conflicting objectives while training a model, to create a well-fitted model for a given phenomenon while keeping it with enough generalization power to perform correctly in the face of new unseen data. There are many strategies to deal with it, but one of particular interest is the employment of ensembles. The idea behind it is to take advantage of many different models to keep a good level of generalization over the problem. This is one of the premises of the PSO-PINN, leveraging the pool of possible solutions in the population to avoid overfitting. This presumption may not be satisfied, though, when the swarm is fully converged. In this situation, even though the swarm is stationed in a presumably good local optimum, there is a high risk that this position is not well-suited for all samples, especially the unseen data.

The Multi-Modal approach aims to prevent this situation by forcibly splitting the population into a number of sub-swarms, consequently reaching different local optima after training. To reach this objective, a number of modifications were made to the PSO-PINN algorithm. First, for the multi-modal version, the original population is clustered using the k-means algorithm [148], making it a two-step multi-modal optimization. Also, during the training, we created checkpoints to remove

the sub-swarms converging to bad local optima, with no reposition. This is done using two criteria, (1) removing sub-swarms not decreasing the total loss over iterations and (2) removing sub-swarms outperformed when compared to the others. We intentionally reduce the iteration between the sub-swarms to a minimum, facilitating scalability and making it easier to create strategies for distributed computing.

For the experiments described in this section, we set k to a tenth of the population size, thus forcing the swarm to create 10 sub-swarms. This number may vary for different problems, network architectures, and population sizes. We also kept the number of removal checkpoints the same as k, totaling ten checkpoints for the experiments described in this section. It is valid to point out that removal is not mandatory at these checkpoints. The removal only happens if the sub-swarm is notably falling in the conditions mentioned before.

This section follows covering two examples. The first one, the Helmholtz equation, aims to illustrate the comportment, properties, and performance of the Multi-Modal PSO-PINN. In the second example, the Allen-Cahn equation, we intentionally introduced some noisy data points. This second experiment was designed to evaluate how well the Multi-Modal PSO-PINN can generalize over uncertainty.

4.6.1 The Helmholtz Equation

The Helmholtz Equation is a time-independent PDE largely used to describe the behavior of wave and diffusion processes (although it can easily be extended to physical models for both space and time). In the general form, the time invariant equation can be described as follows:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -k^2 u \tag{4.19}$$

In this particular example, we employ the Helmholtz equation to model evolution in a 2D spatial domain (x, y) whose result belongs to the closed-form analytical solution:

$$u(x, y) = \sin(a_1 \pi x) \sin(a_2 \pi y) \tag{4.20}$$

Resulting in a slightly change in the general form of the the PDE:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + k^2 u - f(x, y) = 0 \tag{4.21}$$

with the following Dirichlet boundary conditions:

$$\begin{aligned} u(-1, y) = u(1, y) = 0 \\ u(x, -1) = u(x, 1) = 0 \\ x \in [-1, 1], y \in [-1, 1] \end{aligned} \tag{4.22}$$

The $f(x, y)$ term is responsible for enforcing the PDE to the desired particular case, and it is described as follows:

$$\begin{aligned} f(x, y) = &-(a_1\pi)^2 \sin(a_1\pi x) \sin(a_2\pi y) \\ &-(a_2\pi)^2 \sin(a_1\pi x) \sin(a_2\pi y) \\ &+k^2 \sin(a_1\pi x) \sin(a_2\pi y) w \end{aligned} \tag{4.23}$$

For this example we adopted $a_1 = 1$, $a_2 = 4$, and $k = 1$, to allow the comparison to the Helmholtz PDE results reported in [15] and [51]. We have sampled 25 points over the boundaries, given a total of 100 boundary points. For the residual, we selected 1000 collocation points from the mesh of the two dimensions $x \in [-1, 1], y \in [-1, 1]$.

In order to better visualize the benefits and improvements of the multi-modal approach, we solved this problem using the PSO-PINN multi-modal training and the PSO-PINN algorithm as proposed in [93]. Both of the training observed exactly the same parametrization. The neural network architecture was composed of two input neurons representing the dimensions (x, y), followed by six hidden layers of 20 neurons each and an output neuron for the PDE solution. The models were trained for 6000 iterations with the following values for β, c_1, c_2, and α, respectively:

0.999, 0.0008, 0.005, and 0.0001.

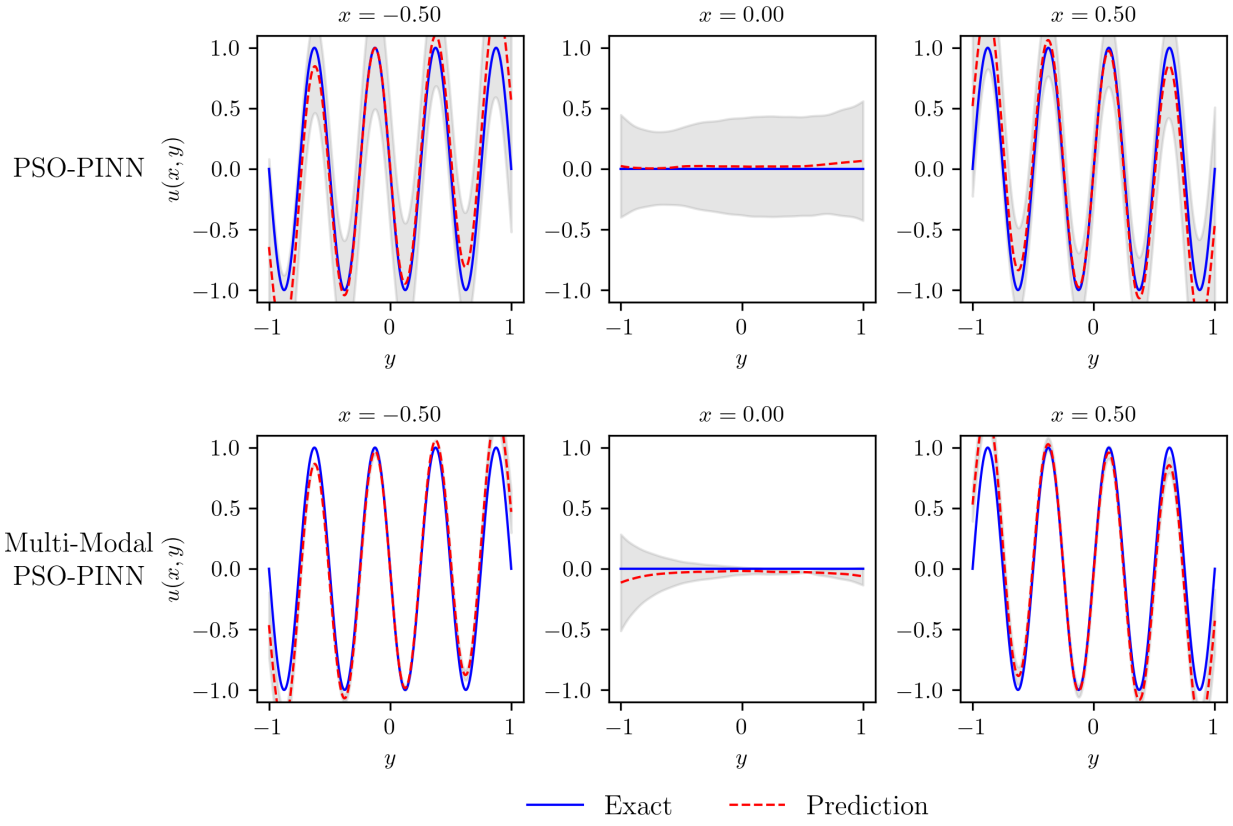

Figure 4.4: Multi-Modal PSO-PINN solutions for the Helmholtz Equation – On top, the PSO-PINN solution. Bellow, the Multi-Modal PSO-PINN solution.

We can see in Figure 4.4 that both solutions have a good fit to the Exact solution. Notice that one of the main properties of PSO-PINN is give not only the solution, but also the related uncertainty to the solution. Thus, there certainly are particles better fitted to the solution within the population, but we opted to use the average solution as result, with two standard deviations measuring the uncertainty related to each point in the solution domain space. Also, we lay emphasis on the exact solution being covered by the uncertainty boundaries, which reinforces the goals of PSO-PINN approach. Notwithstanding, these results are comparable to the solutions reported in [15] and [51], especially if we take into account that we used only 6k first order optimization iterations against 20k

(10k first order optimization plus 10k second order optimization) and 40k (first order optimization) from these works, respectively.

One may think of the solution depicted in Figure 4.4 as a contradictory result. After all, how the solution of the uni-modal PSO-PINN can have a larger deviation when compared with the multi-modal solution, assumedly composed of many different local minima. This happens because although they are converging in the same local minima, the particles of the uni-modal PSO-PINN are in sub-optimal positions in the same valley. The Figure 4.5 illustrates it well. It is a heatmap representing the euclidean distance between the particles in the population. Notice the scale changes between each snapshot (the initial positions, and the positions at the end of the training of the PSO-PINN and the multi-modal PSO-PINN), these values are the distance in the many-dimensional ($d = 2181$, the length of θ) and may vary given the network architecture. Also, the number of particles is reduced in the Multi-Modal PSO-PINN, since the remotion of sub-swarms is one of the steps of the algorithm.

There is a remarkable particle in the PSO-PINN solution, around the 50th position we can clearly see a darker line symbolizing other particles surrounding this one. That might be the g_{best}, the very best particle of the swarm, influencing the movement of all the other particles. At this point, the swarm is not yet fully converged, as advisable for avoiding overfitting (which would not be the case, of course, since this example is not dealing with sub-sampling or noisy data, but still a characteristic of the algorithm).

On the other hand, we have the at least four local optima covered in the Multi-Modal PSO-PINN solution. Even though these sub-swarms are well separated in the search domain, we can still see there is a good convergence on their solutions.

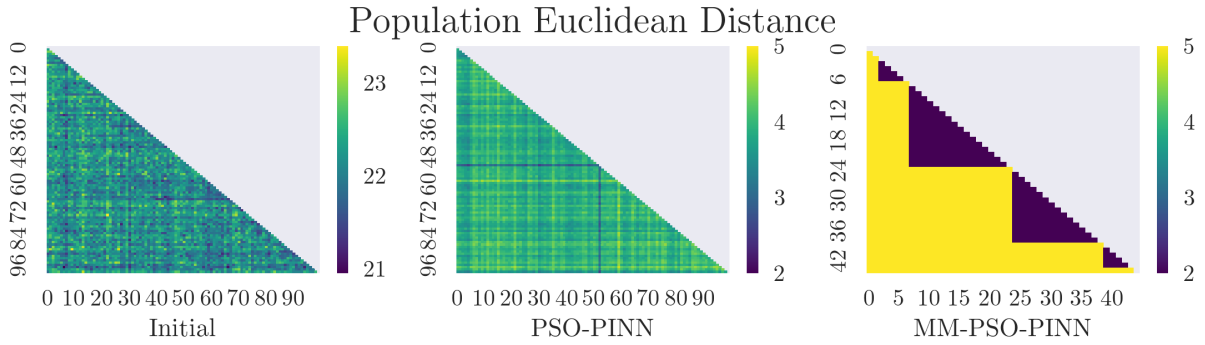

Figure 4.5: Population Euclidean Distance for Helmholtz Equation.

4.6.2 The Allen-Cahn Equation

The Allen-Cahn reaction-diffusion PDE is often used in material engineering, typically found in phase-field models to simulate the phase separation process in the microstructure evolution of metallic alloys [149, 150]. Here, we consider the Allen-Cahn equation as described in [51]:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + 5(u - u^3), \quad x \in [-1, 1], \quad t \in [0, 1] \tag{4.24}$$

with the following initial and boundary conditions:

$$\begin{aligned} u(x, 0) &= x^2 \cos(\pi x) \\ u(-1, t) &= u(1, t) \\ \frac{\partial u}{\partial x}(t, -1) &= \frac{\partial u}{\partial x}(t, 1) \end{aligned} \tag{4.25}$$

To allow a direct comparison to the result reported in [93], we kept $D = 0.0001$. We also follow the same parametrization, network architecture, and a number of iterations. The only change, though, is in the population size. We increased the population from 20 particles to 100 particles since the Multi-Modal approach requires a certain population size in order to split it into sub-swarms. To keep the comparison fair, we re-made the experiment, using 100 particles in the population.

The Table 4.1 lists all the L_2 errors, and variances for the solution found using an ADAM ensemble, the original PSO-PINN method, and the Multi-Modal PSO-PINN. Figure 4.6 help us to visualize how these results are manifested in the solution. The solid blue line represents the analytical solution for (4.24). Each row represents the solution (dashed red line) using an ensemble of conventional PINNs (running ADAM optimization), the PSO-PINN swarm, and the Multi-Modal PSO-PINN, respectively. The thin lines represent the individuals' solutions within each ensemble, while the grey shadow depicts two standard deviations along the predicted solutions.

Although there are several particles with good results in the ADAM ensemble, the high variance and a large number of non-fitted solutions made it an ill-suited method for using ensemble methods. The PSO-PINN had a better consensus among its solutions, but some high-variance zones in the prediction prevented the algorithm from yielding a high-fidelity solution. Despite the difference in the population size, these results are in agreement with the reports in [93]. The Multi-Modal PSO-PINN was able to efficiently solve this issue, decreasing the L_2 error and variance, which reflects a good fit to the Analytical solution.

Table 4.1: Allen-Cahn equation results. Errors and variance for each optimization strategy adopted in this experiment.

	L2 Error	Var
ADAM Ensemble	2.15e-01	1.96e-01
PSO-PINN	7.32e-02	1.46e-02
Multi-Modal PSO-PINN	5.80e-02	1.14e-02

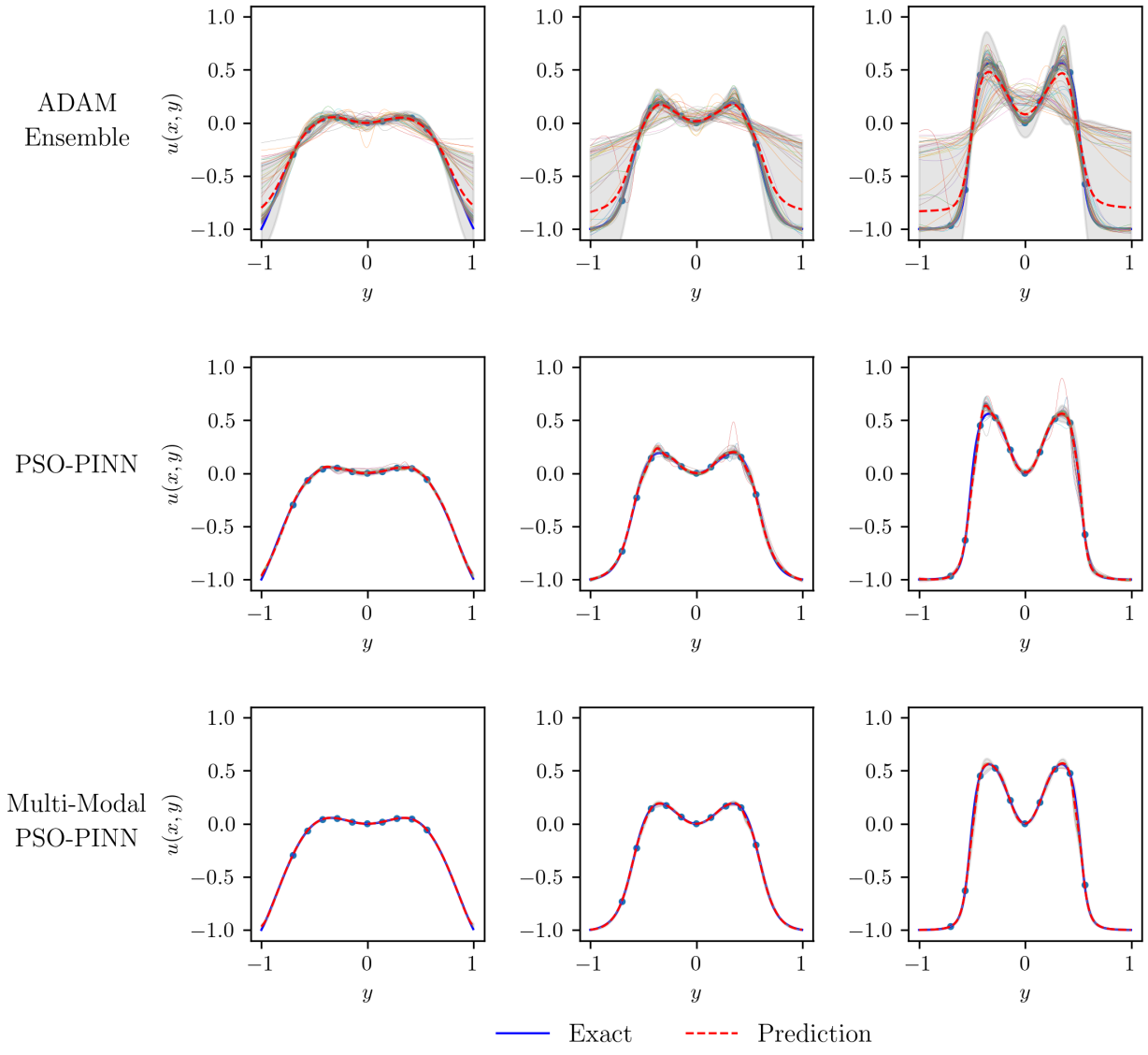

Figure 4.6: Results of the Allen-Cahn Equation – On top, the solution from an ensemble composed by conventional PINNs using ADAM optimization. Followed by the PSO-PINN solution and the Multi-Modal PSO-PINN solution.

4.7 Conclusion

In this paper, we introduced powerful training strategies for Machine Learning ensemble training. In particular, we extended the PSO-PINN algorithm, taking full advantage of the swarm properties inherited from the PSO. Here we presented the Multi-Objective and the Multi-Modal PSO-PINN to better explore the training for PINNs. These two approaches intend to tackle different potentials of the PINNs. The first one, the Multi-Objective PSO-PINN, intends to assist the training when it comes to setting the influence from the analytical model and training data. The Multi-Modal PSO-PINN aims to have better diversity after the training process. It enforces the PSO population to split itself in sub-swarms, yielding a solution composed of multiple local optima.

The Multi-Objective PSO-PINN can be used to appraise the problem and define how the prior information and the available data would best benefit the training. Much is said about the elasticity of PINNs, which are capable of bouncing between a strict mathematical PDE solver to a more yielding data-driven approach. Till now, this property was partially explored, most often relying on the weighting of the Total Loss function. The approach described here yields a set of Pareto non-dominated solutions, where the better one can be chosen based on how well-defined the terms are. For example, if there is known the dataset is considerably noisy, one may choose a model respecting more the physical constraints. On the other hand, if for some reason, the environment is not well defined on the PDE, or the PDE parameters are loosely fitted, one can choose a more data-driven solution.

The Multi-Modal PSO-PINN, on the other hand, improves the performance of the PSO-PINN algorithm. Not only it consistently finds solutions better fitted to the problems, but it also reduces the uncertainty levels related to the prediction. This characteristic is partially explained by the diversity of the swarm's population. During the training, this population is enforced to find many local optima resulting in robust solutions, especially when exposed to noisy data or missing physics.

5. CONCLUSION

The main focus of this dissertation was neural network training algorithms for Deep Learning. This topic has shown evident growth in the last decade, supporting several advancements in a diverse range of applications from science and engineering to business, law, and entertainment. During this time, we saw the rise of several new approaches for creating, training, and evaluating Deep Learning methods. Here, we contributed new methods to this environment, creating new architectures and improving the training process of deep neural networks. The major contributions are the gGAN and the PSO-PINN. The first is a novel method to apply GANs in bioinformatics, particularly designed to the genetic field. In the former, we propose a new algorithm for deep neural network optimization remarkably suited for PINNs (although robust enough to be applied in other domains).

The method proposed in Chapter 2, i.e., the gGAN architecture, is a new approach to deal with the limited genetically labeled data available. Creating genetic datasets is usually an expensive and time-demanding task requiring a good amount of human labor, as it involves patient recruitment, laboratory work, and bioinformatics expertise. It provides a means to create synthetic genetic data, a most valuable asset to studies that require several experimental phases consuming genetic data. Also, it modifies the usual perspective on GANs: in addition, to generate labeled genetic profiles, it also provides a self-aware classifier. While the labeled output of the discriminator predicts whether the individual with the corresponding genetic profile is likely to develop the disease, the unlabeled output predicts whether the genetic profile is real or synthetic. Given a real genetic profile, if the gGAN classifies it as not real (i.e., synthetic), we can infer that it most likely never had seen anything similar to that sample before, and thus it is not appropriate for prediction by the model.

The Particle-Swarm Optimization technique to train PINNs (PSO-PINN), described in Chapter 3, aims to mitigate a reported pathological behavior of gradient-based optimization when applied to PDEs with irregular solutions. PSO-PINN relies in an ensemble of PINN solutions, which allows robust predictions with quantified uncertainty (heavily grounded in the variance estimation). Comprehensive experimental results show that PSO-PINN, using a modified PSO algorithm with

a behavioral coefficient schedule, outperforms other PSO variants for training PINNs, as well as PINN ensembles trained with standard ADAM.

In Chapter 4, we propose the Multi-Objective and the Multi-Modal versions of the PSO-PINN. The Multi-Objective PSO-PINN changes the current paradigm of PINNs, allowing the users to train the PINN for multiple objectives in parallel and analyze the results after training. The Multi-Modal PSO-PINN aims to increase the accuracy and reduce the uncertainty related to the PINNs solution. This is done by promoting the diversity of solutions in the swarm population. Also, the Multi-Modal PSO-PINN algorithm was designed to satisfy requirements for distributed computing, which resulted in a highly parallelizable technique, allowing massive computation and distributed processing strategies. This can boost the search space of the algorithm, as new particles are added to the swarm. The distributed computing strategies would then allocate this population across multiple GPUs, clusters, and machines.

REFERENCES

[1] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[2] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

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

[5] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[6] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, *et al.*, "Evolving deep neural networks," in *Artificial intelligence in the age of neural networks and brain computing*, pp. 293–312, Elsevier, 2019.

[7] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization.," *Journal of machine learning research*, vol. 12, no. 7, 2011.

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[9] R. R. Halde, "Application of machine learning algorithms for betterment in education system," in *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*, pp. 1110–1114, IEEE, 2016.

[10] R. R. Robinson and C. Thomas, "Ranking of machine learning algorithms based on the performance in classifying ddos attacks," in *2015 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*, pp. 185–190, IEEE, 2015.

[11] D. Ravì, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G.-Z. Yang, "Deep learning for health informatics," *IEEE journal of biomedical and health informatics*, vol. 21, no. 1, pp. 4–21, 2016.

[12] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 3, pp. 653–664, 2016.

[13] F. Xing, Y. Xie, H. Su, F. Liu, and L. Yang, "Deep learning in microscopy image analysis: A survey," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 10, pp. 4550–4568, 2017.

[14] W. Min, E. Y. Ha, J. Rowe, B. Mott, and J. Lester, "Deep learning-based goal recognition in open-ended digital games," in *Tenth artificial intelligence and interactive digital entertainment conference*, 2014.

[15] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient flow pathologies in physics-informed neural networks," *SIAM Journal on Scientific Computing*, vol. 43, no. 5, pp. A3055–A3081, 2021.

[16] S. Wang, X. Yu, and P. Perdikaris, "When and why pinns fail to train: A neural tangent kernel perspective," *Journal of Computational Physics*, vol. 449, p. 110768, 2022.

[17] H. C. Assistance, "Summary of the hipaa privacy rule," *Office for Civil Rights*, 2003.

[18] S. Sheehan and Y. S. Song, "Deep learning for population genetic inference," *PLoS computational biology*, vol. 12, no. 3, p. e1004845, 2016.

[19] D. R. Schrider and A. D. Kern, "Supervised machine learning for population genetics: a new paradigm," *Trends in Genetics*, vol. 34, no. 4, pp. 301–312, 2018.

[20] L. Flagel, Y. Brandvain, and D. R. Schrider, "The unreasonable effectiveness of convolutional neural networks in population genetic inference," *Molecular biology and evolution*, vol. 36, no. 2, pp. 220–238, 2018.

[21] S. Bhatt, P. W. Gething, O. J. Brady, J. P. Messina, A. W. Farlow, C. L. Moyes, J. M. Drake, J. S. Brownstein, A. G. Hoen, O. Sankoh, *et al.*, "The global distribution and burden of dengue," *Nature*, vol. 496, no. 7446, p. 504, 2013.

[22] C. Xavier-Carvalho, C. C. Cardoso, F. de Souza Kehdy, A. G. Pacheco, and M. O. Moraes, "Host genetics and dengue fever," *Infection, Genetics and Evolution*, vol. 56, pp. 99–110, 2017.

[23] Y. M. Useche, B. N. Restrepo, D. M. Salgado, C. F. Narváez, O. Campo, and G. Bedoya, "Association of il4r-rs1805016 and il6r-rs8192284 polymorphisms with clinical dengue in children from colombian populations," *Journal of infection and public health*, vol. 12, no. 1, pp. 43–48, 2019.

[24] J. J. Just, "Genetic predisposition to hiv-1 infection and acquired immune deficiency virus syndrome: a review of the literature examining associations with hla," *Human immunology*, vol. 44, no. 3, pp. 156–169, 1995.

[25] A. J. Czaja, H. A. Carpenter, P. J. Santrach, and S. B. Moore, "Genetic predispositions for the immunological features of chronic active hepatitis," *Hepatology*, vol. 18, no. 4, pp. 816–822, 1993.

[26] C. Davi, A. Pastor, T. Oliveira, F. B. de Lima Neto, U. Braga-Neto, A. W. Bigham, M. Bamshad, E. T. Marques, and B. Acioli-Santos, "Severe dengue prognosis using human genome data and machine learning," *IEEE Transactions on Biomedical Engineering*, vol. 66, no. 10, pp. 2861–2868, 2019.

[27] T. A. Manolio, "Cohort studies and the genetics of complex disease," *Nature genetics*, vol. 41, no. 1, p. 5, 2009.

[28] E. Keogh and A. Mueen, "Curse of dimensionality," *Encyclopedia of machine learning*, pp. 257–258, 2010.

[29] W. H. Organization, S. P. for Research, T. in Tropical Diseases, W. H. O. D. of Control of Neglected Tropical Diseases, W. H. O. Epidemic, and P. Alert, *Dengue: guidelines for diagnosis, treatment, prevention and control*. World Health Organization, 2009.

[30] E. J. Nascimento, U. Braga-Neto, C. E. Calzavara-Silva, A. L. Gomes, F. G. Abath, C. A. Brito, M. T. Cordeiro, A. M. Silva, C. Magalhães, R. Andrade, *et al.*, "Gene expression profiling during early acute febrile stage of dengue infection can predict the disease outcome," *PloS one*, vol. 4, no. 11, p. e7892, 2009.

[31] A. R. Brasier, H. Ju, J. Garcia, H. M. Spratt, S. S. Victor, B. M. Forshey, E. S. Halsey, G. Comach, G. Sierra, P. J. Blair, *et al.*, "A three-component biomarker panel for prediction of dengue hemorrhagic fever," *The American journal of tropical medicine and hygiene*, vol. 86, no. 2, pp. 341–348, 2012.

[32] A. R. Brasier, Y. Zhao, J. E. Wiktorowicz, H. M. Spratt, E. J. Nascimento, M. T. Cordeiro, K. V. Soman, H. Ju, A. Recinos III, S. Stafford, *et al.*, "Molecular classification of outcomes from dengue virus-3 infections," *Journal of Clinical Virology*, vol. 64, pp. 97–106, 2015.

[33] S. Chatterjee, N. Dey, F. Shi, A. S. Ashour, S. J. Fong, and S. Sen, "Clinical application of modified bag-of-features coupled with hybrid neural-based classifier in dengue fever classification using gene expression data," *Medical & biological engineering & computing*, vol. 56, no. 4, pp. 709–720, 2018.

[34] B. P. Azevedo, P. C. S. Farias, A. F. Pastor, C. C. M. Davi, H. V. P. d. C. Neco, R. E. de Lima, and B. Acioli-Santos, "Aa ido1 variant genotype (g2431a, rs3739319) is associated with severe dengue risk development in a den-3 brazilian cohort," *Viral immunology*, 2019.

[35] A. L. V. Gomes, L. J. Wee, A. M. Khan, L. H. Gil, E. T. Marques Jr, C. E. Calzavara-Silva, and T. W. Tan, "Classification of dengue fever patients based on gene expression data using support vector machines," *PloS one*, vol. 5, no. 6, p. e11267, 2010.

[36] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[37] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional gans," *arXiv preprint arXiv:1706.02633*, 2017.

[38] B. Yelmen, A. Decelle, L. Ongaro, D. Marnetto, C. Tallec, F. Montinaro, C. Furtlehner, L. Pagani, and F. Jay, "Creating artificial human genomes using generative models," *bioRxiv*, p. 769091, 2019.

[39] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in neural information processing systems*, pp. 2234–2242, 2016.

[40] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[41] M. T. Cordeiro, A. M. Silva, C. A. Brito, E. J. Nascimento, M. C. F. Magalhães, G. F. Guimarães, N. Lucena-Silva, E. M. F. de Carvalho, and E. T. Marques Jr, "Characterization of a dengue patient cohort in recife, brazil," *The American journal of tropical medicine and hygiene*, vol. 77, no. 6, pp. 1128–1134, 2007.

[42] . G. P. Consortium *et al.*, "A global reference for human genetic variation," *Nature*, vol. 526, no. 7571, p. 68, 2015.

[43] S. Richards, N. Aziz, S. Bale, D. Bick, S. Das, J. Gastier-Foster, W. W. Grody, M. Hegde, E. Lyon, E. Spector, *et al.*, "Standards and guidelines for the interpretation of sequence variants: a joint consensus recommendation of the american college of medical genetics and genomics and the association for molecular pathology," *Genetics in medicine*, vol. 17, no. 5, p. 405, 2015.

[44] M. S. Naslavsky, G. L. Yamamoto, T. F. de Almeida, S. A. Ezquina, D. Y. Sunaga, N. Pho, D. Bozoklian, T. O. M. Sandberg, L. A. Brito, M. Lazar, *et al.*, "Exomic variants of an elderly

cohort of brazilians in the abraom database," *Human mutation*, vol. 38, no. 7, pp. 751–763, 2017.

[45] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *Advances in neural information processing systems*, pp. 8778–8788, 2018.

[46] 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*, vol. 378, pp. 686–707, 2019.

[47] M. Raissi, A. Yazdani, and G. E. Karniadakis, "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations," *Science*, vol. 367, no. 6481, pp. 1026–1030, 2020.

[48] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021.

[49] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[50] S.-i. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, vol. 5, no. 4-5, pp. 185–196, 1993.

[51] L. McClenny and U. Braga-Neto, "Self-adaptive physics-informed neural networks using a soft attention mechanism," *arXiv preprint arXiv:2009.04544*, 2020.

[52] R. van der Meer, C. W. Oosterlee, and A. Borovykh, "Optimally weighted loss functions for solving pdes with neural networks," *Journal of Computational and Applied Mathematics*, p. 113887, 2021.

[53] A. D. Jagtap and G. E. Karniadakis, "Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations," *Communications in Computational Physics*, vol. 28, no. 5, pp. 2002–2041, 2020.

[54] K. Shukla, A. D. Jagtap, and G. E. Karniadakis, "Parallel physics-informed neural networks via domain decomposition," *arXiv preprint arXiv:2104.10013*, 2021.

[55] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4, pp. 1942–1948, IEEE, 1995.

[56] A. Chakraborty and A. K. Kar, "Swarm intelligence: A review of algorithms," *Nature-Inspired Computing and Optimization*, pp. 475–494, 2017.

[57] F. Van den Bergh and A. P. Engelbrecht, "Cooperative learning in neural networks using particle swarm optimizers," *South African Computer Journal*, vol. 2000, no. 26, pp. 84–90, 2000.

[58] G. Das, P. K. Pattnaik, and S. K. Padhy, "Artificial neural network trained by particle swarm optimization for non-linear channel equalization," *Expert Systems with Applications*, vol. 41, no. 7, pp. 3491–3496, 2014.

[59] S. Mirjalili, "How effective is the grey wolf optimizer in training multi-layer perceptrons," *Applied Intelligence*, vol. 43, no. 1, pp. 150–161, 2015.

[60] S. J. Mousavirad, G. Schaefer, S. M. J. Jalali, and I. Korovin, "A benchmark of recent population-based metaheuristic algorithms for multi-layer neural network training," in *Proceedings of the 2020 genetic and evolutionary computation conference companion*, pp. 1402–1408, 2020.

[61] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[62] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[63] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *arXiv preprint arXiv:1612.01474*, 2016.

[64] L. Deng and J. Platt, "Ensemble deep learning for speech recognition," in *Proc. interspeech*, 2014.

[65] X. Qiu, L. Zhang, Y. Ren, P. N. Suganthan, and G. Amaratunga, "Ensemble deep learning for regression and time series forecasting," in *2014 IEEE symposium on computational intelligence in ensemble learning (CIEL)*, pp. 1–6, IEEE, 2014.

[66] Y. Cao, T. A. Geddes, J. Y. H. Yang, and P. Yang, "Ensemble deep learning in bioinformatics," *Nature Machine Intelligence*, vol. 2, no. 9, pp. 500–508, 2020.

[67] M. Ganaie, M. Hu, *et al.*, "Ensemble deep learning: A review," *arXiv preprint arXiv:2104.02395*, 2021.

[68] K. Haitsiukevich and A. Ilin, "Improved training of physics-informed neural networks with model ensembles," *arXiv preprint arXiv:2204.05108*, 2022.

[69] R. K. Yadav *et al.*, "Ga and pso hybrid algorithm for ann training with application in medical diagnosis," in *2019 Third International Conference on Intelligent Computing in Data Sciences (ICDS)*, pp. 1–5, IEEE, 2019.

[70] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[71] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, JMLR Workshop and Conference Proceedings, 2011.

[72] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," *Journal of machine learning research*, vol. 18, 2018.

[73] R. Eberhart and J. Kennedy, "Particle swarm optimization," in *Proceedings of the IEEE international conference on neural networks*, vol. 4, pp. 1942–1948, Citeseer, 1995.

[74] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, vol. 3, pp. 1945–1950, IEEE, 1999.

[75] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," *Soft Computing*, vol. 22, no. 2, pp. 387–408, 2018.

[76] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, JMLR Workshop and Conference Proceedings, 2010.

[77] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[78] R. J. LeVeque *et al.*, *Finite volume methods for hyperbolic problems*, vol. 31. Cambridge university press, 2002.

[79] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[80] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get m for free," *arXiv preprint arXiv:1704.00109*, 2017.

[81] N. Baker, F. Alexander, T. Bremer, A. Hagberg, Y. Kevrekidis, H. Najm, M. Parashar, A. Patra, J. Sethian, S. Wild, *et al.*, "Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence," tech. rep., USDOE Office of Science (SC), Washington, DC (United States), 2019.

[82] M. Raissi, "Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations," *arXiv preprint arXiv:1804.07010*, 2018.

[83] L. Yang, X. Meng, and G. E. Karniadakis, "B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data," *Journal of Computational Physics*, vol. 425, p. 109913, 2021.

[84] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks (pinns) for fluid mechanics: A review," *Acta Mechanica Sinica*, pp. 1–12, 2022.

[85] M. Dissanayake and N. Phan-Thien, "Neural-network-based approximations for solving partial differential equations," *communications in Numerical Methods in Engineering*, vol. 10, no. 3, pp. 195–201, 1994.

[86] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[87] J. Revels, M. Lubin, and T. Papamarkou, "Forward-mode automatic differentiation in julia," *arXiv preprint arXiv:1607.07892*, 2016.

[88] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[89] B. Van Merriënboer, O. Breuleux, A. Bergeron, and P. Lamblin, "Automatic differentiation in ml: Where we are and where we should be going," *Advances in neural information processing systems*, vol. 31, 2018.

[90] X. Jin, S. Cai, H. Li, and G. E. Karniadakis, "Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations," *Journal of Computational Physics*, vol. 426, p. 109951, 2021.

[91] S. Liao, T. Xue, J. Jeong, S. Webster, K. Ehmann, and J. Cao, "Hybrid full-field thermal characterization of additive manufacturing processes using physics-informed neural networks with data," *arXiv preprint arXiv:2206.07756*, 2022.

[92] R. L. Burden, J. D. Faires, and A. M. Burden, *Numerical analysis*. Cengage learning, 2015.

[93] C. Davi and U. Braga-Neto, "Pso-pinn: Physics-informed neural networks trained with particle swarm optimization," *arXiv preprint arXiv:2202.01943*, 2022.

[94] Q. Lin, J. Li, Z. Du, J. Chen, and Z. Ming, "A novel multi-objective particle swarm optimization with multiple search strategies," *European Journal of Operational Research*, vol. 247, no. 3, pp. 732–744, 2015.

[95] P. Dhal and C. Azad, "A deep learning and multi-objective pso with gwo based feature selection approach for text classification," in *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, pp. 2140–2144, IEEE, 2022.

[96] Y. Li, Y. Chen, J. Zhong, and Z. Huang, "Niching particle swarm optimization with equilibrium factor for multi-modal optimization," *Information Sciences*, vol. 494, pp. 233–246, 2019.

[97] S. Dennis and A. Engelbrecht, "A review and empirical analysis of particle swarm optimization algorithms for dynamic multi-modal optimization," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, IEEE, 2020.

[98] L. Sun, H. Gao, S. Pan, and J.-X. Wang, "Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data," *Computer Methods in Applied Mechanics and Engineering*, vol. 361, p. 112732, 2020.

[99] E. Haghighat, M. Raissi, A. Moure, H. Gomez, and R. Juanes, "A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics," *Computer Methods in Applied Mechanics and Engineering*, vol. 379, p. 113741, 2021.

[100] E. Zhang, M. Dao, G. E. Karniadakis, and S. Suresh, "Analyses of internal structures and defects in materials using physics-informed neural networks," *Science advances*, vol. 8, no. 7, p. eabk0644, 2022.

[101] F. Sahli Costabal, Y. Yang, P. Perdikaris, D. E. Hurtado, and E. Kuhl, "Physics-informed neural networks for cardiac activation mapping," *Frontiers in Physics*, vol. 8, p. 42, 2020.

[102] G. Kissas, Y. Yang, E. Hwuang, W. R. Witschey, J. A. Detre, and P. Perdikaris, "Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4d flow mri data using physics-informed neural networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 358, p. 112623, 2020.

[103] M. Liu, L. Liang, and W. Sun, "A generic physics-informed neural network-based constitutive model for soft biological tissues," *Computer methods in applied mechanics and engineering*, vol. 372, p. 113402, 2020.

[104] J. H. Lagergren, J. T. Nardini, R. E. Baker, M. J. Simpson, and K. B. Flores, "Biologically-informed neural networks guide mechanistic modeling from sparse experimental data," *PLoS computational biology*, vol. 16, no. 12, p. e1008462, 2020.

[105] E. J. R. Coutinho, M. Dall'Aqua, L. McClenny, M. Zhong, U. Braga-Neto, and E. Gildin, "Physics-informed neural networks with adaptive localized artificial viscosity," *arXiv preprint arXiv:2203.08802*, 2022.

[106] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and multidisciplinary optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[107] M. Emmerich and A. H. Deutz, "A tutorial on multiobjective optimization: fundamentals and evolutionary methods," *Natural computing*, vol. 17, no. 3, pp. 585–609, 2018.

[108] Y. Tian, L. Si, X. Zhang, R. Cheng, C. He, K. C. Tan, and Y. Jin, "Evolutionary large-scale multi-objective optimization: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 8, pp. 1–34, 2021.

[109] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," *Advances in neural information processing systems*, vol. 31, 2018.

[110] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[111] B. Bahmani and W. Sun, "Training multi-objective/multi-task collocation physics-informed neural network with student/teachers transfer learnings," *arXiv preprint arXiv:2107.11496*, 2021.

[112] P. Thanasutives, M. Numao, and K.-i. Fukui, "Adversarial multi-task learning enhanced physics-informed neural networks for solving partial differential equations," in *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9, IEEE, 2021.

[113] J.-A. Désidéri, "Multiple-gradient descent algorithm (mgda) for multiobjective optimization," *Comptes Rendus Mathematique*, vol. 350, no. 5-6, pp. 313–318, 2012.

[114] A. Dosovitskiy and J. Djolonga, "You only train once: Loss-conditional training of deep networks," in *International conference on learning representations*, 2019.

[115] A. Navon, A. Shamsian, G. Chechik, and E. Fetaya, "Learning the pareto front with hypernetworks," *arXiv preprint arXiv:2010.04104*, 2020.

[116] X. Lin, Z. Yang, Q. Zhang, and S. Kwong, "Controllable pareto multi-task learning," *arXiv preprint arXiv:2010.06313*, 2020.

[117] D. Beasley, D. R. Bull, and R. R. Martin, "A sequential niche technique for multimodal function optimization," *Evolutionary computation*, vol. 1, no. 2, pp. 101–125, 1993.

[118] E. Cuevas and M. González, "An optimization algorithm for multimodal functions inspired by collective animal behavior," *Soft Computing*, vol. 17, no. 3, pp. 489–502, 2013.

[119] Z. Wei, W. Gao, G. Li, and Q. Zhang, "A penalty-based differential evolution for multimodal optimization," *IEEE Transactions on Cybernetics*, 2021.

[120] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, "Particle swarm optimization: A comprehensive survey," *IEEE Access*, 2022.

[121] S. Mostaghim and J. Teich, "Strategies for finding good local guides in multi-objective particle swarm optimization (mopso)," in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706)*, pp. 26–33, IEEE, 2003.

[122] R. K. Yadav *et al.*, "Pso-ga based hybrid with adam optimization for ann training with application in medical diagnosis," *Cognitive Systems Research*, vol. 64, pp. 191–199, 2020.

[123] R. Mohapatra, S. Saha, C. A. C. Coello, A. Bhattacharya, S. S. Dhavala, and S. Saha, "Adaswarm: Augmenting gradient-based optimizers in deep learning with swarm intelligence," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 2, pp. 329–340, 2021.

[124] H. Zhou, G. Sun, S. Fu, J. Liu, X. Zhou, and J. Zhou, "A big data mining approach of pso-based bp neural network for financial risk management with iot," *IEEE Access*, vol. 7, pp. 154035–154043, 2019.

[125] J. Yu, S. Wang, and L. Xi, "Evolving artificial neural networks using an improved pso and dpso," *Neurocomputing*, vol. 71, no. 4-6, pp. 1054–1060, 2008.

[126] Y. Wang, H. Zhang, and G. Zhang, "cpso-cnn: An efficient pso-based algorithm for fine-tuning hyper-parameters of convolutional neural networks," *Swarm and Evolutionary Computation*, vol. 49, pp. 114–123, 2019.

[127] K. Janocha and W. M. Czarnecki, "On loss functions for deep neural networks in classification," *arXiv preprint arXiv:1702.05659*, 2017.

[128] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[129] C. L. Wight and J. Zhao, "Solving allen-cahn and cahn-hilliard equations using the adaptive physics informed neural networks," *arXiv preprint arXiv:2007.04542*, 2020.

[130] R. van der Meer, C. W. Oosterlee, and A. Borovykh, "Optimally weighted loss functions for solving pdes with neural networks," *Journal of Computational and Applied Mathematics*, vol. 405, p. 113887, 2022.

[131] D. Liu and Y. Wang, "A dual-dimer method for training physics-constrained neural networks with minimax architecture," *Neural Networks*, vol. 136, pp. 112–125, 2021.

[132] H. Xu, H. Zhang, Z. Hu, X. Liang, R. Salakhutdinov, and E. Xing, "Autoloss: Learning discrete schedules for alternate optimization," *arXiv preprint arXiv:1810.02442*, 2018.

[133] A. A. Heydari, C. A. Thompson, and A. Mehmood, "Softadapt: Techniques for adaptive loss weighting of neural networks with multi-part loss functions," *arXiv preprint arXiv:1912.12355*, 2019.

[134] R. Bischof and M. Kraus, "Multi-objective loss balancing for physics-informed deep learning," *arXiv preprint arXiv:2110.09813*, 2021.

[135] F. M. Rohrhofer, S. Posch, and B. C. Geiger, "On the pareto front of physics-informed neural networks," *arXiv preprint arXiv:2105.00862*, 2021.

[136] J. R. Baldwin, J.-B. Pingault, T. Schoeler, H. M. Sallis, and M. R. Munafò, "Protecting against researcher bias in secondary data analysis: challenges and potential solutions," *European Journal of Epidemiology*, vol. 37, no. 1, pp. 1–10, 2022.

[137] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.

[138] M. Preuss, M. Epitropakis, X. Li, and J. E. Fieldsend, "Multimodal optimization: Formulation, heuristics, and a decade of advances," in *Metaheuristics for Finding Multiple Solutions*, pp. 1–26, Springer, 2021.

[139] X. Li, "A multimodal particle swarm optimizer based on fitness euclidean-distance ratio," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pp. 78–85, 2007.

[140] S. Yazdani, H. Nezamabadi-Pour, and S. Kamyab, "A gravitational search algorithm for multimodal optimization," *Swarm and Evolutionary Computation*, vol. 14, pp. 1–14, 2014.

[141] A. Ahrari, K. Deb, and M. Preuss, "Multimodal optimization by covariance matrix self-adaptation evolution strategy with repelling subpopulations," *Evolutionary computation*, vol. 25, no. 3, pp. 439–471, 2017.

[142] S. Maree, T. Alderliesten, D. Thierens, and P. A. Bosman, "Benchmarking the hill-valley evolutionary algorithm for the gecco 2018 competition on niching methods multimodal optimization," *arXiv preprint arXiv:1807.00188*, 2018.

[143] S. Maree, T. Alderliesten, and P. A. Bosman, "Benchmarking hillvallea for the gecco 2019 competition on multimodal optimization," *arXiv preprint arXiv:1907.10988*, 2019.

[144] A. Dourado and F. A. Viana, "Physics-informed neural networks for missing physics estimation in cumulative damage models: a case study in corrosion fatigue," *Journal of Computing and Information Science in Engineering*, vol. 20, no. 6, 2020.

[145] T. Kadeethum, T. M. Jørgensen, and H. M. Nick, "Physics-informed neural networks for solving inverse problems of nonlinear biot's equations: Batch training," in *54th US Rock Mechanics/Geomechanics Symposium*, OnePetro, 2020.

[146] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "Deepxde: A deep learning library for solving differential equations," *SIAM Review*, vol. 63, no. 1, pp. 208–228, 2021.

[147] A. D. Jagtap, Z. Mao, N. Adams, and G. E. Karniadakis, "Physics-informed neural networks for inverse problems in supersonic flows," *arXiv preprint arXiv:2202.11821*, 2022.

[148] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[149] N. Moelans, B. Blanpain, and P. Wollants, "An introduction to phase-field modeling of microstructure evolution," *Calphad*, vol. 32, no. 2, pp. 268–294, 2008.

[150] C. Kunselman, V. Attari, L. McClenny, U. Braga-Neto, and R. Arroyave, "Semi-supervised learning approaches to class assignment in ambiguous microstructures," *Acta Materialia*, vol. 188, pp. 49–62, 2020.