TPINN: An improved architecture for distributed physics informed neural networks

Sreehari M
Department of Mechanical Engineering
Indian Institute of Technology Madras
Chennai, India 600036
me18s046@smail.iitm.ac.in

Balaji Srinivasan
Department of Mechanical Engineering
Indian Institute of Technology Madras
Chennai, India 600036
sbalaji@iitm.ac.in

Abstract

Significant progress has been made to obtain approximate solutions to PDEs using neural networks as a basis. One of these approaches (and the most popular and well-developed one) is the Physics Informed Neural Network (PINN). PINN has proved to provide promising results in various forward and inverse problems with great accuracy. However, PINN cannot be employed in its native form for solving problems where the PDE changes its form or when there is a discontinuity in the parameters of PDE across different sub-domains. Using separate PINNs for each sub-domain and connecting the corresponding solutions by interface conditions is a possible solution for this. However, this approach demands a high computational burden and memory usage. Here, we present a new method, Transfer Physics Informed Neural Network (TPINN), where one or more layer of PINN across different non overlapping sub-domains are changed keeping the other layers same for all the sub-domains. Solutions from different sub-domains are connected via problem specific interface conditions which are incorporated in to the loss function. We demonstrate the efficacy of TPINN through three heat transfer problems.

1 Introduction

Artificial Neural Network (ANN) has gained much popularity in the recent past and is used widely in different scientific domains for highly complex and nonlinear function approximation purposes. Thanks to the Universal approximation theorem, which validates the approximation capabilities of neural networks [1]. The solution of differential equations using neural networks is an active research area for the past several years. Lagaris et al. [2] used ANN to solve initial and boundary value problems defined on orthogonal box boundaries in which the trail solution consists of two parts; the first part satisfies initial/boundary condition and the second part involves a feedforward neural network containing adjustable parameters. PDE defined on geometry with complex boundary is solved using multilayer perceptron and radial basis functions in [3]. In [4, 5], nonlinear ordinary differential equations are solved using feedforward neural networks by determining the output weights using the weighted residual method.

Raissi et al. [6] developed an efficient neural network-based approach Physics Informed Neural Network (PINN) for solving forward and inverse problems involving nonlinear partial differential equations by using the benefits of automatic differentiation. It can combine the forward and the inverse problem in one platform effortlessly. However, PINN in its native form fails when PDE formulation changes or when there is a discontinuity in the parameters of PDE across different sub-domains. The sub-domains can be obtained by partitioning the global computational domain or physical sub-domains part of the problem definition, which adds up to the total computational domain. One way to solve this issue is to use separate PINNs for each sub-domain and connect the corresponding solutions by interface conditions [7]. This approach demands a high computational burden and memory usage.

In this paper, we present Transfer Physics Informed Neural Network (TPINN) in which hidden layers of PINNs are partially shared across sub-domains. Solutions from different sub-domains are connected using problem-specific interface conditions incorporated into the loss function. We found out that TPINN reduces memory requirements, computational burden, and appreciably enhance accuracy.

The paper is organized as follows. In Section 2 we do an overview of PINN. In Section 3 we present the algorithm of TPINN. In Section 4, we demonstrate the capability of TPINN using one forward and inverse problem of heterogeneous heat conduction. Finally, we conclude the paper in Section 5.

2 Physics informed neural network (PINN)

For solving PDEs Let the general form of a time dependant non-linear partial differential equation (PDE) for the function $u(x_1, x_2, .., x_N; t)$ of a time variable t and one or more spatial variables $x_1, x_2, .., x_N$ be:

$$\begin{aligned} &u_t + f(x_1, .., x_N; u; u_{x_1}, .., u_{x_N}; u_{x_1x_1}, .., u_{x_1x_N}, ..; \gamma), \\ &x = [x_1, x_2, .., x_N]^T \in \Omega \subset \mathbb{R}^N, \ t > 0, \end{aligned} \tag{1}$$

subject to appropriate boundary conditions and initial condition $u(x, 0) = u_o$. Let $\mathscr{F}^L : \mathbb{R}^{D_{in}}\mathbb{R}^{D_{out}}$ be a feed forward neural network with L layers and N_q neurons in the qth layer where $1 \leq q \leq L$. Input vector to the neural network is denoted by $\mathbf{y} \in \mathbb{R}^{D_{in}}$. If we denote the set of all weights and biases as $\Theta = \{\mathbf{W}^q, \mathbf{b}^q\}$, the output of the neural network is given by $\hat{\mathbf{u}}(\mathbf{y})_\Theta = \mathscr{F}^L(\mathbf{y}, \Theta)$. Consider Eq. (1). Let us define $g(x, t)$ to be given by the left-hand-side of Eq. (1); ie,

$$g(x, t) := u_t + f(x_1, .., x_N; u; u_{x_1}, .., u_{x_N}; u_{x_1x_1}, .., u_{x_1x_N}, ..; \gamma) \tag{2}$$

When we approximate $u(x, t)$ by a deep neural network which outputs $\hat{u}(x, t)$, we get a physics informed neural network (PINN) which approximates $g(x, t)$ as $\hat{g}(x, t)$ because of the assumption we made in Eqn. (2). This network can be constructed by applying the chain rule for differentiating compositions of functions using automatic differentiation. The total parameters of this neural network is learned by minimizing the mean squared loss $MSE_{PINN} = MSE_{pde} + MSE_{bc} + MSE_{ic}$ where, $MSE_{pde} = \frac{1}{N_{pde}} \sum_{i=1}^{N_{pde}} |\hat{g}(x^i_{pde}, t^i_{pde})|^2, MSE_{bc} = \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} |\hat{u}(x^i_{bc}, t^i_{bc}) - u(x^i_{bc}, t^i_{bc})|^2, MSE_{ic} = \frac{1}{N_{ic}} \sum_{i=1}^{N_{ic}} |\hat{u}(x^i_{ic}, t^i_{ic}) - u(x^i_{ic}, t^i_{ic})|^2$. $\{x^i, t^i\}$ represent the random scattered data points from the computational domain upon which the constraints to satisfy governing PDE / BC / IC are imposed. Fig. 1 demonstrates a two layer deep PINN used for forward problems.

For solving inverse problems In inverse problem we have unknown parameters γ of PDE in Eqn. (1) along with the additional information on points $\{x^i_{inv}, t^i_{inv}, \}_{i=1}^{N_{inv}} : \mathscr{L}(u, x_{inv}, t_{inv}) = 0$. PINN solves inverse problem just like forward problems by embedding the unknown parameters γ to Θ and they are learned by minimizing the loss $MSE_{PINN} = MSE_{pde} + MSE_{bc} + MSE_{ic} + MSE_{inv}$ where $MSE_{inv} = \frac{1}{N_{inv}} \sum_{i=1}^{N_{inv}} |\mathscr{L}(\hat{u}, x^i_{inv}, t^i_{inv})|^2$.

3 Transfer physics informed neural network (TPINN)

For solving PDEs Let the total computational domain Ω be made up of N_{SD} non overlapping sub-domains such that, $\Omega = \bigcup_{j=1}^{N_{SD}} \Omega_j$. Let $g_j(x, t)$ be the PDE formulation with solution $u_j(x, t)$ defined in Ω_j along with proper boundary conditions, initial condition and interface conditions. Let $\top^L_j : \mathbb{R}^{D_{in}}\mathbb{R}^{D_{out}}$ denote the neural network which approximates $u_j(x, t)$ corresponding to the jth sub-domain. For any $p \subset \{2, .., L\} \in \mathbb{N}$, let $\Theta_{shared} = \{\mathbf{W}^q, \mathbf{b}^q\}, \forall q \notin p, \Theta_j = \{\mathbf{W}^q, \mathbf{b}^q\}, \forall q \in p$, where Θ_j represents the set of weights and biases which are unique for each sub-domain neural network and Θ_{shared} is common to all sub-domain neural networks. The output of the neural network constructed using Θ_{shared} and Θ_j for jth sub-domain is given by $\hat{u}_j(x, t) = \hat{\mathbf{u}}_{\Theta_j, \Theta_{shared}} = \top^L_j(\mathbf{x}, \Theta_j, \Theta_{shared})$. Using different Θ_j and Θ_{shared}, we generate the neural network for each

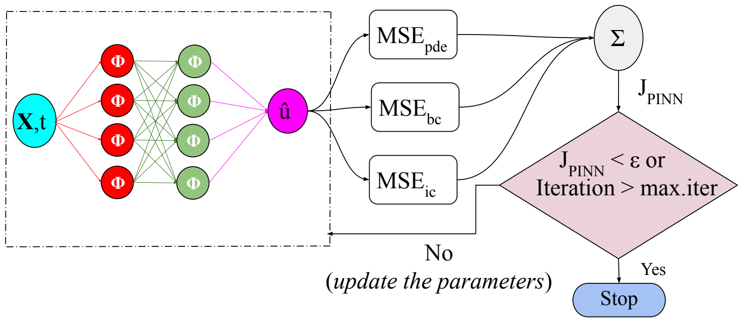


Figure 1: Schematic of a PINN used for forward problems.

sub-domain. The complete solution corresponding to the whole computational domain is given by $\hat{u}(x,t) = \hat{\mathbf{u}}(\mathbf{x})_\Theta = \bigcup_{j=1}^{N_{SD}} \top_j^L(\mathbf{x}, \Theta_j, \Theta_{shared})$. When the assumption we made in the Eqn. (2) is applied on the neural network $\hat{u}(x,t)$, we get *transfer physics informed neural network* (TPINN) which approximates each $g_j(x,t)$ as $\hat{g}_j(x,t)$ and $g(x,t)$ as $\hat{g}(x,t)$. The total parameters of TPINN is learned by minimizing the mean squared loss given by $J_{TPINN} = \lambda_{pde}J_{pde} + \lambda_{bc}J_{bc} + \lambda_{ic}J_{ic} + \lambda_{if}J_{if}$ where $J_{pde} = \sum_{j=1}^{N_{SD}}(\frac{1}{N_{pde}^j}\sum_{i=1}^{N_{pde}^j}|\hat{g}_j(x_{pde}^{i,j}, t_{pde}^{i,j})|^2)$, $J_{bc} = \sum_{j=1}^{N_{SD}}(\frac{1}{N_{bc}^j}\sum_{i=1}^{N_{bc}^j}|\hat{u}_j(x_{bc}^{i,j}, t_{bc}^{i,j}) - u_j(x_{bc}^{i,j}, t_{bc}^{i,j})|^2)$, $J_{ic} = \sum_{j=1}^{N_{SD}}(\frac{1}{N_{ic}^j}\sum_{i=1}^{N_{ic}^j}|\hat{u}_j(x_{ic}^{i,j}, t_{ic}^{i,j}) - u_j(x_{ic}^{i,j}, t_{ic}^{i,j})|^2)$. $\{x^{i,j}, t^{i,j}\}$ represent the random scattered data points from the jth sub-domain upon which the constraints to satisfy governing PDE / BC / IC / interface condition are imposed.
The interface loss J_{if} consists of problem specific interface conditions determined by the governing physical law. We give the information about the relationship between the latent variable u and its derivatives w.r.t the independent variables of two adjacent sub-domains to the model with the help of interface loss. Let $h(x, t, u, u_x, u_t, u_{xx}, .., \gamma)$ be the problem specific flux to be conserved across the interface. As an example, for a heat conduction problem, h, u, γ represent the local heat flux density, temperature and thermal conductivity respectively. The conservation of heat flux density $h = -\gamma\nabla u \cdot \hat{\mathbf{n}}$, where $\hat{\mathbf{n}}$ is the unit vector normal to the interface, which is governed by Fourier's law of heat conduction will be included in the interface loss. In general the interface loss is given by, $J_{if} = \sum_{k=1}^{N_I}(\frac{1}{N_{if}^k}\sum_{i=1}^{N_{if}^k}|\hat{u}_{k^+}(x_{if}^{i,k}, t_{if}^{i,k}) - \hat{u}_{k^-}(x_{if}^{i,k}, t_{if}^{i,k})|^2) + \sum_{k=1}^{N_I}(\frac{1}{N_{if}^k}\sum_{i=1}^{N_{if}^k}|\hat{h}_{k^+} - \hat{h}_{k^-}|^2)$ where N_I is the total no. of interfaces. $\{x_{if}^{i,k}, t_{if}^{i,k}\}_{i=1}^{N_{if}^k}$ indicate the training data points sampled from the kth interface to inform the interface conditions connecting the solutions of the adjacent k^+th and k^-th sub-domains to the model. $\hat{h}_{k^+}$, $\hat{h}_{k^-}$ are the flux values approximated by the neural networks corresponding to the k^+th and k^-th sub-domains respectively at the interface. $[\lambda_{pde}, \lambda_{bc}, \lambda_{ic}, \lambda_{if}]$ are penalty factors which penalizes the violation of the constraints in the corresponding loss. Algorithm of TPINN is shown in Algorithm. 1. Fig. 2 illustrates the schematic of a TPINN.

For solving inverse problems In inverse problem we have unknown parameters γ_j of PDE in Eqn. (1) along with the additional information on points $\{x_{inv}^{i,j}, t_{inv}^{i,j},\}_{i=1}^{N_{inv}^j} : \mathscr{L}_j(u_j, x_{inv}^j, t_{inv}^j) = 0$ for each jth sub-domain. TPINN solves the inverse problem just like forward problems by embedding the unknown parameters γ_j to Θ_j and are learned by minimizing the loss $J_{TPINN} = \lambda_{pde}J_{pde} + \lambda_{bc}J_{bc} + \lambda_{ic}J_{ic} + \lambda_{if}J_{if} + \lambda_{inv}J_{inv}$, where $J_{inv} = \sum_{j=1}^{N_{SD}}(\frac{1}{N_{inv}^j}\sum_{i=1}^{N_{inv}^j}|\mathscr{L}_j(\hat{u}_j, x_{inv}^{i,j}, t_{inv}^{i,j})|^2)$.

4 Demonstration examples

We solve one forward and two inverse problems of heterogeneous heat conduction. PINN performs poorly here because of the thermal conductivity discontinuity at the interface of two different materials.

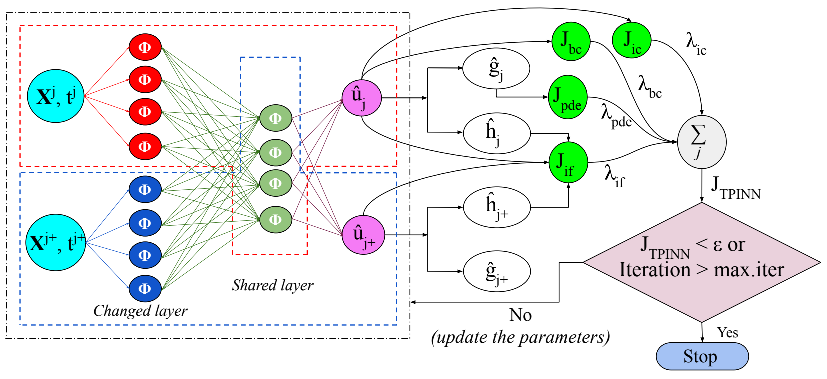


Figure 2: Schematic of a TPINN consisting two hidden layers with 4 neurons in each hidden layer. The first hidden layer is changed across the adjacent jth and j^+th sub-domains $\forall\ j$.

Algorithm 1 TPINN algorithm

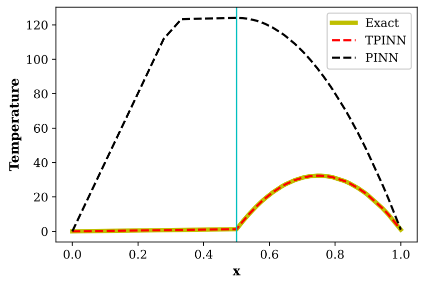
```
Step 1: Identify the non-overlapping sub-domains.
Step 2: Decide the architecture of TPINN by fixing the no. of layers ($N_L$) and no. of neurons in each layer ($N_q$) for each sub-domain.
Step 3: Decide the set $p$ which contains the layer(s) of the TPINN to be changed across each sub-domain.
Step 4: Construct TPINN for $j$th sub-domain using $\Theta_{shared}$ and $\Theta_j$, $j$ = 1,2,...,$N_{SD}$.
Step 5: Prepare random scattered training data points from all sub-domains.
Step 6: Prepare random scattered training data points from all the interfaces.
Step 7: Choose proper values for penalty factors.
Step 8: Initialize the parameters of the neural networks constructed in Step 4 using suitable initialization method. Train these neural networks for the best parameters using the training data prepared in Step 5 & Step 6 by minimizing the loss $J_{TPINN}$ using a suitable optimization method. $\Theta^*$ = arg min$_\Theta(J_{TPINN})$
```

4.1 1D steady-state heat conduction in a two-layered composite slab with heat generation

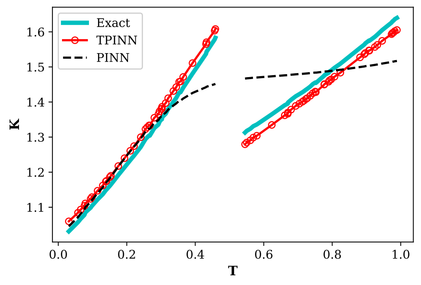
Consider the system $\frac{\partial}{\partial x}(K\frac{\partial T}{\partial x})+Q = 0$ with $K_1 = 1, Q_1 = 0$ for $x \in [0, 0.5]$ and $K_2 = 0.01, Q_2 = -10$ for $x \in [0.5, 1]$. Boundary conditions are $T(x = 0) = 0$ & $T(x = 1) = 1$. We solve this strong heterogeneous forward problem of finding the temperature distribution along the slabs using TPINN with the first hidden layer change taking each slab as a sub-domain. The depth and width of the neural networks used for both sub-domains are 1 and 100, respectively. We used 38 random interior data points and 2 data points each at the boundary and interface nodes per sub-domain for training. A penalty factor of 1 is used for all losses. Optimization is carried out initially using Adam optimizer with a learning rate 0.001 for 5000 iterations followed by Sequential Least-Squares Programming (SLSQP). Fig. 3a illustrates the temperature prediction by PINN and TPINN.

4.2 The inverse problem of determining temperature dependant thermal conductivity of two-layered composite slab in a 1D steady-state heterogeneous heat conduction

Consider the non-linear system $\frac{\partial}{\partial x}(K(T)\frac{\partial T}{\partial x}) = 0$ with $T_1(x) = ln(1.3884x + 1)$ & $K_1(T) = e^T$ for $x \in [0, 0.5]$ and $T_2(x) = 2ln(0.6942x + 0.9545)$ & $K_2(T) = e^{0.5T}$ for $x \in [0.5, 1]$. 40 noisy temperature values and a heat flux value from random interior x locations per sub-domain are used to estimate the thermal conductivity. No boundary and interface points are used. Each slab is taken as a sub-domain. Here we cannot set $K(T)$ as a parameter of the neural network, as it is a function of T. So we construct two TPINNs here; T-TPINN (input: x & output: T) for temperature and K-TPINN (input: T & output: K) for thermal conductivity. We have adopted the first layer change for T-TPINN and last layer change for K-TPINN. The depth and width of the neural networks used for both sub-domains are 3 and 16, respectively. Optimization is accomplished initially using Adam optimizer with a learning rate of 0.001 for 150000 iterations, followed by the SLSQP optimizer. Penalty factors are $\lambda_{pde} = 1, \lambda_{inv}$ for T prediction loss is 5 and for flux prediction loss is 1. Fig. 3b illustrates the thermal conductivity predictions by PINN and TPINN. As we can see, TPINN's prediction is agreeing with the exact solution reasonably well.



(a) Temperature comparison for problem 4.1.



(b) Thermal conductivity prediction for problem 4.2.

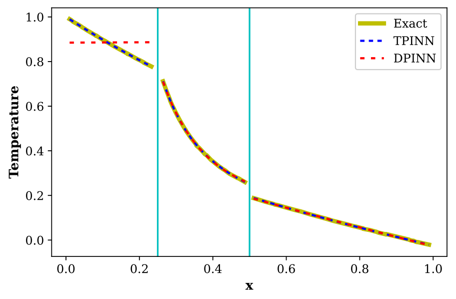
Figure 3

4.3 The inverse problem of determining spatially varying thermal conductivity of three-layered composite slab in a 1D steady-state heterogeneous heat conduction

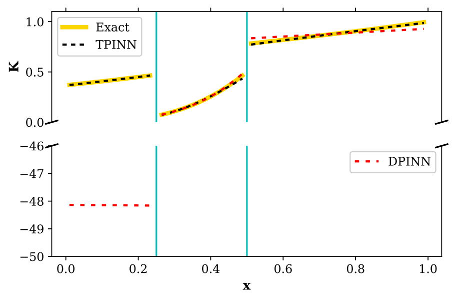
Consider the system $\frac{\partial}{\partial x}(K(x)\frac{\partial T}{\partial x}) = 0$ with $K_1(x) = e^{(x-1)}$, $T_1(x) = 1 - 0.3655(e - e^{1-x})$ for $x \in [0, 0.25]$, $K_2(x) = 4x^3$, $T_2(x) = 1 - 0.3655(2.601 - \dfrac{1}{8x^2})$ for $x \in (0.25, 0.5]$, $K_3(x) = e^{0.5(x-1)}$, $T_3(x) = 1 - 0.3655(4.669 - 2e^{0.5(1-x)})$ for $x \in (0.5, 1]$. 40 noisy temperature values and a heat flux value from random interior x locations per sub-domain are used to estimate the thermal conductivity. No boundary and interface points are used. Each slab is taken as a sub-domain. Here we cannot set $K(x)$ as a parameter of the neural network, as it is a function of x. So we construct TPINN with the first hidden layer change with input: x & output: (T, K). The depth and width of the neural network used for all sub-domains are 4 and 16, respectively. Optimization is accomplished initially using Adam optimizer with a learning rate of 0.001 for 20000 iterations, followed by the SLSQP optimizer. Penalty factors are $\lambda_{pde} = 1, \lambda_{inv}$ for T prediction loss is 1.5 and for flux prediction loss is 1. We solve this problem with DPINN [7] also. DPINN uses independent and different PINN with its own set of parameters for each sub-domain. The depth and width of the neural network used in DPINN for all sub-domains are 4 and 9, respectively; thus, both TPINN and DPINN have nearly the same number of total parameters. Fig. 4a illustrates the temperature predictions by TPINN & DPINN and Fig. 4b illustrates the thermal conductivity predictions by TPINN & DPINN. Even though DPINN is able to predict with a good accuracy in second and third sub-domains, DPINN is unable to achieve accuracy in the first sub-domain. TPINN is able to predict with excellent accuracy in all three sub-domains.

5 Concluding remarks

In this paper, we introduce Transfer Physics Informed Neural Network (TPINN) for solving forward and inverse problems respecting the PDE, initial condition, and boundary conditions specified. TPINN is constructed by changing the hidden layers of PINN partially for each sub-domains. TPINN outperforms PINN when there is a discontinuity in the parameter of PDE across non-overlapping sub-domains. The numerical examples shown confirm the efficacy of TPINN in solving the forward and inverse problems effectively, especially in heterogeneous domains. Also, the inverse problems in a heterogeneous domain are hard to solve even with the conventional approaches, which involve the execution of forward and inverse problems iteratively until convergence. TPINN replaces this conundrum and solves the inverse problem efficiently by fitting a surrogate model.



(a) Temperature prediction for problem 4.3.



(b) Thermal conductivity prediction for problem 4.3

Figure 4

Acknowledgments

This work was supported by Robert Bosch Centre for Data Science and Artificial Intelligence, Indian Institute of Technology Madras, Chennai (Project No. CR1920ME615RBCX008832).

References

[1] K. Hornik (1991) Approximation capabilities of multilayer feedforward networks. Neural Netw 4(2):251–257.

[2] Lagaris IE, Likas A, Fotiadis DI (1998) Artifcial neural networks for solving ordinary and partial diferential equations. IEEE Trans Neural Netw 9:987–1000

[3] Lagaris IE, Likas A (2000) Neural-network methods for boundary value problems with irregular boundaries. IEEE Trans Neural Netw 11(05):1041–1049

[4] A.J. Meade Jr., A.A. Fernandez (1994) The numerical solution of linear ordinary differential equations by feedforward neural networks. Mathematical and Computer Modelling 19(12):1–25.

[5] A.J. Meade Jr., A.A. Fernandez (1994) Solution of nonlinear ordinary differential equations by feedforward neural networks. Mathematical and Computer Modelling 20(9):19–44.

[6] M. Raissi, P. Perdikaris, G.E. Karniadakis (2018) Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. Journal of Computational Physics 378:686-707

[7] V. Dwivedi, N. Parashar, B. Srinivasan (2020) Distributed learning machines for solving forward and inverse problems in partial differential equations. Neurocomputing. https://doi.org/10.1016/j.neucom.2020.09.006