Exploration and Analysis of Physics Informed Neural Network Architectures and Their Applications

James Byrne
Department of Computer and Software Engineering
Technology University of the Shannon, Athlone Campus
Athlone, Westmeath, Ireland
A00324581@student.tus.ie

Abstract *Physics-Informed Neural Networks (PINNs) are neural networks that incorporate the physics equations of the domain being modelled as part of the neural network itself. They are currently being used to solve various types of partial differential equations found in science and engineering and have shown promise over traditional methods such as finite element analysis. This article will provide a review of the common architectures currently being researched and areas for future research.*

Keywords *Physics–Informed Neural Networks, Scientific Machine Learning, Deep Neural Networks, Nonlinear equations, Numerical methods, Partial Differential Equations, Finite Element Analysis*

I. Introduction

While recent attention in Neural Networks and Deep Learning in general have focussed on Large-Language-Models, a lot of exciting work is happening in the area of Physics Informed Neural Networks (PINNs). PINNs are a merging of neural networks with traditional physics models. It exists as a subfield within the overall area of scientific machine learning (SciML). The addition of the physics equations to the neural network encourages consistency with the known physics of the domain being modelled thus leading to a better model. They also increase the generalisation of the models and thus allow for more accurate extrapolation.

An overview of what PINNs are is provided as well as a summary of the more common architectures types under investigation.

II. Physics Informed Neural Networks

A. What is a Physics Informed Neural Network

Physics Informed Neural Networks (PINNs) are a merging of neural networks with traditional physics models which are expressed using Partial Differential Equations (PDEs). The choice of using a neural network to solve PDEs is well established and supported by the universal approximation theorem [1] which implies that any continuous function can be arbitrarily closely approximated by a multi-layer perceptron. PINNs try to numerically solve the PDEs by using a neural network to minimise the loss function.

The addition of the physics equations to the neural network encourages consistency with the known physics of the domain being modelled thus leading to a better model. They also increase the generalisation of the models and thus allow more accurate extrapolation.

Fig 1 shows the three main scenarios where physics and data combine. Scientific mathematical models will involve lots of physics with small amounts of data to help validate the mathematical model (left-hand side). Typically neural networks try to create a model based on big data with no knowledge of the underlying physics (right-hand side). On the other hand, PINNs try to fill the gap in the middle where you have some knowledge of the underlying governing equations for the physics domain and some data. This data is typically gathered through direct measurement and is complicated, noisy and usually has a time component. Another issue is that not all the boundary conditions may be known or be measurable.

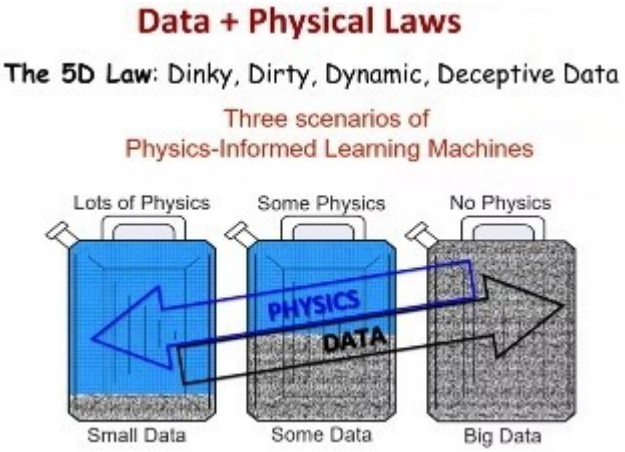


Fig 1: Role of Physics and Data

B. Role of Physics

Physics deals with the study of matter, energy and it's properties. These properties and their inter relationships are modelled though PDEs. An example is the heat equation *Fig 2*.

This equation describes the relationship between temperature (T) as a function of time (t) and distance (x). k/ρc is the heat diffusivity. For simple equations like this one, an analytic solution is normally available. The finite difference method [2] can also be used to numerically approximate the solution.

$$\frac{\partial T}{\partial t} = \frac{k}{\rho c}\frac{\partial^2 T}{\partial x^2}$$

Fig 2: Heat Equation in 1 Dimension

Maziar Raissi1, Paris Perdikaris, and George Em Karniadakis [3] proposed the PINN approach based on a feed forward neural network.

C. *Neural Network as an Optimiser*

PINNs as first envisioned by Maziar Raissi1, Paris Perdikaris2, and George Em Karniadakis rely on three main key features in neural networks to function:

- Availability of gradients (enabled by backpropagation)
- Loss function incorporating physics (custom loss function)
- Neural network learning algorithm

Taking the simple case of a neural network taking an input x and producing an output $\hat{y}$, *Fig 3*. The neural network structure is normally a multi-layer perceptron which generates $\hat{y}$ based on the input x and the expected value for y. The loss function is the difference between $\hat{y}$ and y and is a measure of the deviation or inaccuracy that we want to minimise. For a regular neural network the loss function would be solely due to the data fit and would depend on the nature of the problem, i.e. MSE (regression) or Cross Entropy (binary or multi-class classification problem) and be given by L(θ)

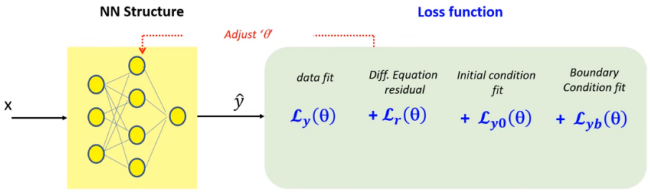


Fig 3: Loss Function Development

The aim of the learning algorithm is to use gradient descent to adjust the weight and biases of the neural network structure to minimise L(θ)

$$\theta' = \theta - \eta \nabla \theta L(\theta), \text{ where } \eta \text{ is the learning rate} \quad [1]$$

In order to solve the differential equation in our PINN there are three other loss functions terms which need to be accounted for:

- residual of the differential equation
- initial condition fit
- boundary condition fit

These three additional terms are added to the loss function with the aim of minimising them. Sometimes a term may be zero, for example if there is no initial boundary condition.

D. *Some examples*

M.Raissia, P.Perdikarisb, and G.E.Karniadakis [4] gave a couple of different examples of the use of PINNs.

In the first example *Fig 4*, they show the forward approximation of Schrodinger equation. The black crosses are the initial and boundary condition data. The governing PDE is then incorporated into the PINN thus allowing it to predict the solution across the total space as shown.

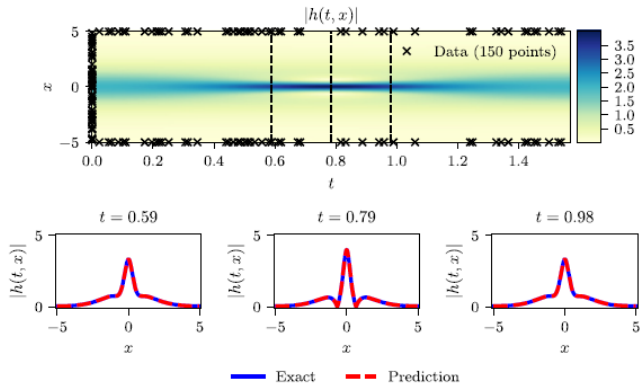
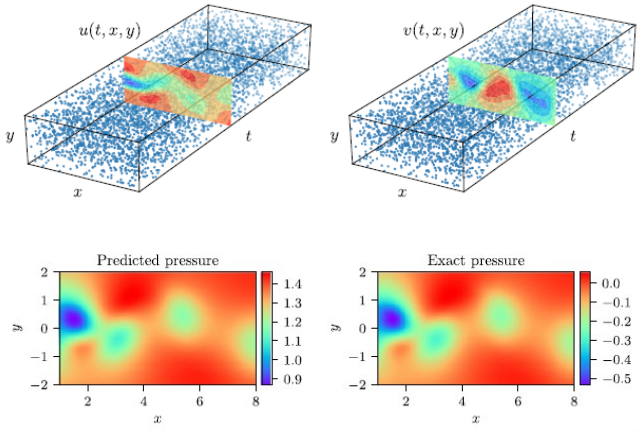


Fig 4: Forward approximation of Schrodinger equation

PINNs are also useful for the inverse discovery of solutions. In *Fig 5* velocity data was randomly distributed throughout the domain. There was no boundary or initial condition data. The PINN was able to achieve good agreement between the predicted and actual pressure fields.



Correct PDE	$u_t + (uu_x + vu_y) = -p_x + 0.01(u_{xx} + u_{yy})$ $v_t + (uv_x + vv_y) = -p_y + 0.01(v_{xx} + v_{yy})$
Identified PDE (clean data)	$u_t + 0.999(uu_x + vu_y) = -p_x + 0.01047(u_{xx} + u_{yy})$ $v_t + 0.999(uv_x + vv_y) = -p_y + 0.01047(v_{xx} + v_{yy})$
Identified PDE (1% noise)	$u_t + 0.998(uu_x + vu_y) = -p_x + 0.01057(u_{xx} + u_{yy})$ $v_t + 0.998(uv_x + vv_y) = -p_y + 0.01057(v_{xx} + v_{yy})$

Fig 5: Inverse discovery: Navier-Stokes equations

The PINN was also able to infer unknown parameters of the PDE including density and viscosity of the fluid.

III. Finite Basis PINNs (FBPINNs)

Finite Basis PINNs (FBPINNs) is a scalable domain decomposition approach proposed by [5] to address some of the limitations of the traditional PINN for large domains and/or multi-scale solutions.

A. *Architecture*

Rather than trying to solve the problem at hand using a single domain, FBPINNs splits the domain into several overlapping subdomains each of which contain their own neural network. Each neural network learns the full solution. In the overlapping regions, the solution is the sum of the overlapping neural networks. A smooth window function [6] ensures that each neural network is confined to its own subdomain. The input variables are separately normalized over each of the subdomains.

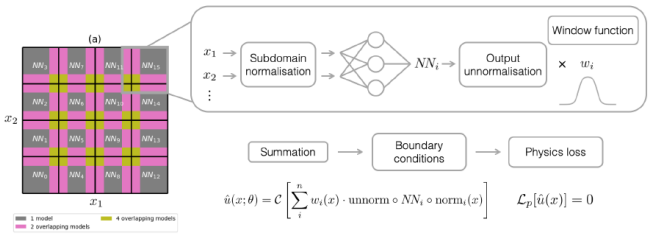


Fig 6: FBPINN Workflow

Flexibility in the scheduling of the training is used to improve convergence of the solution, with the solution learning from the boundary condition. FBPINNs address scaling issues by using domain decomposition, subdomain normalization, and parallel training to reduce the complexity of the optimization problem and thus ensure efficient convergence. This architecture enables the effective solution of large, multi-scale problems related to differential equations by combining neural networks with classical finite element methods.

B. *Key contributions and innovations*

FBPINNs use a combination of domain decomposition, individual subdomain normalisation, and flexible training schedules to try and eliminate some of the issues observed when scaling PINNs in particular for large domains and/or multi-scale solutions. They were also found to accurately solve both the smaller and larger scale problems which were studied. FBPINNs were also found to be more data-efficient than PINNs, and they could be trained in a parallel fashion. This is a major innovation as it will allow FBPINNs to scale more easily for larger and more complex domains thus ensuing their competitiveness over regular approaches such as finite difference or finite element methods.

C. *Performance on benchmark datasets*

The results cited in the paper indicate that FBPINNs can accurately solve problems where standard PINNs can struggle and especially for large domains. For smaller domains the results are similar to standard PINNs. Tests involving the wave equation and high-frequency sinusoids show that FBPINNs significantly outperform PINNs in terms of converging to a more accurate solution and requiring fewer training steps to do so.

D. *Strengths and limitations*

FBPINNs outperform standard PINNs for solving problems, especially on large domains. They achieve higher accuracy and require less training compared to PINNs on large-scale problems (wave equation, high-frequency sinusoids). They are also more data-efficient, needing smaller networks and less computation for training.

However FBPINNs do require fine-tuning the configuration for each problem due to varying scaling issues. Performance can also be affected by subdomain division (e.g., Burgers equation with discontinuity).

While they show promise they warrant further investigation and research into how factors such as subdomain size, higher dimensions, training data requirements and optimization complexity affect accuracy and if there's an optimal configuration. This research plus fine-tuning will be required for broader applicability.

IV. GEOMETRY-ADAPTIVE CONVOLUTIONAL NEURAL NETWORKS (PHYGEONET)

PhyGeoNet [7], is a novel approach that combines physics-informed learning with convolutional neural networks to solve parameterized steady-state partial differential equations (PDEs) on irregular domains with unlabelled data. To negate the limitation of CNNs being designed for image data and rectangular domains, an elliptic mapping *Fig 7* is used to transform from the irregular physical domain to a regular reference domain.

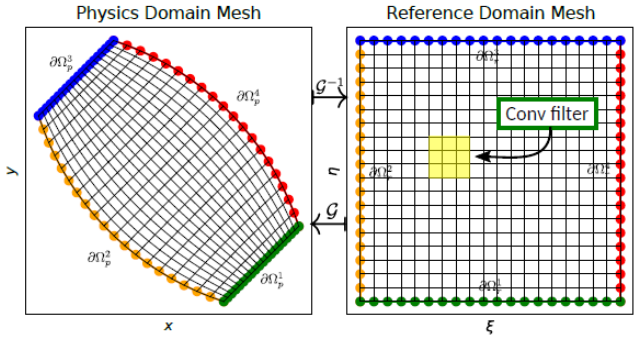


Fig 7: Diagram of the coordinate mappings between an irregular physical domain and regular reference domain

Physical parameters and coordinates of the physical domain are incorporated into the network architecture as input channels, allowing for both non-parametric and parametric solutions of PDEs.

A. *Key contributions and innovations*

The use of an elliptic mapping means means that cartesian-grid based CNN architectures can be used directly and without modification unlike graph/geodesic CNNs which require specific convolutional filters. This architecture allows the learning of solutions to PDEs without any training data.

This approach incorporates boundary conditions directly into the CNN architecture, ensuring they are strictly enforced for irregular geometries. It's also shown to be effective for both parametric heat equations and Navier-Stokes equations. The Navier-Stokes example also demonstrated that this CNN architecture was able to learn the solutions of the parametric equations on complex geometries without any labelled training data.

In a direct comparison to traditional fully-connected neural networks (FC-NNs) used in PINNs this CNN approach achieves superior accuracy and efficiency.

B. *Performance*

Overall the results Fig 8 shows that PhyGeoNet is quicker to converge and more than 10 times faster to train. The accuracy is also much higher if the total training budget is fixed to be the same.

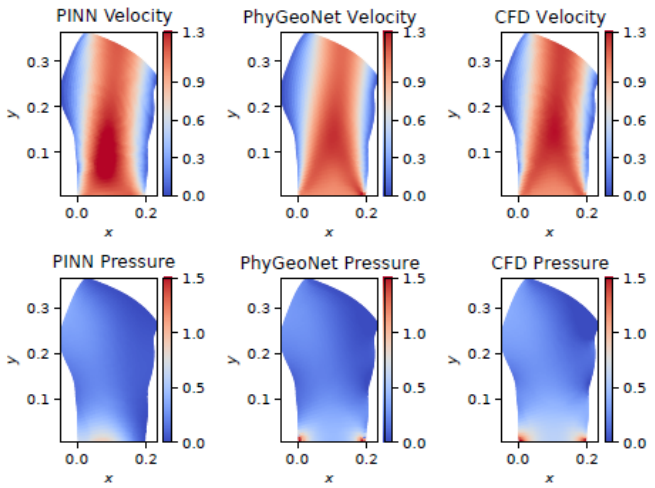


Fig 8: Velocity (the first row) and pressure (the second row) contours of PINN (left column), PhyGeoNet (middle column), and CFD benchmark (right column)

C. *Strengths and limitations*

PhyGeoNet demonstrates promising results in learning PDE solutions without labelled data. It also enables the solution of PDEs on irregular domains, expanding the applicability of physics-informed learning. And finally the incorporation of physical parameters into the CNN, enhancing the model's ability to capture complex physical phenomena.

While the current framework works well for parametrized steady-state PDEs further work is required to enable it to handle dynamic systems. Issues also exist when dealing with very complex domains with more than five continuous edges, limiting the applicability to highly intricate geometries. Another area for future development is it's application to the Navier-Stokes equations in turbulence regimes without labels.

V. PHYSICS-INFORMED GENERATIVE ADVERSARIAL NETWORKS (PIGANS)

Generative adversarial networks (GANs) are another architecture which shows great promise. Liu Yang, Dongkun Zhang and George EM Karniadakis [8] used a Wasserstein GANs with gradient penalty (WGAN-GP) to model stochastic processes from limited data. PI-GANs are composed of a discriminator, which is represented by a simple feed forward neural network, and of generators, which are a combination of feed forward DNNs and a neural network induced by the Stochastic Differential Equations (SDEs).

The architecture is shown in *Fig 9*. The generators for k and u and the discriminator are feed forward neural networks. The generators for f and b are neural networks representing the SDEs. The snapshots represent the sensor data.

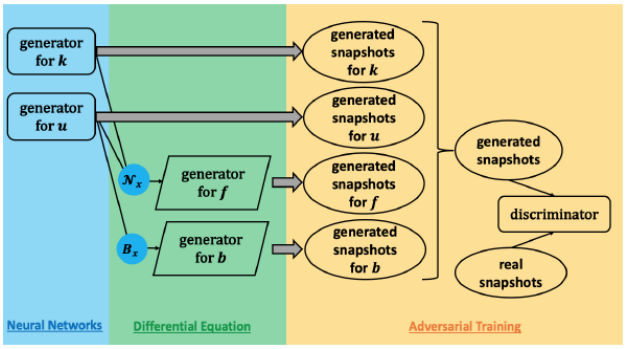


Fig 9: Schematic of solving SDEs with PI-GANs

A more general architecture with multiple discriminators was also investigated for cases where data are collected in multiple groups.

A. *Key contributions and innovations*

PIGANs can handle various scenarios including forward problems (finding solutions from known conditions), inverse problems (inferring unknown conditions from solutions), and mixed problems (incomplete information). They also achieve good accuracy for various SDEs, including high-dimensional problems (30 dimensions).

B. *Performance on benchmark datasets*

The results cited showed that PIGANs has clear advantages over regular GANs and will successfully approximate Gaussian processes and elliptic SDEs with good agreement with benchmarks.

C. *Strengths and limitations*

The PIGANs method can suffer from overfitting in both discriminators and generators and will miss small-scale details due to sparse sensor data. Noise in the measured data is also not accounted for. It's also more expensive to train compared to traditional physics-informed neural networks, especially for low-dimensional problems.

While PIGANs are a promising approach for solving SDEs, they require additional development in terms of data pre-processing, network optimisation, scalability and overall efficiency.

VI. CONCLUSION

This article presented an overview of PINNs and how they are a powerful approach for solving problems related to differential equations. They can perform well for high-dimensional PDEs and can be extended to solve inverse and discovery problems. They perform best on "messy/mixed" problems with some noisy data and the physics is not fully known. PINNs are mostly unsupervised.

However "vanilla" PINNs have some major limitations including computational cost, poor convergence and scaling to more complex problems.

It's also not obvious which neural network architecture to use for a particular problem domain. We have looked at the Feed Forward Neural Networks, Finite Basis PINNS, CNN and GANs. Work has been published on other architectures including adaptive activation functions, extreme learning machines. SIRENS, LSTM's.

PINNs remain an active and fast-moving field of research.

References

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

[2] "Finite Difference Method," https://en.wikipedia.org/wiki/Finite_difference_method. Accessed: Mar. 26, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Finite_difference_method

[3] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations," Nov. 2017, [Online]. Available: http://arxiv.org/abs/1711.10561

[4] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J Comput Phys*, vol. 378, pp. 686–707, Feb. 2019, doi: 10.1016/j.jcp.2018.10.045.

[5] B. Moseley, A. Markham, and T. Nissen-Meyer, "Finite Basis Physics-Informed Neural Networks (FBPINNs): a scalable domain decomposition approach for solving differential equations," Jul. 2021, [Online]. Available: http://arxiv.org/abs/2107.07871

[6] "Window function." Accessed: Mar. 28, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Window_function

[7] H. Gao, L. Sun, and J.-X. Wang, "PhyGeoNet: Physics-Informed Geometry-Adaptive Convolutional Neural Networks for Solving Parameterized Steady-State PDEs on Irregular Domain," 2020.

[8] L. Yang, D. Zhang, and G. E. M. Karniadakis, "Physics-informed generative adversarial networks for stochastic differential equations," *SIAM Journal on Scientific Computing*, vol. 42, no. 1, pp. A292–A317, 2020, doi: 10.1137/18M1225409.