Journal of Mechanical Science and Technology 38 (7) 2024

Review Article
DOI 10.1007/s12206-024-0624-9

Keywords:
· Deep learning
· Learning multiple instances
· Physics-informed neural network
· PINN applications
· PINN architecture
· Reduced order modeling

Correspondence to:
Dong-Hoon Choi
dhchoi@pidotech.com

Citation:
Ryu, I., Park, G.-B., Lee, Y., Choi, D.-H. (2024). Physics-informed neural network for engineers: a review from an implementation aspect. Journal of Mechanical Science and Technology 38 (7) (2024) 3499~3519.
http://doi.org/10.1007/s12206-024-0624-9

Received September 19th, 2023
Revised March 6th, 2024
Accepted March 25th, 2024

† Recommended by Editor Hyun-Gyu Kim

Physics-informed neural network for engineers: a review from an implementation aspect

Ikhyun Ryu, Gyu-Byung Park, Yongbin Lee and Dong-Hoon Choi

PIDOTECH Inc., 114, Beobwon-ro, Songpa-gu, Seoul 05854, Korea

Abstract In order to offer guidelines for physics-informed neural network (PINN) implementation, this study presents a comprehensive review of PINN, an emerging field at the intersection of deep learning and computational physics. PINN offers a novel approach to solve physics problems by leveraging the flexibility and scalability of neural networks, even with small or no data. First, a general description of different physics problem types and target tasks addressable with PINN was provided. A generic PINN architecture was described in detail using a component-wise approach, with components ranging from collocation points to optimization methods. Then, we surveyed studies that sought to improve upon each of these components. To offer practical insights, we highlighted studies that focused on key issues of PINN implementation and showcased three practical applications. Lastly, a summary and potential research directions were provided to offer guidelines for reliable and customized PINN implementations.

1. Introduction

Based on the recent advancements in deep learning technology, artificial intelligence has led to success in various fields. With the increasing accessibility of big data, advances in sensor technology, and high-performance hardware sources such as GPUs and NPUs, deep learning technology is expected to continue to grow. However, in engineering design, the acquisition of labeled data is often costly and time-consuming, limiting the full utilization of such advanced techniques. Therefore, several attempts have been proposed to solve engineering problems by incorporating physics knowledge that has been built up over hundreds of years into artificial neural networks (ANNs) in addition to available sparse data if any. Those were named "physics-informed neural networks", "physics-based neural networks", "physics-guided neural networks", or "theory-guided neural networks" [1].

Among different names, most recent studies utilized the name “physics-informed neural network (PINN)”, and we will also use ‘PINN’ as the representative name in this review paper. Several early studies had attempted to incorporate physics knowledge into ANNs (refer to Appendix for more details), upon which recent PINNs were developed. Currently considered one of the most significant starting points, Raissi et al. [2] presented a PINN in 2019 (later referred to as *Vanilla* PINN) that computes the solution of the governing equations by training the network using the collocation points within the computational domain along with the automatic differentiation (AD). Afterwards, numerous studies have been proposed to handle physics problem types other than PDEs and to improve the accuracy and/or computational efficiency by modifying some parts of the PINN architecture, which will be reviewed in this paper.

As numerous studies have been suggested to enhance the performance of PINN, some recent review papers summarizing major studies have also been made available. Cai et al. [3] presented a review of the applications of PINN in fluid mechanics, as well as some case studies of flow problems to illustrate the methodologies. The case studies covered three-dimensional incompressible, two-dimensional steady compressible and biomedical flows. Here, thorough explanations of the problem setup, implementation of PINN, and outcomes for each case study were provided. Similarly, Cai et al. [4] presented a review of PINN applications in heat transfer problems and applications in the thermal design of power electronics. Additionally, general reviews on PINN and some of its applications were discussed by Karniadakis et al. [5] and Cuomo et al. [6]. Both reviews highlighted studies that enhanced the performance of the *Vanilla* PINN and presented their key concepts along with some theoretical aspects. These reviews can help to understand the principles of PINN and its modifications and applications. Engineers from different fields, however, might not have a clear understanding of how to appropriately select and apply the many studies that are ongoing simultaneously.

This review attempts to provide a guideline for implementing PINN improvements and presents three practical engineering applications that illustrate the implementation of some of the PINN improvements reviewed. In Sec. 2, we describe physics problem types and target task types for which PINN can be utilized. A generic PINN architecture is presented in Sec. 3 to set the stage for Sec. 4's description of research that focused on improving each component (part) of the PINN architecture. Sec. 5 reviews studies on what the authors consider to be some of the key issues in PINN. In Sec. 6, three exemplary applications of PINN to practical engineering problems are provided for readers to better comprehend the effect of the studies discussed in Secs. 4 and 5. Finally, Sec. 7 concludes with a summary of the paper and a discussion of prospective study areas.

2. Types of physics problem and target task

This section categorizes the types of physics problems and target tasks for which PINN is utilized, providing background knowledge for the studies of PINN discussed in the following sections. A physics problem type is defined by governing equations as well as initial conditions and/or boundary conditions, which are to be explained in Secs. 2.1 and 2.2, respectively. In Sec. 2.3, additional physics knowledge that can be considered is explained, which consist of algebraic equations and knowledge graph. The target task types are then described in Sec. 2.4.

2.1 Governing equations

There exist various types of differential equations in which PINN can be implemented: ordinary differential equations (ODEs), partial differential equations (PDEs), fractional differential equations (FDEs), integro-differential equations (IDEs), and stochastic differential equations (SDEs).

ODEs can be generally formulated as

$$F(\mathrm{x}, y, y', y'', \ldots, y^{(n)}) = 0, \tag{1}$$

where y denotes the unknown solution, where derivatives of which are taken with respect to input x (often representing time).

PDEs, most commonly used to express various physical phenomena, can be classified into a steady state and unsteady state by temporal dependence. In most cases, engineering problems are described by a composition of unsteady PDEs and can be generally formulated as

$$F(\mathbf{x}, t, y(\mathbf{x}, t), \partial_t y, \partial_{\mathbf{x}} y, \ldots, \partial_t{}^{(n)} y, \partial_{\mathbf{x}}{}^{(n)} y) = 0, \tag{2}$$

where the unknown solution $y(\mathbf{x}, t)$ depends on more than one independent variables, as opposed to one in ODEs.

FDEs, another type of differential equations that contain non-integer differential operators, can be expressed as

$$D^q y(x) = f(x, y(x), y'(x), \ldots, y^n(x)), \tag{3}$$

$$D^q_{x_0} y(x) := \frac{1}{\Gamma(m-q)} \frac{d^m}{dx^m} \int_{x_0}^{x} \frac{y(u)}{(x-u)^{q-m+1}} du, \tag{4}$$

where D^q denotes the fractional derivative of order q, and f denotes the forcing function represented by the input x, output y and its derivatives up to order of n. Eq. (4) shows the definition of D^q, where m denotes the smallest integer with $m > q$ and Γ denotes the gamma function for generalization of the factorial function.

IDEs contain both derivatives and integrals of the output y, which can be expressed as the following:

$$F\left(x, y(x), y'(x), \ldots, y^n(x), \int g(x, t, y(t)) dt\right) = 0, \tag{5}$$

where $y(x)$ denotes the unknown solution to be computed, g is a known function, the integral of which provides the past values of y as time progresses.

SDEs express random parameters of the problem as stochastic processes, which can be generally formulated as the following:

$$N_x[y(\mathbf{x};\omega); k(\mathbf{x};\omega)] = 0, \mathbf{x} \in D, \omega \in \Omega, \tag{6}$$

where $k(\mathbf{x};\omega)$ denotes the random parameter, ω denotes the random input in random space Ω.

2.2 Initial conditions and boundary conditions

Boundary conditions (BCs) can be expressed in forms of Dirichlet, Neumann, Robin, periodic boundary conditions, etc. In most of PINN studies, initial conditions (ICs) and BCs are enforced in a ‘soft’ manner, i.e., as loss terms in a loss function. As such, by training the neural network, the loss function is to be minimized and corresponding IC/BCs are satisfied, along with the governing equations. More detailed formulation of the loss function is provided in Sec. 3.

2.3 Additional physics knowledge

In the scope of PINN, prescribed differential equations and corresponding IC/BCs are mainly considered as the physics problem to be solved. However, in practical problems, it may be beneficial to incorporate additional physics knowledge that cannot be described with differential equations and IC/BCs.

As one of the additional physics knowledge, algebraic equations can be incorporated into the governing equations and/or constraints to further enforce some known physics, which could potentially enhance the performance of PINN.

In case there exist fixed relationships or correlations between input variables, such characteristics can be described by formulating a knowledge graph. To enforce the knowledge graph into PINN, a special type of neural networks, graph neural networks (GNNs), can be utilized to process the knowledge graph as an input of PINN. More details on incorporating GNNs into PINN are described further in Sec. 4.2.9.

It is worth noting that creating a general framework of problem formulation that can be applied to numerous engineering domains of interest is challenging. Engineers are therefore encouraged to experiment with and incorporate different types of physics knowledge in their fields of interest for PINN application.

2.4 Target task types

PINN is mainly utilized to solve the following target task types: forward problem, inverse problem, and uncertainty quantification.

The forward problem is one of the most common target task types considered in engineering, where the physics are expressed as a set of known differential equations and ICs and/or BCs: for example, in heat and mass transfer, fluid mechanics, solid mechanics, etc. With or without any available labeled experimental/simulation data, the neural network can be trained to comply to the given governing set of differential equations and ICs and/or BCs and provide the desired solution with appropriate formulation of PINN.

The inverse problem is also another common target task type in engineering, where some parameters or properties of the physics system are unknown. In this case, some sparse labeled experimental/simulation data are required for precise training of the network and for inferring the desired solution as well as the unknown parameters or properties. One of the applications includes characterization of the material properties in material sciences, as the underlying problems may be of ill-posed and inverse type. As solving the inverse problem is often more challenging than solving the forward problem, more sophisticated modifications of PINN may be required.

Uncertainty quantification (UQ) is the other target task type, where the uncertainties of the problem are to be quantified with PINN. UQ is often required when the random perturbations that affect the physical phenomena are to be considered, hence has important applications in various scientific and engineering fields. As described in Sec. 2.1, SDEs define the physical system as a stochastic process with random variables, aiming to express the randomness of the problem in the form of differential equations. As such, by solving the SDEs with PINN, noisy and stochastic phenomena from random inputs can be quantified and further be analyzed.

3. Physics-informed neural network architecture

A general architecture of PINN can be depicted as in Fig. 1. At first, collocation points are selected. Then, a neural network $\mathcal{N}(\mathbf{z};\boldsymbol{\theta})$ takes the space-time coordinate vector $\mathbf{z} := [x_1, x_2, \ldots, x_n; t] = [\mathbf{x}; \mathrm{t}]$ as an input and computes approximation of an output to the problem. Such neural network approximation of the output *u* is described as

$$\hat{u}_{\boldsymbol{\theta}}(\mathbf{z}) \approx u(\mathbf{z}), \tag{7}$$

where $\hat{u}_{\boldsymbol{\theta}}$ represents a neural network approximation realized with a set of neural network parameters $\boldsymbol{\theta}$. The approximate solution is then substituted for differentiation to compute the derivatives included in the governing equations using automatic differentiation (AD) or numerical differentiation (ND). Then, loss terms, representing errors between estimated values and true values, are evaluated. There can be four kinds of loss terms in PINN: L_0 for initial conditions, L_b for boundary conditions, L_f for governing equations, and L_d for labeled data within the computational domain. A weighted sum of the four loss terms in Eq. (8) can be used to represent the loss function to be minimized for neural network training.

$$L(\boldsymbol{\theta}) = \omega_0 L_0 + \omega_b L_b + \omega_{\mathrm{f}} L_{\mathrm{f}} + \omega_d L_d, \tag{8}$$

where

$$L_0 = \frac{1}{N_0}\sum_{i=1}^{N_0}\left|g(\mathbf{x}_0^i, t=0)\right|^2, \tag{9}$$

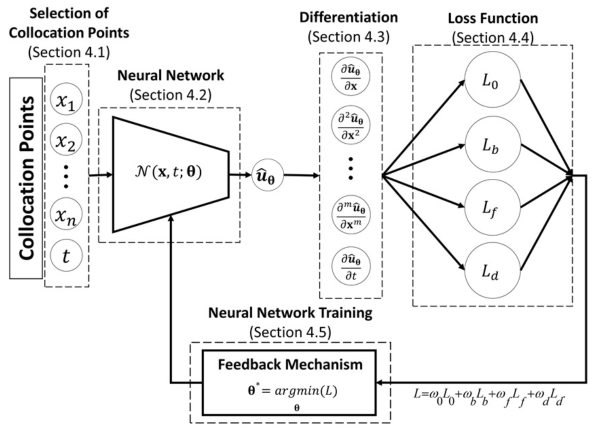
Selection of Collocation Points (Section 4.1)
Collocation Points
x_1
x_2
x_n
t
Neural Network (Section 4.2)
$\mathcal{N}(\mathbf{x}, t; \boldsymbol{\theta})$
$\hat{\mathbf{u}}_{\boldsymbol{\theta}}$
Differentiation (Section 4.3)
$\frac{\partial \hat{\mathbf{u}}_{\boldsymbol{\theta}}}{\partial \mathbf{x}}$
$\frac{\partial^2 \hat{\mathbf{u}}_{\boldsymbol{\theta}}}{\partial \mathbf{x}^2}$
$\frac{\partial^m \hat{\mathbf{u}}_{\boldsymbol{\theta}}}{\partial \mathbf{x}^m}$
$\frac{\partial \hat{\mathbf{u}}_{\boldsymbol{\theta}}}{\partial t}$
Loss Function (Section 4.4)
L_0
L_b
L_f
L_d
Neural Network Training (Section 4.5)
Feedback Mechanism
$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{argmin}(L)$
$L=\omega_0 L_0+\omega_b L_b+\omega_f L_f+\omega_d L_d$

Fig. 1. Physics-informed neural network architecture.

$$L_b = \frac{1}{N_b}\sum_{i=1}^{N_b}\left|g(\mathbf{x}_b^i, \mathrm{t}_b^i)\right|^2, \tag{10}$$

$$L_\mathrm{f} = \frac{1}{N_\mathrm{f}}\sum_{i=1}^{N_\mathrm{f}}\left|f(\mathbf{x}_\mathrm{f}^i, t_\mathrm{f}^i)\right|^2, \tag{11}$$

$$L_d = \frac{1}{N_d}\sum_{i=1}^{N_d}\left|u(\mathbf{x}_d^i, t_d^i) - \mathrm{u}^i\right|^2. \tag{12}$$

Here, g and f denote arbitrary initial/boundary functions and differential equations, N_0 denotes the number of collocation points for ICs, $\{\mathbf{x}_0^i, \mathrm{t}_0\}_{i=1}^{N_0}$, N_b the number of collocation points for BCs, $\{\mathbf{x}_b^i, \mathrm{t}_b^i\}_{i=1}^{N_b}$, N_f the number of collocation points in the computational domain, $\{\mathbf{x}_\mathrm{f}^i, \mathrm{t}_\mathrm{f}^i\}_{i=1}^{N_\mathrm{f}}$. In case any additional labeled data are available, it can be included for training; N_d denotes the number of labeled data in the computational domain, $\{\mathbf{x}_d^i, \mathrm{t}_d^i, u^i\}_{i=1}^{N_d}$. Depending on the type of a physics problem to be solved and availability of labeled data, some weight values can become zero. For example, if no labeled data is available, ω_d = 0. Finally, the loss function is fed into a feedback mechanism to update neural network parameters $\boldsymbol{\theta}$, and the updated $\boldsymbol{\theta}$ is used for the neural network at the next iteration. The whole process iterates until the values of neural network parameters converge to the optimum values, $\boldsymbol{\theta}^*$, that minimize the loss function.

4. Component-wise studies of PINN architecture

This section describes studies that modified each component of the PINN architecture depicted in Fig. 1 to enhance the performance. The studies in this section can be adaptively applied to the fields of interest to build a 'customized' PINN model. An overview of the studies for the five components of PINN is provided in Table 1.

4.1 Selection of collocation points

PINN is a collocation-based technique that requires appropriate selection of collocation points within the computational domain for proper training of PINN. Instead of simple uniform or random sampling for the selection of collocation points, various sampling techniques have been proposed and applied to improve the accuracy and/or computational speed of PINN.

Sampling techniques can be classified into two types: non-adaptive (domain-based) sampling and adaptive (response-based) sampling. Non-adaptive sampling is a one-shot sampling technique that uses only information from input variables to select all necessary collocation points. On the other hand, adaptive sampling is a sequential sampling technique that repeatedly proceeds with sampling by using information from both input variables and corresponding response values until a predefined goal is reached. As such, non-adaptive sampling techniques select collocation points only once before the start of neural network training. However, adaptive sampling techniques select the collocation points sequentially by using both response values and input variable values until the predefined goal is reached.

Das and Tesfamariam [7] compared the performance of PINN on five PDE examples using ten different non-adaptive sampling techniques: full factorial design (FFD), central composite design (CCD), centroidal voronoi tessellation (CVT), maximin latin hypercube (MLH), Sobol, Halton, Hammersley, Faure, full grid design (FGD), and sparse grid design (SGD). Hammersley sampling, one of the quasi-random sampling techniques, showed the best performance, followed by SGD and Sobol sampling. Mean squared errors (MSEs) were compared to evaluate performance based on the assumption that the analytical solution of each of the five PDE examples is the true solution. Furthermore, Mou et al. [8] introduced a mixed sampling technique that combined cartesian grid sampling and Latin hypercube sampling, where the collocation points were selected by adjusting the ratio of each sampling technique according to the problem. The ratio of each sampling technique was determined through a trial-and-error process.

As the first adaptive sampling technique applied in PINN, residual-based adaptive refinement (RAR) proposed by Lu et al. [9] was considered, which is a greedy algorithm that samples additional points in the locations with large PDE residual values at each training iteration. Hanna et al. [10] proposed another adaptive sampling technique named residual-based adaptive distribution (RAD), which samples additional points in a distributed manner using the prescribed probability density function (PDF) proportional to PDE residual values, instead of using just the residual values as in RAR. Wu et al. [11] then proposed the adaptive sampling technique named residual-based adaptive refinement with distribution (RAR-D) that combined the ideas of the RAR and RAD to achieve a balance between the accuracy

Table 1. A summary of component-wise studies of PINN architecture.

Components	Key studies
Selection of collocation points	Non-adaptive sampling [7-8] Adaptive sampling [9-13]
Neural networks	Fully connected neural networks [14-21] Shallow/sparse neural networks [22-24] Convolutional neural networks [25-29] Generative neural networks [30-33] Sequence models [34-36] Deep operator networks [37-40] Bayesian networks [41] Transformer networks [42] Graph neural networks [43, 44] Multi-output PINN [45]
Differentiation	Coupled-automatic-numerical differentiation [46]
Loss function	Collocation loss function [47-56] Variational loss function [57-61] L^p norm [62]
Neural network training	Particle swarm optimization [63] Non-dominated sorting genetic algorithm [64] Importance sampling [65, 66] Training without stacked backpropagation [67]

and computational cost. The performance of three adaptive sampling techniques, RAR, RAD, and RAR-D, were compared; it was revealed that RAD showed the best performance, and RAR-D showed comparable performance to RAD and yet improved computational efficiency. Peng et al. [12] proposed a residual-based adaptive node generation (RANG) technique that attempted to combine the advantages of quasi-uniform sampling and residual-based adaptive sampling. They additionally utilized a memory mechanism to enhance the performance, but only the evaluation in two-dimensional problems was provided; hence, further research is required for its implementation in higher-dimensional problems. Subramanian et al. [13] suggested an adaptive self-supervision algorithm that utilized the gradient values instead of the loss term residuals for adaptive sampling. For most non-convex problems, appropriate constraints should be incorporated into the neural network to avoid falling into local minima during optimization. The key idea of the algorithm is to periodically adjust the ratio of uniform sampling and adaptive sampling using the cosine annealing scheme during each specific number of epochs, while utilizing uniform sampling for the remaining epochs, to maintain a balance between local adaptability and domain coverage.

Depending on the physics problem to be solved, each or the combination of the suggested techniques can be appropriately applied and to select the collocation points.

4.2 Neural networks

Besides the fully connected neural networks (FCNNs) utilized in *Vanilla* PINN [2] in 2019, studies have proposed modifications to FCNNs or other types of neural network architectures to handle diverse types of input data and/or enhance the performance of PINN. This section reviews studies on modifications to FCNNs and utilizing other types of neural network architectures, including shallow/sparse neural networks, convolutional neural networks, generative neural networks, sequence models, deep operator network, bayesian neural networks, transformer networks, and multi-output physics informed neural network.

4.2.1 Fully connected neural networks

Fully connected neural networks (FCNNs) enable modeling of complex nonlinear relationships between inputs and outputs through multiple hidden layers with an appropriate selection of activation functions such as hyperbolic tangent (tanh), rectified linear unit (ReLU), sigmoid, softmax, etc. Selecting an appropriate activation function typically involves trial-and-error depending on the problem of concern. Jagtap et al. [14] attempted to resolve the issues with vanishing or exploding gradients by presenting an adaptive activation function, in which an adaptable hyperparameter and a scaling factor are added to the activation function. Based on Ref. [14], Jagtap et al. [15] introduced the concept of a locally adaptive activation function, which adds a slope recovery term to the activation function per layer and neuron. Gnanasambandam et al. [16] proposed a self-scalable hyperbolic tangent (Stan) activation function to resolve the scalability issue. Stan improved the convergence speed and generalization performance by adding a self-scaling term to the tanh activation function to compensate for the case where the orders of magnitude between input and output differ. Abbasi and Andersen [17] proposed physical activation functions (PAFs), which is a generic activation function derived from physical laws. PAFs are designed to incorporate known physical or mathematical laws directly into neural networks, replacing traditional activation functions such as tanh or sigmoid. By incorporating PAFs, the interpretability of neural networks and performance of the out-of-distribution prediction (extrapolation) were enhanced, especially for networks with smaller sizes.

Peng et al. [18] proposed prior dictionary based PINN (PD-PINN) technique in which prior information of the problem is incorporated into the additional dictionary fusion layer, which is later combined with the output layer of the neural network via inner products. Based on the spectral methods that utilize the orthogonal basis of the problem, theoretically established spectral convergence can be utilized, and the issue with truncation error is also resolved thanks to the universal approximation theory. The dictionary is a combination of word functions containing prior information and is non-intrusive to existing neural networks; it can be expressed in various types according to the problems such as spatial-based, frequency-based, and data-driven.

Residual neural network (ResNet) is a deep neural network that provides residual connections, proposed in Ref. [19]. ResNet includes identity mapping of inputs, and the output of one layer may skip one or more layers and be added to the output of a subsequent layer. ResNet is typically known to resolve the problem of vanishing gradients and can be beneficial in training FCNNs with many hidden layers. Cheng and Zhang [20] proposed Res-PINN, which combined PINN with Resnet to leverage such advantages of identity mapping. The performance of Res-PINN was evaluated by solving Burger's equation and Navier-Stokes equation, and it was shown to enhance the accuracy and stability.

As mentioned in Sec. 2.3, algebraic equations can be incorporated into the governing equations of PINN as additional physics knowledge. In addressing a system of equations that consist of both differential and algebraic equations, or differential-algebraic equations (DAEs), Moya and Lin [21] proposed DAE-PINN framework, which modified the neural network architecture of PINN to enhance its capability. DAE-PINN modified its FCNNs structure to incorporate implicit Runge-Kutta methods and a penalty-based method, which targeted on resolving the issues of stiffness and complex dynamics. As *Vanilla* PINN with conventional FCNNs struggles in resolving these issues, DAE-PINN broadened the applicability of PINN in such scenarios that may occur in more practical problems.

4.2.2 Shallow/sparse neural networks

To improve the training efficiency, techniques utilizing shallow neural networks were proposed. Dwivedi and Srinivasan [22] proposed the physics-informed extreme learning machine (PIELM) technique that incorporates the extreme learning machine (ELM) technique into PINN. By utilizing the ELM technique that only trains the weights of the output layer, the PIELM technique successfully reduced the training cost by decreasing the number of training parameters. However, the PIELM technique is only applicable to the linear PDEs. Schiassi et al. [23] proposed extreme theory of functional connections (X-TFC), which combines the theory of functional connections (TFC) with ELM. Resolving the shortcomings of the PIELM technique, X-TFC was applied to non-linear PDEs with formulating constrained expressions to automatically satisfy BC/ICs.

On the other hand, Ramabathiran and Ramachandran [24] proposed sparse, physics-based, and partially interpretable neural networks (SPINNs), which utilized sparse neural network (SNN) consisting of a mesh encoding layer and kernel layer to obtain solutions of ODEs and PDEs. Here, numerous meshless techniques could be carried out; as an example, the radial basis function (RBF) was implemented through SPINN. Additionally, to apply SPINN to solve time dependent PDEs, the Finite Difference method was incorporated into SPINN. As a result, the computational efficiency was improved, as the number of training parameters is much fewer than in FCNNs.

4.2.3 Convolutional neural networks

It is generally challenging to learn the large-scale spatio-temporal solution field efficiently utilizing FCNNs. To resolve this issue, studies have been conducted using convolutional neural networks (CNNs) to generate the solution field with fewer parameters. Physics-informed geometry-adaptive convolutional neural network (PhyGeoNet) by Gao et al. [25] utilized coordinate transformation techniques that can be applied to computational domains with irregular geometry. PhyGeoNet maps the solution fields in the irregular physical domain into a rectangular reference domain, applying the traditional CNNs along with the 'hard' enforcement of BCs as in Refs. [26, 27]. Performances of PhyGeoNet were evaluated using benchmark problems: a heat equation, a Navier-Stokes equation, and a Poisson equation, with CFD results. Compared to *Vanilla* PINN, PhyGeoNet required nearly 10 times fewer iterations to obtain the solution with less relative errors, but the comparison was available only for steady-state problems. Gong and Tang [28] proposed an energy-based physics-informed neural network (EPINN) for low-frequency electromagnetic computation. The magnetic energy norm error was integrated into the network as the loss function to allow EPINN to focus on regions of interest instead of computing the entire computation domain. EPINN accepted the Gaussian distributions from the results of applying the finite element method (FEM) as input for the encoder-decoder network in a shape of U-Net. The network was then trained with an energy-based loss function to enforce the physics of the problem to obtain the desired solution and showed outstanding performance in interpolation tasks. Zhao et al. [29] proposed a physics-informed convolutional neural network for the temperature field prediction of a heat source layout (HSL-TFP) without using labeled data. Here, the network learns to solve a family of heat conduction equations by mapping the intensity distribution function to the solution of PDE. For training of CNNs in a shape of U-Net, physics-informed loss function based on finite difference approximations of the governing PDE, and Dirichlet and Neumann boundary conditions were enforced in a 'hard' manner.

4.2.4 Generative neural networks

Some studies have attempted to incorporate generative adversarial networks (GANs), which can generate various synthetic data, into PINN. Yang and Perdikaris [30] incorporated the concept of adversarial learning into the PINN's training objective with probabilistic formulation to enhance the model's robustness and stability, where stochastic gradient descent (SGD) was utilized for optimization. Yang et al. [31] proposed physics-informed GANs (PI-GANs) to compute the solutions of stochastic differential equations (SDEs) based on limited and scattered measurements. PI-GANs successfully approximated Gaussian processes and solved elliptic SDEs involving three stochastic processes: the solution, the forcing, and the diffusion coefficient. Specifically, instead of *Vanilla* GANs, Wasserstein GANs with a gradient penalty (WGAN-GP) was applied due to its enhanced stability. Daw et al. [32] supplemented Ref. [30] and suggested physics-informed discriminator–GAN (PID-GAN), which incorporated physical knowledge not only to the generator but also to the discriminator, resolving the issue of gradient dynamics of existing techniques.

More accurate solutions were obtained for probabilistic PDEs or SDEs by utilizing GANs, but the computational cost was much larger than those of employing other neural networks. Therefore, Zhong and Meidani [33] presented physics-informed variational autoencoder (PI-VAE), which incorporates VAE, one of generative models, into PINN to improve the computational efficiency and accuracy. PI-VAE used maximum mean discrepancy (MMD), calculating the distance between probability distributions as the loss function and utilized the mini-batch gradient descent algorithm for training both the encoder and decoder simultaneously. However, PI-VAE is currently limited to low-dimensional problems, as the amount of the mini-batch required grows exponentially for high-dimensional problems.

4.2.5 Sequence models

Recurrent neural networks (RNNs) have been a preferred network architecture to handle dynamical systems, i.e., time-dependent problems, due to their ability to handle sequential data efficiently. This mechanism is known as sequence-to-sequence learning. In addition, to handle the cases with long time-steps or long-term dependencies, long short-term memory (LSTM) and gated recurrent unit (GRU) have been employed to resolve the issues of vanishing or exploding gradient. Viana et al. [34] presented an approach to consider ordinary differential equations (ODEs) as a form of a directed graph model, and utilized RNN with a hybrid formulation with both pure physics-informed and additional data-driven schemes, depending on data availability. Zhang et al. [35] proposed physics-informed multi-LSTM (PhyLSTM) networks to generate metamodels using LSTM with the earthquake dynamics formulated as reduced-fidelity nonlinear equations of motion. To integrate physical knowledge into LSTM, architectures using two or three deep LSTM networks were proposed, namely $PhyLSTM^2$ and $PhyLSTM^3$. $PhyLSTM^2$ contains one LSTM network for input-output relationship modeling and the other LSTM network for physics modeling, as well as a component to enforce boundary conditions. $PhyLSTM^3$ contains an additional LSTM network used for hysteretic parameter modeling in case of more complicated rate-dependent dynamic problems. Such networks are advantageous in terms of 1) clear interpretability of physical meaning of the problem, 2) generalizability performance, and 3) compensation for data scarcity issues. However, they contain limitations in terms of the computational speed and applicability to other dynamical systems. Ren et al. [36] proposed physics-informed convolutional-recurrent network (PhyCRNet), which applied convolutional LSTM (ConvLSTM) to address the scalability and generalization issues. PhyCRNet utilized ConvLSTM as a single cell structure as in LSTM and could be applied to the spatio-temporal PDE, serving as a universal model. Additionally, PhyCRNet-s was presented, which reduced the computational costs by skipping the encoder part in every specific number of iterations.

4.2.6 Deep operator networks

Lu et al. [37] proposed a novel neural network named the deep operator network (DeepONet) for improving generalization performances, leveraging the universal operator approximation theorem provided in Ref. [38]. DeepONet contains branch net and trunk net, which extract latent representations from the input function and input coordinates, respectively. Wang et al. [39] proposed a physics-informed DeepONet, which incorporated physical constraints into DeepONet. Thus, physics-informed DeepONet can be considered as a data-free method, as opposed to DeepONet, a data-driven method. PDEs and BC/ICs residuals were expressed as the weighted sum in the loss function and optimization of the model parameters was performed to train the neural network. As such, physics-informed DeepONet enhanced generalization performance and data efficiency without paired input-output training data information. In addition, Cai et al. [40] presented DeepM&Mnet in which a few DeepONet structures were utilized either in parallel or in series, for applications to solve multi-physics/multi-scale problems.

4.2.7 Bayesian neural networks

Yang et al. [41] proposed bayesian PINN (B-PINN), which utilizes bayesian neural networks (BNNs) and hamiltonian monte carlo (HMC) or variational inference (VI) to manage scattered noisy data and physical knowledge altogether for computation of PDE solutions and uncertainty quantification. Aleatory uncertainty generated from the noisy data was successfully quantified by calculating the mean and standard deviation of the solution, and the overfitting issue that occurs due to the noisy input data was resolved.

4.2.8 Transformer networks

Li et al. [42] proposed a gradient-optimized PINNs (GOPINNs), which automatically adjusts the coefficients of the penalty term during the model training to enhance the gradient equalization effect of each loss term. Additionally, GO-PINNs utilize two transformer networks with residual connections to update the hidden layers and augment the hidden state, which enhance the stability and prediction accuracy of PINN.

4.2.9 Graph neural networks

In order to accommodate graph-type input such as the knowledge graph mentioned in Sec. 2.3 that models a set of objects and their relationships, graph neural networks (GNNs) can be utilized. Similar to the idea of CNNs, GNNs can provide inference of the node from graphs data with the information obtained from neighboring nodes by aggregators.

Gao et al. [43] proposed physics-informed graph neural Galerkin networks, utilizing graph convolution network (GCN), one of the types of GNNs, to solve forward and inverse PDEs in a unified manner. Here, instead of learning continuous functions as in PINN with FCNNs, discrete learning improved PINN's scalability issue and enabled hard boundary condition enforcement. Additionally, flexibility of GCN that can accommodate unstructured inputs enhanced the applicability of PINN.

To further accommodate the graph data consisting of relational information of inputs, Liu and Pyrcz [44] proposed physics-informed graph neural network (PI-GNN) that incorporated the idea of GNNs into PINN, applied to enhance the production forecasting in hydrocarbon resource development. Here, a customized graph convolution layer was utilized to leverage the relational information, the adjacency matrix, between inputs to provide improved accuracy and interpretability.

4.2.10 Multi-output physics-informed neural network

For handling the target task of uncertainty quantification described in Sec. 2.4, instead of solving the SDEs described in Sec. 2.1, Yang and Foster [45] proposed multi-output physics-informed neural network (MO-PINN) with solutions and uncertainty distributions as outputs to manage sparse noisy data. The MO-PINN modified the structure of FCNN to impose prior knowledge of uncertainties on outputs. For each input, the model creates multiple discrete outputs that form a distribution for uncertainty quantification.

4.3 Differentiation

The differential operators computed by numerical differentiation (ND) and automatic differentiation (AD) are inherently different in nature and each has its own advantages and disadvantages. While ND approximates the derivatives from a local set of physics outputs based on a specific numerical scheme, AD can compute the exact derivatives at any point in a computational domain. Leveraging the exactness of AD, most studies on PINN utilized AD to compute the derivatives. Nevertheless, its use in computing the training loss for PINNs does not ensure accuracy of the model itself unless a sufficiently large set of collocation points is utilized. In many cases, this could potentially increase the computational cost, especially for high-dimensional problems. Chiu et al. [46] suggested a coupled-automatic-numerical PINN (can-PINN), which utilizes not only AD but also ND for the computation of the derivatives. By combining the robustness of ND to the number of collocation points and accuracy of AD, they demonstrated that the proposed can-PINN is indeed highly efficient and consistently provides more accurate solutions compared to using only AD or ND, as shown by the results of their experiments.

4.4 Loss function

As described in Sec. 3, the loss function for PINN is expressed as the weighted sum of the loss terms in Eqs. (8)-(12). Although the weight coefficients were not considered in *Vanilla* PINN, later studies proposed them to consider relative scales for each loss term or to impose different significances for network training. At first, Wight and Zhao [47] presented the concept of an adaptive PINN and suggested how to determine the weight coefficients according to the importance of each loss term. For the time-dependent problem, as it is more important to satisfy the initial condition, weight coefficient $C >> 1$ was multiplied to the loss term for the IC, which is shown as the following:

$$L(\boldsymbol{\theta}) = CL_0(\boldsymbol{\theta}) + L_b(\boldsymbol{\theta}) + L_f(\boldsymbol{\theta}). \tag{13}$$

Later, Eq. (13) was formulated more generally, leading to the weight coefficients multiplied to each of the loss terms as shown in Eq. (8). Based on such formulation, numerous approaches were proposed to compute the weight coefficients of the total loss function to enhance the accuracy and/or computational efficiency of PINN. Wang et al. [48] argued that since the C value of Eq. (13) significantly varies depending on the problem, it is more ideal to tune the value of C as training progresses. As such, learning rate annealing (LRA) was applied to the weight coefficients, namely LRA-PINN, which were treated as learning rate coefficients, where the values were updated at every training epoch.

Yu et al. [49] proposed a gradient-enhanced PINN (gPINN) that enforces the gradients of PDE residual to be 0 by including an additional loss term:

$$L(\boldsymbol{\theta}) = \omega_0 L_0 + \omega_b L_b + \omega_f L_f + \omega_d L_d + \sum_{i=1}^{n} \omega_{gi} L_{gi}, \tag{14}$$

$$L_{gi} = \frac{1}{N_{gi}} \sum_{x_i \in N_{gi}} \left| \frac{\partial f}{\partial x_i} \right|^2. \tag{15}$$

To avoid using the additional loss term as in Eq. (14), Xiong et al. [50] presented a gradient-weighted PINN (gwPINN), which further multiplied the gradient-related weight function that contains gradient information to each of the loss terms in Eq. (8). The weight functions are adaptively formulated based on the given differential equations, considering the derivatives of each loss term. Similar to the LRA-PINN in Ref. [48], Liu et al. [51] presented an adaptive weight PINN (AW-PINN), which adaptively updated the weight coefficients of loss function as network training proceeds. AW-PINN further improved the computational efficiency by reducing the number of hyper-parameters using logarithmic means in the weight update process. Wang et al. [52] proposed a technique that dynamically calculates the weight coefficients of the loss function using a neural tangent kernel (NTK), thereby balancing the convergence rates of each loss term, and theoretically studied the training dynamics of PINN. Liu and Wang [53] presented physics-constrained neural network with minimax (PCNN-MM) to find the saddle point with minimax formulation, considering gradient aspects of the weights in both the neural network and loss function concurrently. McClenny and Braga-Neto [54] proposed a self-adaptive PINN (SA-PINN), which conducts training in a similar manner, while calculating the weight coefficients for each training point to increase flexibility. As opposed to using multiple loss terms, Nasiri and Dargazany [55] suggested a reduced-PINN model that incorporates numerical integration to reduce the number of terms of the loss function for better computational efficiency for the system of ODEs. Maddu et al. [56] proposed an inverse dirichlet weighing technique that applies sequential training to all first-order optimizers, such as Adam, without any additional computational cost. Loss function weights were calculated based on the variances of each loss term's gradient to resolve the vanishing gradient issues, and the multi-scale problems were solved more accurately by balancing the gradient distributions.

The previously described loss functions are generally considered as a type of collocation loss function, as it enforces the residuals on each collocation point to be (nearly) zero. As another type of loss functions, a variational loss function is described using the test functions and variational formulation of the problem, provided by Kharazmi et al. [57].

$$(\mathbf{F}[u(\mathbf{x},t);\boldsymbol{\gamma}],\ v(\mathbf{x}))_\Omega = (f(\mathbf{x}),\ v(\mathbf{x}))_\Omega \tag{16}$$

where $(\cdot,\ \cdot)_\Omega$ denotes an inner product as well as integration over the domain Ω and $v(\mathbf{x})$ denotes a properly chosen test function in a discrete finite dimensional space $V_K = span$ $\{v_k, k = 1, 2, \ldots, K\}$. The variational loss function can be defined accordingly as

$$L^{\mathrm{var}} = L_f^{\mathrm{var}} + L_b, \text{ where} \tag{17}$$

$$L_f^{\mathrm{var}} = \frac{1}{K} \sum_{k=1}^{K} \left| R_k - F_k \right|^2, \tag{18}$$

$$R = (\mathbf{F}[u(\mathbf{x},t);\boldsymbol{\gamma}],\ v(\mathbf{x}))_\Omega,\ F = (f(\mathbf{x}),\ v(\mathbf{x}))_\Omega \tag{19}$$

Variational PINN (VPINN) incorporated such variational formulations of the problem into the loss function. As the integrand of the variational loss function is integrated by parts, the order of the differential operators can be lowered, which effectively lowers the training cost. In addition, utilizing the variational loss function is beneficial for the purpose of domain decomposition, as test functions can be used independently for each subdomain to capture local characteristics and provide more flexible learning approaches. Similar approach was provided by Khydayi-mehr and Zavlanos [58], where the loss function was also formulated as a variational form to solve PDEs, namely VarNet. VPINNs and VarNet both utilized Petrov-Galerkin trial functions; however, while VarNet used piecewise polynomials, VPINNs used global polynomials as test functions. Based on [57], Kharazmi et al. [59] further applied *hp*-refinement via domain decomposition and projection onto the space of high-order polynomials to VPINNS and proposed *hp*-variational PINNs (*hp*-VPINNs). E and Yu [60] proposed deep Ritz method, a combination of deep learning and the Ritz method, which numerically solves variational problems formulated from PDEs by training a deep neural network to learn an effective representation of the solution space. As such, utilizing the variational loss function offer certain advantages to PINN, such as order reduction of differential operators and better locality.

Furthermore, Bai et al. [61] proposed a modified least squares weighted residual (LSWR) loss function, which integrates the residuals in the computational domain as

$$L(\theta) = \chi_1 \int_{\Omega} R_f^2 d\Omega + \chi_2 \int_{\Gamma} R_b^2 d\Gamma, \tag{20}$$

where R_f, R_b denote the residuals of the PDEs and boundary conditions and χ_1, χ_2 denotes the two scaling factors to balance the scales of the two residuals. By applying the LSWR loss function to PINN, better generalization capability was achieved and effectively alleviated the scalability issue with the two scaling factors. In addition, Wang et al. [62] conducted a theoretical investigation using L^p norm, $(p \in (1, +\infty))$, for the type of the loss function, instead of MSE (L^2 norm). As a result, it was confirmed that utilizing the L^p norm is more appropriate in terms of the performance for high-dimensional problems.

4.5 Neural network training

The last part of PINN is to train the neural network via the feedback mechanism, which is essentially the minimization problem of the loss function to find the optimal neural network parameter set $\boldsymbol{\theta}^*$ as:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} (L(\boldsymbol{\theta})). \tag{21}$$

In *Vanilla* PINN, $\boldsymbol{\theta}^*$ that minimizes the loss function is computed using the Adam optimizer, which is a type of the stochastic gradient descent algorithms. Moreover, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) optimizer can be utilized to obtain more precise results by computing the Hessian matrix that determines the optimization direction but tends to converge quickly to the local minima. Additionally, the limited-memory Broyden-Fletcher-Goldfarb-Shanno with box constraints (L-BFGS-B) optimizer, which is the combination of L-BFGS, a limited-memory version of BFGS, and BFGS-B that is applied to the bound constrained optimization problem. Furthermore, Adam and L-BFGS-B were applied sequentially for fine-tuning purposes in Ref. [68].

Davi and Neto [63] proposed an approach that utilized particle swarm optimization (PSO), which is one of the population-based stochastic optimization techniques, for the neural network training of PINN, namely PSO-PINN. Each collocation point selected by a uniform or random sampling was treated as a single particle (candidate solution), and the final solution was obtained from moving the position of each particle by exchanging information with neighboring particles. Moreover, the ensemble of PINN solutions can be obtained with PSO-PINN, allowing for uncertainty quantification.

Lu et al. [64] proposed NSGA-PINN, a framework that combines the non-dominated sorting genetic algorithm II (NSGA-II) with PINN to perform multi-objective optimization-based training of neural networks. NSGA-PINN utilizes non-dominated sorting, crowding distance calculation, and crowded binary tournament selection to generate a diverse set of high-quality solutions that can effectively manage multiple objectives in the PINN training procedure. With the iterative integration of NSGA-II and the ADAM optimizer, NSGA-PINN resolved the issue of local minima, enhancing the optimization performance of PINN.

Nabian et al. [65] applied the concept of importance sampling for more efficient training of the network. Instead of calculating the L^2 norm of the loss gradient, the loss values of subsets of the collocation points (seeds) were calculated with piecewise constant approximation to reduce the number of backpropagations. Yang et al. [66] proposed dynamic mesh-based importance sampling (DMIS) to accelerate the convergence without significantly increasing the computational cost. To reduce the computational cost for calculating the sampling probability of each IS point, they proposed a novel sampling weight estimation method, called dynamic mesh-based weight estimation (DMWE), which constructed a dynamic triangular mesh to estimate the weight of every data point efficiently. The triangular mesh constructed by DMWE was updated dynamically according to the loss distribution of the whole domain during the training process.

He et al. [67] presented an approach to train the network without stacked backpropagation to resolve the scalability issue of PINN. Here, the PDE solution was represented with a Gaussian smoothed model, while the dependency was computed based on Stein's identity to significantly reduce the computational cost by eliminating stacked backpropagations of AD. Furthermore, since only forward-pass computation was required for the loss calculation, the computation time could be further reduced through parallelization.

5. Studies on key topics of PINN

In Sec. 4, studies that attempted to improve the performance of PINN by focusing on the component-wise PINN architecture depicted in Fig. 1 were described. In this section, studies on some key topics of PINN are described to provide a guideline for implementing PINN in various engineering fields of interest. Topics covered in this section are: 1) domain decomposition, 2) temporal causality, 3) meta-learning, 4) reduced-order modeling, 5) model ensembles, 6) extrapolation, and 7) multiple instances learning.

5.1 Domain decomposition

As the number of dimensions of a problem increases, the amount of the collocation points required for accurate training grow exponentially. Such phenomenon is referred to as the curse of dimensionality, which is considered as one of the common challenges in computational science and machine learning. Although PINN can mitigate some aspects of the curse of dimensionality, owing to its mesh-free nature, achieving a reasonable level of accuracy in complex and high-dimensional problems with reduced computational cost remains a challenge.

To address this issue, various studies have been proposed that decompose the computational domain into sub-domains and train a reduced number of neural network parameters for each sub-domain in parallel. Such approach, as known as domain decomposition, not only improves the computational efficiency in training PINN, but also enhances the ability of the trained model to represent complex behaviors of the problem, such as discontinuous solutions, more accurately. Jagtap et al. [69] presented a conservative PINN (cPINN) that connects each sub-domain using interface conditions. Flux continuity and average solution were used as interface conditions, and a distinct PINN architecture was applied to each sub-domain to increase the degrees of freedom and solve problems with discontinuous solutions. Meng et al. [70] proposed a parareal PINN (PPINN), a parallel technique with the decomposition of the temporal domain to improve computational efficiency, which performed correction of each sub-domain through PINN using the results from fast coarse-grained (CG) solvers. PPINN significantly improved PINN training speed by using the results of the fast coarse-grained solver as the initial condition of each sub-domain. Based on cPINN, Jagtap and Karniadakis [71] presented an extended PINN (XPINN), a general framework that applies a spatio-temporal domain decomposition to PINN. XPINN is a more flexible technique in which a separate PINN is applied to each sub-domain, similar to the cPINN approach. However, instead of using the flux continuity condition that increases the training complexity of the interface condition, only the average solution and residual values at the interface were compared. Stiller et al. [72] presented GatedPINN, which utilizes conditional computing and an adaptive domain decomposition. Conditional computing is a technique of activating certain units within the neural network according to the input inside each spatial domain, with the activation condition determined from the gating network. In addition, the gating network determined which neural network structure to be used for a specific input, enabling the simultaneous adaptive domain decomposition. Thereafter, Hu et al. [73] presented an augmented PINN (APINN) that complemented XPINN. APINN enhanced the generalization performance by fine-tuning of domain decomposition and parameter sharing between sub-PINN using a trainable gate network.

5.2 Temporal causality

Time-dependent problems, prevalent in the engineering and scientific domains, inherently rely on the principle of causality, the concept that the current and future states depend on past states. As such, in order to solve time-dependent problems accurately with PINN, it is crucial to enforce the temporal causality in the training of PINN to ensure the physical plausibility of the solution. Without considering the temporal causality, information may not propagate appropriately through time within the computational domain and cause training challenges, leading to non-physical solutions and model inconsistencies.

Studies have been proposed that incorporate the laws of causality for PINN to better respect the inherent time-dependent nature of the problem. Wang et al. [74] proposed causal PINN, which induces minimization of the residual values of the previous time segments by adjusting the temporal weight steepness of each time segment, accomplished by the temporal domain decomposition and inclusion of a causality parameter. Daw et al. [75] proposed the evolutionary sampling (Evo) and causal evolutionary sampling (causal Evo), based on the evolutionary algorithm proposed by Eiben et al. [76]. Evo is the technique in which points with high residual are extracted for each iteration and re-sampled within the points in uniform distribution to compensate for the propagation failure of PINN. Causal Evo, on the other hand, additionally considered the temporal causality during the sampling procedure. Compared to adaptive sampling techniques discussed in Sec. 4.1, both Evo and causal Evo enhanced the performance of PINN with fewer collocation points. Guo et al. [77] proposed an adaptive causal sampling method (ACSM) that considers temporal causality. The ACSM utilizes a distribution ratio that adaptively selects collocation points across spatio-temporal subdomains, thereby balancing both the magnitudes of the loss terms and the temporal causality. Utilizing ACSM increased the accuracy up to two orders of magnitude and improved computational efficiency, with the same number of collocation points when compared to causal PINN. ACSM was applied to Cahn Hilliard and KdV equations, which contain high-order derivatives and strong nonlinearity, and was shown to improve the accuracy and computational efficiency with fewer collocation points, demonstrating its applicability to high-dimensional problems. Similar to the sampling techniques considering the causality in Refs. [75, 77], Mattey and Ghosh [78] proposed backward compatible PINN (bc-PINN) that re-trains the same network to compute the solution that satisfies the solutions of the previous time segments through transfer learning. Here, the previous predictions were considered as data terms to train the corresponding future predictions.

Penwarden et al. [79] summarized previous causality enforcement techniques applied in PINN, and proposed a causality-enforcing framework that contained two techniques: stacked-decomposition and window-sweeping. The stacked-decomposition technique incorporated the time-marching and XPINN methods; for the decomposed temporal domain of n segments or subdomains, sequential training of dS number of segments were proceeded. With dS = 1, each segment or subdomain was trained sequentially (time-marching), and with $dS = n$, all segments were parallelly trained, which was equivalent to XPINN. Next, the window-sweeping technique involved moving a soft-causality window through time and applying a weight mask on each collocation point in various ways, inspired by Ref. [74].

5.3 Meta-learning

Meta-learning is a cross-disciplinary field of research that encompasses multi-task learning and transfer learning. By incorporating meta-learning – the principles of "learning to learn" – into the training of PINN, prior challenges such as static loss function weights, slow adaptation to new problems, and limited generalization capabilities can be addressed without extensive retraining. This approach can significantly broaden the applicability of PINN in modeling complex, time-dependent, or high-dimensional physical problems.

Psaros et al. [80] presented a technique for constructing a loss function expression with a bi-level minimization problem using meta-learning. In the inner optimization, the training of the PINN follows the same procedure, whereas the weights of the loss function are calculated in the outer optimization and then updated in the inner optimization to minimize the total loss. Goswami et al. [81] utilized the concept of transfer learning to enhance the computational efficiency. Here, the network weights and biases were calculated with Xavier initialization in the first iteration, and the parameters, except those of the output layer, were fixed with the values from the previous iteration so that only the parameters of the output layer were further updated. Bahmani and Sun [82] treated the training of PINN as a multi-objective/multi-task problem, and utilized transfer learning. Each loss term was considered as an objective, and a gradient surgery technique was proposed that compensated for the conflicts in gradients of multiple objectives. Afterwards, the transfer learning technique was applied by utilizing the computationally cheap solvers in advance (pre-training step) and adding the resultant values to the total loss function as an auxiliary label. Desai et al. [83] proposed one-shot transfer learning technique as a general framework for applying transfer learning to PINN. This technique initially learns the rich latent space of a specific family of differential equations through batch training of PINN, then solves the same family of differential equations with a one-shot inference via transfer learning, thus significantly reducing the computational time when the coefficients of BC/ICs and forcing functions change. Xu et al. [84] proposed transfer learning based boundary-condition-learnable PINN that was applied to solve more practical inverse problems in structural analysis. As a multi-task learning method for loss weights, Bayesian modeling was applied for computing relative confidences between tasks, together with the maximum likelihood estimation (MLE). Unknown loads were added as BCs in the loss function, and transfer learning was applied for quicker computation. Penwarden et al. [85] presented a detailed study on the incorporation of the meta-learning technique into PINN. From the PINN's perspective, meta-learning can be considered as a model-agnostic meta-learning (MAML), which finds the optimal parameters of the neural network and loss functions while training proceeds. For linear mapping of the task parameter set and weight vector containing all weights and biases, prediction models such as Gaussian process (GP) model and radial basis function (RBF) were utilized, followed by the network training, resulting in a bi-level optimization problem.

5.4 Reduced-order modeling

Reduced-order modeling (ROM) is a renowned technique that simplifies the analysis of complex systems by focusing on their dominant features or dynamics, thus reducing the overall complexity. Such an approach can be beneficial when incorporated into PINN, in order to tackle high-dimensional problems with complex dynamics with reduced computational costs. As such, studies have been proposed to incorporate the idea of ROM into PINN, so that the training process of PINN can be efficiently executed with aids of the reduced basis or dominant modes derived from ROM.

Chen et al. [86] conducted a study that integrated the concept of reduced-order modeling (ROM) into PINN and proposed a new strategy called physics-reinforced neural network (PRNN). PRNN utilized both PINN trained with the reduced-order equation and the projection data from the existing high-fidelity snapshots onto the reduced space to improve the prediction accuracy. Numerical results showed that the PRNN could predict reliable reduced-order solutions with higher accuracy when compared to the PINN or a purely data-driven neural network.

5.5 Model ensembles

Model ensembles refer to the strategy of utilizing multiple networks or models to achieve more robust predictions. In the context of PINN, studies have been conducted where multiple PINN models are utilized; each model in the ensemble is trained with different initializations or hyperparameters, enhancing the overall robustness of the prediction.

Haitsikevich and Ilin [87] proposed a study to improve on the network training through model ensembles. Here, PDE residuals are calculated with an ensemble of PINNs that were initialized with different weights while utilizing the same loss function. A collocation point where all ensembles match (ensemble agreement) with an error smaller than a prescribed threshold value is added as "pseudo-label". Starting with initial points, PINN ensembles gradually explore the computational domain to find a collocation point that meets the conditions of the ensemble arrangement and uses the added "pseudo-label" for loss function calculation. This process is repeated for all collocation points to conduct loss function minimization, and although the computational cost was shown to be larger than that of using a single PINN, a more robust and accurate solution was derived.

5.6 Extrapolation

Improving PINN's capabilities for extrapolation (i.e., predicting points outside the computational domain) presents a significant challenge. In general, PINN is trained within the specified computational domain, which in some cases can limit its applicability in scenarios that extend beyond these bounds. To broaden the applicability of PINN, especially in scenarios where data collection is challenging, it is crucial to enhance its extrapolation capability.

Kim et al. [88] introduced the dynamic pulling method (DPM) to improve PINN's extrapolation performance by applying a novel loss function and neural network training techniques. Linka et al. [89] discussed the pros and cons of a neural network family (ANN, PINN, SA-PINN) and bayesian inference (BI) family (BI, BNN, BPINN), respectively, in terms of the extrapolation performance. As per their study, SA-PINN exhibited the best extrapolation performance, but its training procedure was not robust and stable due to the complicated loss function. On the other hand, BPINN provided credible intervals for the solution with adequate extrapolation performance but required precise scaling and large amount of training data.

5.7 Multiple instances learning

One of the significant issues with implementing PINN is its limited ability to train a model for a single instance of BC/ICs or governing equations. While transfer learning can be applied after the training of one instance to reduce computational costs of training another instance, the unsupervised handling of multiple instances can potentially enhance the applicability of PINN in various practical problems.

Based on deep galerkin method (DGM) by Sirignano and Spiliopoulos [90] that utilized the mesh-free galerkin methods to train neural networks, Chudomelka et al. [91] proposed a legendre-galerkin deep neural network (LGNet) that utilized the Legendre polynomial as the basis function of the Galerkin method. The solution was reconstructed from the weighted sum of the basis function multiplied by coefficients α calculated via neural networks. Here, the neural network was constructed with the combination of CNN and FCNN. Based on LGNet, Choi et al. [92] presented the unsupervised legendre-galerkin neural network (ULGNet), where multiple instances of PDE can be trained in an unsupervised manner using forcing functions as the input. Here, the basis function ϕ in an appropriate Hilbert space H spanned by $\{\phi_k\}_{k=0}^{N-2}$, where N is a finite integer, is defined as a combination of Legendre polynomials as

$$\phi_k(\mathbf{x}) = L_k(\mathbf{x}) + a_k L_{k+1}(\mathbf{x}) + b_k L_{k+2}(\mathbf{x}), \tag{22}$$

where L_k denotes Legendre polynomials of degree k, and a_k, b_k can be chosen to represent various boundary conditions. Finally, the solution $u(\mathbf{x})$ is approximated as

$$u(\mathbf{x}) \approx \hat{u}(\mathbf{x}) = \sum_{k=0}^{N-2} \hat{\alpha}_k \phi_k(\mathbf{x}). \tag{23}$$

Here, the predicted solution $\hat{u}(\mathbf{x})$ is obtained by multiplying the coefficients $\{\hat{\alpha}_k\}_{k=0}^{N-2}$ and basis functions $\{\phi_k\}_{k=0}^{N-2}$, where $\{\hat{\alpha}_k\}_{k=0}^{N-2}$ denotes the outputs of the neural network from the input forcing functions. Subsequently, the training of ULGNet is conducted by minimizing the weak (variational) loss function to enforce the governing equations, similar to that of PINN.

For the optimization algorithm, L-BFGS was utilized and the parameters in ULGNet are updated accordingly. As such, using ULGNet, different instances of forcing functions and/or IC/BCs can be solved in an unsupervised manner for the governing equations of the same type, allowing for learning multiple instances that was infeasible with PINN.

6. PINN applications in engineering: an overview

Most of the studies regarding PINN involve solving typical examples such as Poisson equations, Allen-Cahn equations, advection-diffusion equations, Navier-Stokes equations, Schrödinger equations, and etc. While solving these equations is indeed a crucial task for demonstrating the performance enhancement of the proposed method for PINN compared to the existing methods, it may often be insufficient for implementing PINN in practical problems. In addition, as there exists no universal and fully optimized PINN framework at present, studies that attempted to apply PINN in engineering problems may not have implemented some of the novel techniques. This section provides three (practical applications) examples of PINN applied to solve some practical engineering problems and shows how some of the techniques discussed in Secs. 4 and 5 could be implemented to improve the performance of PINN. Secs. 6.1, 6.2, and 6.3 deal with the studies of PINN applications in fluid mechanics, heat transfer, and bearing fatigue prognosis, respectively.

6.1 Fluid mechanics application

As an example of PINN applications in Fluid Mechanics, the modeling of multi-phase flow and transport phenomena by Hanna et al. [10] is to be discussed, which is applied in practice in oil reservoirs, water resources management, and composite processing. In multi-phase flow, high-fidelity data are necessary to apply grid-based classical methods for adequate accuracy due to the moving flow-front, large discontinuity, and shock. Hence, a more flexible and meshless framework has been proposed by implementing PINN.

First of all, multiple governing equations and BC/ICs were defined with respect to the two-phase flow and transport phenomena. The governing equations consist of Darcy's law, mass conservation (incompressible flow), and advection equation (Eqs. (24)-(26)), with the fraction function c defined to separate the two fluid phases: $c = 0$ for one fluid phase and $c = 1$ for the other. Accordingly, the viscosity μ was defined as Eq. (27) using c.

$$\mathbf{v} = -\frac{1}{\mu}\mathbf{K}\cdot\nabla p, \tag{24}$$

$$\nabla\cdot\mathbf{v} = 0, \tag{25}$$

$$c_t + \mathbf{v}\cdot\nabla c = 0, \tag{26}$$

$$\mu = c\mu_2 + (1-c)\mu_1, \tag{27}$$

where $\mathbf{v}$ denotes the volume average Darcy's velocity, $\mathbf{K}$ the permeability tensor, and ∇p the pressure gradient. Next, BCs for $\mathbf{v}, p, c$ and ICs for c are as follows:

$$c(\mathbf{x}, t=0) = c_0(\mathbf{x}), \tag{28}$$

$$p(\mathbf{x}_{inlet}, t) = p_{in}, \tag{29}$$

$$p(\mathbf{x}_{outlet}, t) = p_{out}, \tag{30}$$

$$\mathbf{v}\cdot\mathbf{n} = 0 \quad \text{(Impermeable wall)}, \tag{31}$$

$$c(\mathbf{x}_{inlet}, t) = 1. \tag{32}$$

Accordingly, the loss function was composed of six loss terms (3 for the governing equations and 3 for the BC/ICs):

$$\begin{aligned} Loss = {} & \lambda_v loss_v + \lambda_c loss_c + \lambda_p loss_p + \\ & \lambda_1 loss_{f_1} + \lambda_2 loss_{f_2} + \lambda_3 loss_{f_3}, \end{aligned} \tag{33}$$

where MSE formulation was used with each weight coefficient set simply as 1.

Secondly, collocation points were selected within the computational domain using three different sampling techniques: uniform sampling, RAR, and RAD as described in Sec. 4.1. A total of 2500 collocation points (50×50 grid) were computed with uniform sampling, while for RAR and RAD 1600 uniformly sampled collocation points (40×40 grid) trained with Adam optimizer as well as additional point enrichment every 50

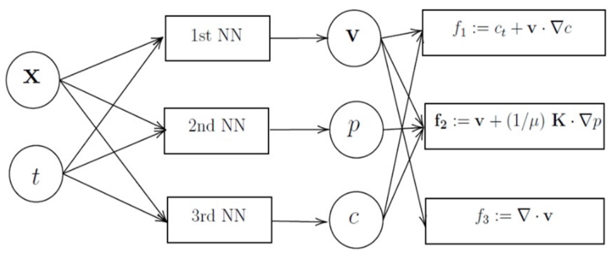
X
t
1st NN
2nd NN
3rd NN
v
p
c
$f_1 := c_t + \mathbf{v}\cdot\nabla c$
$\mathbf{f}_2 := \mathbf{v} + (1/\mu)\,\mathbf{K}\cdot\nabla p$
$f_3 := \nabla\cdot\mathbf{v}$

Fig. 2. Schematic diagram of the PINN for two-phase flow modeling [10].

BFGS iterations up to 2500 collocation points were computed. Furthermore, 1000 randomly sampled collocation points were used as a test set to compare the generalization performance.

Thirdly, FCNN was utilized as the neural network structure, the overall composition of which is shown in Fig. 2. For each output, three separate FCNNs were used, where each FCNN is comprised of five hidden layers with 20 neurons in each layer. Hyperbolic tangent (tanh) activation function was used in all hidden layers, while sigmoid activation function was used for the output layer of the pressure and fraction function, and linear activation function was used for the output layer of the velocity.

Lastly, the Adam optimizer was used for the network training. As a result, with the analytical solution as the reference, RAD provided the best training and validation accuracy in terms of the flow front location.

This study dealt with the PINN application for modeling the two-phase flow and transport phenomena, and successfully improved the solution accuracy with the proposed adaptive sampling technique (RAD) of the collocation points. With the main focus on applying adaptive sampling scheme and using multiple FCNNs, a 'customized' PINN based on the architecture presented in Fig. 1 was provided; such procedure can be benchmarked for other application fields of interest. Furthermore, studies provided in Secs. 4 and 5 regarding other ways to improve the performance of PINN could be potentially implemented and tested.

6.2 Heat transfer application

This section describes the study by Amini Niaki et al. [93], which modeled the thermochemical curing process of the composite-tool system within the autoclave using PINN. Considering the heat conduction from the exothermic composite cure process and to the autoclave, along with the temperature dependence of the curing process itself, is a rather complicated problem; utilizing *Vanilla* PINN is not feasible due to the discontinuity of the composite-tool interface. To resolve this issue the *Vanilla* PINN was modified using an adaptive loss weight technique, along with the transfer learning technique to further enhance the computational efficiency.

The exothermic heat transfer of solids (i.e., heat transfer with internal heat generation) can be represented with the following PDE:

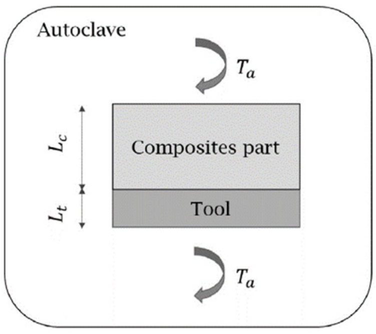

Fig. 3. Schematic diagram of composite-tool system in an autoclave [93].

$$\frac{\partial}{\partial t}(\rho C_p T) = \frac{\partial}{\partial x}(k_{xx}\frac{\partial T}{\partial x}) + \frac{\partial}{\partial y}(k_{yy}\frac{\partial T}{\partial y}) + \frac{\partial}{\partial z}(k_{zz}\frac{\partial T}{\partial z}) + \dot{Q}, \tag{34}$$

where T, C_p, k, ρ denote temperature, specific heat capacity, conductivity, and density of the solid, respectively, and $\dot{Q}$ denotes the internal heat generation rate. Here, $\dot{Q}$ is affected by the resin degree of cure α, represented as:

$$\dot{Q} = v_r \rho_r H_r \frac{d\alpha}{dt}. \tag{35}$$

Assuming the solid of the curing process was homogenous and physical properties were not affected by temperature, Eqs. (34) and (35) can be simplified as Eqs. (36) and (37). These equations were used as the governing equations of PINN.

$$\frac{\partial T}{\partial t} = a\frac{\partial^2 T}{\partial x^2} + b\frac{d\alpha}{dt}, \text{ where } a = \frac{k}{\rho C_p}\ , b = \frac{v_r \rho_r H_r}{\rho C_p}. \tag{36}$$

$$\frac{d\alpha}{dt} = g(\alpha, T). \tag{37}$$

Therefore, the problem can be formulated as a coupled system of differential equations for T and α in the spatio-temporal domain, and the interface condition of the composite-tool is described as in Eqs. (38) and (39) and Fig. 3. The temporal change of the autoclave air temperature was given by Fig. 4.

$$a = \begin{cases} a_t & \text{for} \quad 0 < x < L_t \\ a_c & \text{for} \quad L_t < x < L_t + L_c \end{cases}, \tag{38}$$

$$b = \begin{cases} 0 & \text{for} \quad 0 < x < L_t \\ b_c & \text{for} \quad L_t < x < L_t + L_c \end{cases}, \tag{39}$$

In addition, the prescribed BC, convective BC, and IC were defined as follows:

$$T|_{x=0} = T_a(t), \tag{40}$$

$$T|_{x=L_t+L_c} = T_a(a), \tag{41}$$

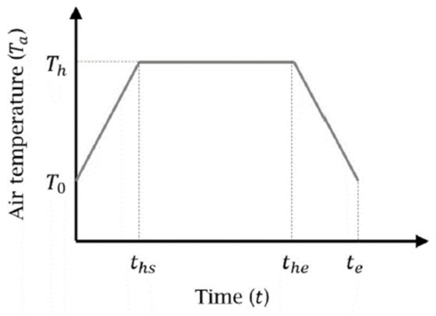

Fig. 4. Autoclave air temperature vs. processing time [93].

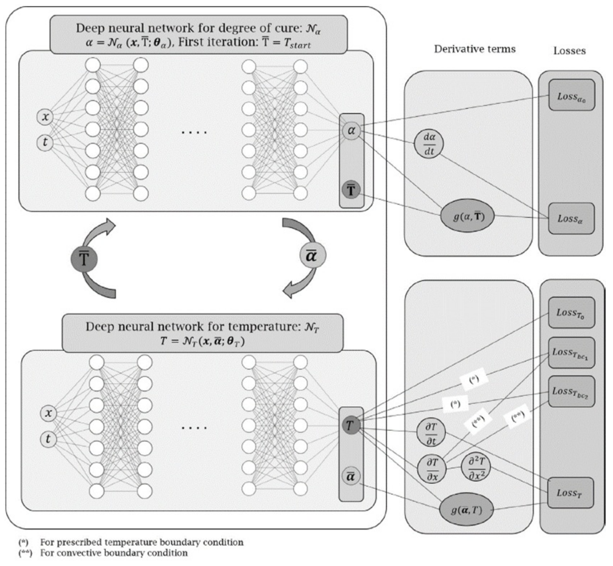
Deep neural network for degree of cure: $\mathcal{N}_\alpha$
$\alpha = \mathcal{N}_\alpha(\boldsymbol{x}, \bar{T}; \boldsymbol{\theta}_\alpha)$, First iteration: $\bar{T} = T_{start}$
Derivative terms
Losses
x
t
....
α
$\bar{T}$
$\frac{d\alpha}{dt}$
$g(\alpha, \bar{T})$
$Loss_{\alpha_0}$
$Loss_\alpha$
$\bar{T}$
$\bar{\alpha}$
Deep neural network for temperature: $\mathcal{N}_T$
$T = \mathcal{N}_T(\boldsymbol{x}, \bar{\alpha}; \boldsymbol{\theta}_T)$
x
t
....
T
$\bar{\alpha}$
$Loss_{T_0}$
$Loss_{T_{BC_1}}$
$Loss_{T_{BC_2}}$
$Loss_T$
(*)
(**)
$\frac{\partial T}{\partial t}$
$\frac{\partial T}{\partial x}$
$\frac{\partial^2 T}{\partial x^2}$
$g(\alpha, T)$
(*) For prescribed temperature boundary condition
(**) For convective boundary condition

Fig. 5. Schematic diagram of a PINN architecture with sequential training first on α, then on T [93].

$$h_t(T|_{x=0} - T_a(t)) = k_t \frac{\partial T}{\partial x}|_{x=0}, \tag{42}$$

$$h_c(T_a(t) - T|_{x=L_t+L_c}) = k_c \frac{\partial T}{\partial x}|_{x=L_t+L_c}, \tag{43}$$

$$T|_{t=0} = T_0(x), \tag{44}$$

$$\alpha|_{t=0} = \alpha_0(x). \tag{45}$$

Based on the problem formulation above, the performance of PINN was evaluated on four cases of different types of BCs and composite thicknesses. A uniform sampling technique was used to select the collocation points, with 500 points in $0 < x < L_t + L_c$, 1000 points in $0 < t < t_e$, 10000 points for IC, and 5000 points at each of the boundaries.

Next, considering that T and α are affected as training proceeds, two separate PINNs were constructed as shown in Fig. 5. After the training of the PINN for α, the other PINN for T was trained sequentially with the loss function minimization. In addition, two FCNNs $(\mathcal{N}_T^-, \mathcal{N}_T^+)$ were constructed inside the PINN structure for T, to treat the presence of discontinuity in the composite-tool interface. The authors utilized the FCNN for α composed of 7 hidden layers with 30 nodes per layer, and for T, 7 hidden layers with 20 nodes per layer. In addition, the loss function consists of the PDE residual terms and BC/IC terms for T and α as

$$L_{\mathcal{N}_\alpha} = L_\alpha + \omega_{\alpha_0} L_{\alpha_0}, \tag{46}$$

$$L_{\mathcal{N}_T} = L_T + \omega_{T_0} L_{T_0} + \omega_{T_{bc_1}} L_{T_{bc_1}} + \omega_{T_{bc_2}} L_{T_{bc_2}}. \tag{47}$$

Here, the adaptive loss weight algorithm in Ref. [48] described in Sec. 4.4 was used to normalize the gradients and resolve the issue with unbalanced gradients. For the activation function, the hyperbolic tangent (tanh) function was used for all hidden layers, and the softplus function and sigmoid function were used for the output layer of the neural network for T and α, respectively.

To train the network, the Adam optimizer with mini-batch optimization with a batch size of 512 was utilized, along with the learning rate scheduler that began with the learning rate of 10^{-3} and reduced it by half in the absence of improvement in the solution. Additionally, once the training was completed for one of the case studies, the network parameters were used for the initialization of the other case study; i.e., transfer learning was applied to enhance the convergence speed.

This study provides a demonstration of modifying the neural network structure to apply PINN appropriately to practical problems, utilizing both the adaptive loss weight algorithm and transfer learning to build a 'customized' PINN model.

6.3 Bearing fatigue prognosis

Since the main bearing inside the wind turbine is significantly affected by the condition of the grease used as a lubricant, developing a predictive model of the bearing fatigue is a highly complex task requiring accurate high-fidelity data. In addition, formulation of the governing equations for the grease is challenging due to the large variations and uncertainties (even within the same type) of the grease, which is influenced by the surrounding environment and the turbine's operating conditions.

To address these issues, Yucesan and Viana [34] proposed a hybrid PINN, incorporating a reduced-order physics submodel in conjunction with neural networks (as described in Sec. 4.2.5) to model the bearing fatigue and grease degradation, respectively.

The bearing fatigue was calculated using the standardized bearing life formula in ODE form as follows:

$$\frac{da_{BRG}}{dt} = \frac{1}{c_1 c_2(t)} \left(\frac{P(t)}{C} \right)^{\frac{10}{3}}, \tag{48}$$

where *P*(*t*) represents the equivalent dynamic bearing load, *C* the design load rating, c_1 a reliability level factor, and c_2 an adjustment factor based on grease properties.

Here, c_2 accounts for the state of grease by degradation over time, which is represented as a function of viscosity and contamination ratio as:

$$c_2(t) = f(\nu_t(t), \eta_c(t)), \tag{49}$$

where

$$\nu_t(t) = a_{GRS}(t)(\nu_{\deg} - \nu_{prs}) + \nu_{prs}, \tag{50}$$

$$\eta_c(t) = a_{GRS}(t)(\eta_{c\deg} - \eta_{cprs}) + \eta_{cprs}. \tag{51}$$

Next, the grease damage was calculated based on the grease service life from a median of the uncertainty distribution, as a function of bearing temperature as:

$$L_{50,GRS}(t) = f(T_{BRG}(\mathrm{t})). \tag{52}$$

Then, from Palmgren-Miner's rule, an incremental damage is calculated by adopting a quadratic relation between the life and damage for grease:

$$\Delta a_{50,GRS}(t) = \left(\frac{1}{L_{50,GRS}(t)} \right)^2. \tag{53}$$

With the incremental damage, the cumulative grease damage can be expressed as follows:

$$a_{50,GRS}(T) = \sum_{t=0}^{T} \Delta a_{50,GRS}(t). \tag{54}$$

For arbitrary k quantile of the distribution, Eqs. (52)-(54) can be expressed using the quantile ratio C_k that shifts the quantile curve as follows:

$$a_{k,GRS}(T) = C_k a_{50,GRS}(T). \tag{55}$$

To predict the bearing fatigue and grease degradation, as shown in Fig. 6, LSTM network was used for modeling time-series data, with wind speed and bearing temperature taken as inputs. As training data, time-series supervisory control and data acquisition (SCADA) data were used for the grease dam-

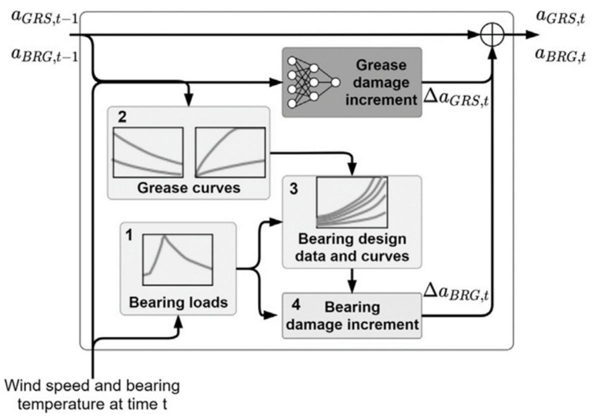

Fig. 6. Hybrid PINN using LSTM [34].

age calculation, where grease samples from 100 wind turbines were measured for six months. The continuous time domain was discretized based on 10-minute averages of wind speed and bearing temperature values. The FCNN used within the LSTM consists of five layers with 40, 20, 10, 5, and 1 number of neurons for each of the layers. The activation function used for the hidden layers was the exponential linear unit (elu), while that used for the input and output layers was the sigmoid function.

Notably, the loss function type used was mean absolute error (MAE) instead of the commonly used MSE. Training was conducted with a learning rate of 0.0005 for a total of 200 epochs. For the performance evaluation of the model, the results of four types of models that were trained with 10, 20, 50 and 100 turbines, respectively, were validated on another 100 turbines. The hybrid PINN model effectively estimated grease and bearing damage, even with noisy data. It precisely aligned predictions with actual outcomes and optimized the maintenance schedule for individual turbines, enhancing their expected lifespan to approximately 20 years.

Unlike the two examples described in Secs. 6.1 and 6.2, the physics-data hybrid-PINN architecture was proposed by combining LSTM and FCNN structures. It was shown that the greater the amount of experimental data used, the more accurate the prediction performance was obtained. However, only the wind turbine data at one specific location was used for the training, and extreme conditions or complex loading situations were not considered, leading to insufficient diversity of the experimental environment. Nevertheless, the current example serves as a good example of the application of PINN in for the practical problem where some noisy data are present.

7. Summary and discussion

This paper categorized the types of physics problem types and target tasks for which PINN can be utilized and reviewed numerous studies since the publication of the *Vanilla* PINN in 2019 that aimed to enhance the performance of PINN. To review the studies in an organized way, we first reviewed studies focusing on modifying each component of a generic PINN architecture: 1) selection of collocation points, 2) neural networks, 3) differentiation, 4) loss function and 5) neural network training. Thereafter, studies on some key topics in PINN were reviewed, regarding implementation of 1) domain decomposition, 2) temporal causality, 3) meta-learning, 4) reduced-order modeling, 5) model ensembles, 6) extrapolation and 7) multiple instances learning. Moreover, three exemplary applications of PINN on practical engineering problems were provided to demonstrate the effect and describe the ways of implementing some of the studies covered in Secs. 4 and 5.

As previously mentioned, many studies of PINN have been proposed, presenting new techniques to improve the accuracy or computational speed, combining existing numerical methods, and integrating some of the recent machine learning techniques or optimization techniques. However, due to diversities in conditions such as application fields, types of differential equations, neural network structures, and optimization techniques, direct comparisons between the latest studies of PINN are not feasible. Additionally, it still poses as an obstacle for most engineers who lack the respective domain knowledge to fully understand and select from the vast number of techniques. Cuomo et al. [6] provided an in-depth review of PINN and covered many of the theoretical aspects of PINN in detail, the current review paper attempts to provide a guideline for implementing PINN improvements in various engineering fields.

Building on the insights of the review, several key areas for future research are identified, which may be beneficial to implement the previously discussed studies of PINN better. Some of the potential research areas in the authors' opinion are as follows:

1) Meta-learning may be considered from a more macroscopic perspective, including various techniques of the future. Taking the idea from Ref. [85] where loss weight coefficients and neural network parameters were computed altogether with bi-level optimization, among the studies discussed in Secs. 4 and 5, optimal techniques for the customized PINN may as well be selected in an unsupervised manner. As such, a general formulation of PINN for better accessibility and its more optimal implementation could be established, which can potentially enhance the overall performance. Furthermore, a comparative study may be possible to evaluate various techniques that enhanced the performance of PINN.
2) As in the example of Sec. 6.3, hybrid-PINN techniques that utilize both data and physics could be considered in case some data are available, and the performance may be better than that of using PINN without data. However, rather than applying to a specific problem, additional research is required for the general formulation of such hybrid approach, which may also be included in the aforementioned meta-learning technique to be further evaluated.
3) Learning multiple instances of forcing functions and BC/ICs discussed in Sec. 5.7 is a promising research direction that could resolve one of the issues of PINN that it is only possible to learn a single instance at once. The ULGNet could be particularly useful in this context and could be extended to a wider array of problem types, thus enhancing the applicability and robustness of PINN.

Acknowledgments

This research was supported by Korea Institute of Marine Science & Technology Promotion (KIMST) funded by the Ministry of Oceans and Fisheries (20220210).

References

[1] S. W. Kim, I. Kim and J. Lee, Seungchul knowledge integration into deep learning in dynamical systems: an overview and

taxonomy, *Journal of Mechanical Science and Technology,* 35 (2021) 1331-1342.
[2] M. P. Raissi and G. E. P. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *Journal of Computational Physics,* 378 (2019) 686-707.
[3] S. Cai et al., Physics-informed neural networks (PINNs) for fluid mechanics: a review, *Acta Mechanica Sinica*, 37 (12) (2021) 1727-1738.
[4] S. Cai et al., Physics-informed neural networks for heat transfer problems, *J. Heat Transfer*, 143 (6) (2021) 060801.
[5] G. E. Karniadakis et al., Physics-informed machine learning, *Nature Reviews Physics*, 3 (6) (2021) 422-440.
[6] S. Cuomo et al., Scientific Machine learning through physics–informed neural networks: where we are and what's next, *Journal of Scientific Computing*, 92 (3) (2022) 88.
[7] S. Das and S. Tesfamariam, State-of-the-art review of design of experiments for physics-informed deep learning, *arXiv:2202.06416* (2022) DOI: 10.48550/arXiv.2202.06416.
[8] X. Mou, Q. Fang and S. Li, A hybrid neural network and data sampling solver for forward and backward modiied diffusion equations, *Research Square* (2022) https://www.research-square.com/article/rs-2059725/v1 (Preprint).
[9] L. Lu et al., DeepXDE: a deep learning library for solving differential equations, *SIAM Review*, 63 (1) (2021) 208-228.
[10] J. M. Hanna et al., Residual-based adaptivity for two-phase flow simulation in porous media using physics-informed neural networks, *Computer Methods in Applied Mechanics and Engineering*, 396 (2022) 115100.
[11] C. Wu et al., A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks, *Computer Methods in Applied Mechanics and Engineering*, 403 (2023) 115671.
[12] W. Peng et al., RANG: a residual-based adaptive node generation method for physics-informed neural networks, *arXiv:2205.01051* (2022) DOI: 10.48550/arXiv.2205.01051.
[13] S. Subramanian et al., Adaptive self-supervision algorithms for physics-informed neural networks, *arXiv:2207.04084* (2022) DOI: 10.48550/arXiv.2207.04084.
[14] A. D. Jagtap, K. Kawaguchi and G. E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, *Journal of Computational Physics*, 404 (2020) 109136.
[15] A. D. Jagtap, K. Kawaguchi and G. Em Karniadakis, Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks, *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 476 (2239) (2020) 20200334.
[16] R. Gnanasambandam et al., Self-scalable tanh (stan): faster convergence and better generalization in physics-informed neural networks, *arXiv:2204.12589* (2022) DOI: 10.48550/arXiv.2204.12589.
[17] J. Abbasi and P. Ø. Andersen, Physical activation functions (pafs): an approach for more efficient induction of physics into physics-informed neural networks (PINNs), *arXiv:2205.14630* (2022) DOI: 10.48550/arXiv.2205.14630.
[18] W. Peng et al., Accelerating physics-informed neural network training with prior dictionaries, *arXiv:2004.08151* (2020) DOI: 10.48550/arXiv.2004.08151.
[19] K. He et al., Deep residual learning for image recognition, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA (2016).
[20] C. Cheng and G.-T. Zhang, Deep learning method based on physics informed neural network with resnet block for solving fluid flow problems, *Water*, 13 (2021) 423, DOI: 10.3390/w13040423.
[21] C. Moya and G. Lin, DAE-PINN: a physics-informed neural network model for simulating differential algebraic equations with application to power networks, *Neural Computing and Applications*, 35 (5) (2023) 3789-3804.
[22] V. Dwivedi and B. Srinivasan, Physics informed extreme learning machine (PIELM)–a rapid method for the numerical solution of partial differential equations, *Neurocomputing*, 391 (2020) 96-118.
[23] E. Schiassi et al., Extreme theory of functional connections: A fast physics-informed neural network method for solving ordinary and partial differential equations, *Neurocomputing*, 457 (2021) 334-356.
[24] A. A. Ramabathiran and P. Ramachandran, SPINN: sparse, physics-based, and partially interpretable neural networks for PDEs, *Journal of Computational Physics*, 445 (2021) 110600.
[25] 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, *Journal of Computational Physics*, 428 (2021) 110079.
[26] L. Sun et al., Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, *Computer Methods in Applied Mechanics and Engineering*, 361 (2020) 112732.
[27] L. Lu et al., Physics-informed neural networks with hard constraints for inverse design, *SIAM Journal on Scientific Computing*, 43 (6) (2021) B1105-B1132.
[28] R. Gong and Z. Tang, Further investigation of convolutional neural networks applied in computational electromagnetism under physics-informed consideration, *IET Electr. Power Appl.*, 16 (6) (2022) 653-674.
[29] X. Zhao et al., Physics-informed convolutional neural networks for temperature field prediction of heat source layout without labeled data, *Engineering Applications of Artificial Intelligence*, 117 (2023) 105516.
[30] Y. Yang and P. Perdikaris, Adversarial uncertainty quantification in physics-informed neural networks, *Journal of Computational Physics*, 394 (2019) 136-152.
[31] L. Yang et al., Highly-scalable, physics-informed GANs for learning solutions of stochastic PDEs, *2019 IEEE/ACM Third Workshop on Deep Learning on Supercomputers (DLS)*, Penver, CO, USA (2019).
[32] A. Daw, M. Maruf and A. Karpatne, PID-GAN: a GAN framework based on a physics-informed discriminator for

uncertainty quantification with physics, *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, *Association for Computing Machinery*, Virtual Event, Singapore (2021) 237-247.

[33] W. Zhong and H. Meidani, PI-VAE: physics-informed variational auto-encoder for stochastic differential equations, *Computer Methods in Applied Mechanics and Engineering*, 403 (2023) 115664.

[34] Y. A. Yucesan and F. A. C. Viana, A hybrid physics-informed neural network for main bearing fatigue prognosis under grease quality variation, *Mechanical Systems and Signal Processingy*, 171 (2022) 108875.

[35] R. Zhang, Y. Liu and H. Sun, Physics-informed multi-LSTM networks for metamodeling of nonlinear structures, *Computer Methods in Applied Mechanics and Engineering*, 369 (2020) 113226.

[36] P. Ren et al., PhyCRNet: physics-informed convolutional-recurrent network for solving spatiotemporal PDEs, *Computer Methods in Applied Mechanics and Engineering*, 389 (2022) 114399.

[37] L. Lu et al., Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, *Nature Machine Intelligence*, 3 (3) (2021) 218-229.

[38] K. Hornik and M. W. Stinchcombe, Halbert, multilayer feedforward networks are universal approximators, *Neural Networks*, 2 (1989) 359-366.

[39] S. Wang, H. Wang and P. Perdikaris, Learning the solution operator of parametric partial differential equations with physics-informed DeepONets, *Science Advances*, 7 (40) (2021) eabi8605.

[40] S. Cai et al., DeepM&Mnet: inferring the electroconvection multiphysics fields based on operator approximation by neural networks, *Journal of Computational Physics*, 436 (2021) 110296.

[41] 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*, 425 (2021) 109913.

[42] J. Li, J. Chen and B. Li, Gradient-optimized physics-informed neural networks (GOPINNs): a deep learning method for solving the complex modified KdV equation, *Nonlinear Dynamics*, 107 (1) (2022) 781-792.

[43] H. Gao, M. J. Zahr and J.-X. Wang, Physics-informed graph neural Galerkin networks: a unified framework for solving PDE-governed forward and inverse problems, *Computer Methods in Applied Mechanics and Engineering*, 390 (2022) 114502.

[44] W. Liu and M. J. Pyrcz, Physics-informed graph neural network for spatial-temporal production forecasting, *Geoenergy Science and Engineering*, 223 (2023) 211486.

[45] M. Yang and J. T. Foster, Multi-output physics-informed neural networks for forward and inverse PDE problems with uncertainties, *Computer Methods in Applied Mechanics and Engineering*, 402 (2022) 115041.

[46] P.-H. Chiu et al., CAN-PINN: a fast physics-informed neural network based on coupled-automatic–numerical differentiation method, *Computer Methods in Applied Mechanics and Engineering*, 395 (2022) 114909.

[47] C. L. Wight and J. Zhao, Solving allen-cahn and cahn-hilliard equations using the adaptive physics informed neural networks, *arXiv:2007.04542* (2020) DOI: 10.48550/arXiv. 2007.05452.

[48] S. Wang, Y. Teng and P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, *SIAM Journal on Scientific Computing*, 43 (5) (2021) 3055-3081.

[49] J. Yu et al., Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems, *Computer Methods in Applied Mechanics and Engineering*, 393 (2022) 114823.

[50] F. Xiong et al., Gradient-weighted physics-informed neural networks for one- dimensional euler equation dimensional euler equation, *TechRxiv.* (2022) DOI: 10.36227/ techrxiv.20099957.v1 (Preprint).

[51] Y. Liu et al., Physics-informed neural networks based on adaptive weighted loss functions for Hamilton-Jacobi equations, *Mathematical Biosciences and Engineering*, 19 (12) (2022) 12866-12896.

[52] S. Wang, X. Yu and P. Perdikaris, When and why PINNs fail to train: a neural tangent kernel perspective, *Journal of Computational Physics*, 449 (2022) 110768.

[53] D. Liu and Y. Wang, A dual-dimer method for training physics-constrained neural networks with minimax architecture, *Neural Networks*, 136 (2021) 112-125.

[54] L. D. McClenny and U. M. Braga-Neto, Self-adaptive physics-informed neural networks, *Journal of Computational Physics*, 474 (2023) 111722.

[55] P. Nasiri and R. Dargazany, Reduced-PINN: an integration-based physics-informed neural networks for stiff ODEs, *arXiv:2208.12045v1* (2022) DOI: 10.48550/arXiv.2208.12045.

[56] S. Maddu et al., Inverse dirichlet weighting enables reliable training of physics informed neural networks, *Machine Learning: Science and Technology*, 3 (1) (2022) 015026.

[57] E. Kharazmi, Z. Zhang and G. E. Karniadakis, Variational physics-informed neural networks for solving partial differential equations, *arXiv:1912.00873* (2019) DOI: 10.48550/arXiv.1912. 00873.

[58] R. Khodayi-Mehr and M. Zavlanos, VarNet: variational neural networks for the solution of partial differential equations, *Proceedings of the 2nd Conference on Learning for Dynamics and Control* (2020) 298-307.

[59] E. Kharazmi, Z. Zhang and G. E. M. Karniadakis, hp-VPINNs: variational physics-informed neural networks with domain decomposition, *Computer Methods in Applied Mechanics and Engineering*, 374 (2021) 113547.

[60] W. E. and B. Yu, The deep ritz method: a deep learning-based numerical algorithm for solving variational problems, *Communications in Mathematics and Statistics*, 6 (1) (2018) 1-12.

[61] J. Bai et al., A physics-informed neural network technique based on a modified loss function for computational 2D and 3D solid mechanics, *Computational Mechanics*, 71 (2022) 1-20.

[62] C. Wang et al., Is L^2 physics-informed loss always suitable for training physics-informed neural network? *Advances in Neural Information Proceeding Systems 35 (NeurIPS 2022)*, New Orleans, USA (2022).

[63] C. Davi and U. Braga-Neto, PSO-PINN: physics-informed neural networks trained with particle swarm optimization, *arXiv:2202.01943* (2022) DOI: 10. 48550/arXiv.2202.01943.

[64] B. Lu, C. Moya and G. Lin, NSGA-PINN: a multi-objective optimization method for physics-informed neural network training, *Algorithms,* 16 (4) (2023) 194.

[65] M. A. Nabian, R. J. Gladstone and H. Meidani, Efficient training of physics-informed neural networks via importance sampling, *Computer-Aided Civil and Infrastructure Engineering*, 36 (8) (2021) 962-977.

[66] Z. Yang, Z. Qiu and D. Fu, DMIS: dynamic mesh-based importance sampling for training physics-informed neural networks, *arXiv:2211.13944* (2022) DOI: 10.48550/arXiv.2211. 13944.

[67] D. He et al., Learning physics-informed neural networks without stacked back-propagation, *arXiv:2202.09340* (2022) DOI: 10.48550/arXiv.2202.09340.

[68] S. Markidis, The old and the new: can physics-informed deep-learning replace traditional linear solvers?, *Frontiers in Big Data*, 4 (2021).

[69] A. D. Jagtap, E. Kharazmi and G. E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems, *Computer Methods in Applied Mechanics and Engineering*, 365 (2020) 113028.

[70] X. Meng et al., PPINN: parareal physics-informed neural network for time-dependent PDEs, *Computer Methods in Applied Mechanics and Engineering*, 370 (2020) 113250.

[71] 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*, 28 (5) (2020) https://doi.org/10.4208 /cicp.oa-2020-0164.

[72] P. Stiller et al., Large-scale neural solvers for partial differential equations, *Driving Scientific and Engineering Discoveries Through the Convergence of HPC, Big Data and AI.*, Cham: Springer International Publishing (2020).

[73] Z. Hu et al., Augmented Physics-Informed Neural Networks (APINNs): a gating network-based soft domain decomposition methodology, *arXiv:2211.08939* (2022) DOI: 10.48550/arXiv. 2211.08 939.

[74] S. Wang, S. Sankaran and P. Perdilcaris, Respecting causality is all you need for training physics-informed neural networks, *arXiv:2203.07404* (2022) DOI: 10.48550/arXiv.2203. 07404.

[75] A. Daw et al., Mitigating propagation failures in PINNs using evolutionary sampling, *arXiv:2207.02338* (2022) DOI: 10.485 50/arXiv.2207.02338.

[76] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Springer-Verlag Berlin Heidelberg, Germany (2015).

[77] J. Guo, H. Wang and C. Hou, A novel adaptive causal sampling method for physics-informed neural networks, *arXiv:2210.12914* (2022) DOI: 10.48550/arXiv.2210.12914.

[78] R. Mattey and S. Ghosh, A novel sequential method to train physics informed neural networks for allen cahn and cahn hilliard equations, *Computer Methods in Applied Mechanics and Engineering*, 390 (2022) 114474.

[79] M. Penwarden et al., A unified scalable framework for causal sweeping strategies for physics-informed neural networks (PINNs) and their temporal decompositions, *arXiv:2302.14227* (2023) DOI: 10.48550/arXiv.2302.14227.

[80] A. F. Psaros, K. Kawaguchi and G. E. Karniadakis, Meta-learning PINN loss functions, *Journal of Computational Physics*, 458 (2022) 111121.

[81] S. Goswami et al., Transfer learning enhanced physics informed neural network for phase-field modeling of fracture, *Theoretical and Applied Fracture Mechanics*, 106 (2020) 102447.

[82] B. Bahmani and W. Sun, Training multi-objective/multi-task collocation physics-informed neural network with student/ teachers transfer learnings, *arXiv:2107.11496* (2021) DOI: 10. 48550/arXiv.2107.11496.

[83] S. Desai et al., One-shot transfer learning of physics-informed neural networks, *arXiv:2110.11286* (2021) DOI: 10.48550/ arXiv.2110.11286.

[84] C. Xu et al., Transfer learning based physics-informed neural networks for solving inverse problems in engineering structures under different loading scenarios, *Computer Methods in Applied Mechanics and Engineering,* 405 (2023) 115852.

[85] M. Penwarden et al., A metalearning approach for physics-informed neural networks (PINNs): application to para-meterized PDEs, *Journal of Computational Physics*, 477 (2023) 111912.

[86] W. Chen et al., Physics-informed machine learning for reduced-order modeling of nonlinear problems, *Journal of Computational Physics,* 446 (2021) 110666.

[87] K. Haitsiukevich and A. Ilin, Improved training of physics-informed neural networks with model ensembles, *arXiv:2204. 05108* (2022) DOI: 10.48550/arXiv.2204. 05108.

[88] J. Kim et al., DPM: a novel training method for physics-informed neural networks in extrapolation, *Proceedings of the AAAI Conference on Artificial Intelligence,* 35 (9) (2021) 8146-8154.

[89] K. Linka et al., Bayesian physics informed neural networks for real-world nonlinear dynamical systems, *Computer Methods in Applied Mechanics and Engineering,* 402 (2022) 115346.

[90] J. Sirignano and K. Spiliopoulos, DGM: a deep learning algorithm for solving partial differential equations, *Journal of Computational Physics*, 375 (2018) 1339-1364.

[91] B. Chudomelka et al., Deep neural network for solving differential equations motivated by legendre-galerkin appro-ximation, *arXiv:2010.12975* (2020) DOI: 10. 48550/arXiv.2010. 12975.

[92] J. Choi, N. Kim and Y. Hong, Unsupervised legendre-galerkin

neural network for solving partial differential equations, *IEEE Access*, 11 (2023) 23433-23446.
[93] S. Amini Niaki et al., Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture, *Computer Methods in Applied Mechanics and Engineering*, 384 (2021) 113959.
[94] H. K. Lee and S. In, Neural algorithm for solving differential equations, *Journal of Computational Physics,* 91 (1990) 110-131.
[95] D. C. U. Psichogios and H. Lyle, A hybrid neural network-first principles approach to process modeling, *AIChE J.,* 38 (1992) 1499-1511.
[96] M. W. M. G. Dissanayake and N. Phan-Thien, Neural-network-based approximations for solving partial differential equations, *Communications in Numerical Methods in Engineering,* 10 (1994) 195-201.
[97] I. E. Lagaris, A. Likas and D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, *IEEE Transactions on Neural Networks,* 9 (1998) 987-1000.
[98] P. Ramuhalli, L. Udpa and S. S. Udpa, Finite-element neural networks for solving differential equations, *IEEE Transactions on Neural Networks,* 16 (6) (2005) 1381-1392.
[99] A. S. B. Malek, Numerical solution for high order differential equations using a hybrid neural network—optimization method, *Applied Mathematics and Computation,* 183 (2006) 260-271.
[100] R. M. Shekari Beidokhti, Solving initial-boundary value problems for systems of partial differential equations using neural networks and optimization techniques, *Journal of the Franklin Institute*, 346 (2009) 898-913.
[101] M. Kumar and N. Yadav, Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: a survey, *Computers & Mathematics with Applications*, 62 (2011) 3796-3811.
[102] A. Griewank, On Automatic differentiation and algorithmic linearization, *Pesquisa Operacional*, 34 (3) (2014) 621-645.
[103] M. B. Abadi et al., TensorFlow: a system for large-scale machine learning, *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, Savannah, GA, USA (2016).
[104] M. P. Raissi, Paris karniadakis, george em, physics informed deep learning (part I): data-driven solutions of nonlinear partial differential equations, *arXiv:1711.10561* (2017) DOI: 10.48550 /arXiv.1711.10561.
[105] M. Raissi, P. Perdikaris and G. E. Karniadakis, Physics informed deep learning (part II): data-driven discovery of nonlinear partial differential equations, *arXiv:1711.10566* (2017) DOI: 10.48550.arXiv.1711.10566.

Appendix

Fig. A.1 summarizes the basis studies from which *Vanilla* PINN was established. In 1989, Hornik et al. [38] mathematically demonstrated the universal approximation capability of standard multilayer feedforward networks. In 1990, Lee and Kang [94] applied the neural minimization algorithm to the finite difference equations to derive a simple solution to differential equations. Psichogios and Ungar [95] proposed the 'first principles model' utilized together with artificial neural networks as a physical knowledge integration method and computed the relationship between input variables and process parameters. In 1994, Dissanayake and Phan-Thien [96] used artificial neural networks to solve partial differential equations. In 1998, Lagaris et al. [97] proposed an approach to solve both ordinary and partial differential equations using artificial neural networks and trial solutions. Later in 2005, Ramuhalli et al. [98] proposed finite-element neural network (FENN), which embedded the Finite-element model into ANN. In 2006, Malek and Beidokhti [99] utilized the Nelder-Mead method, which is one of the optimization techniques, together with ANN to compute an approximate solution of high-order (up to 4) ordinary differential equations. In 2009, Beidokhti and Malek [100] solved initial-boundary value problems using ANN, minimization techniques, and collocation methods and proposed a hybrid method that utilizes a trial solution with corresponding adjustable parameters. In 2011, Kumar and Yadav [101] classified and compared multilayer perceptron (MLP) and radial basis function (RBF) in terms of approximating solutions of differential equations. Then,

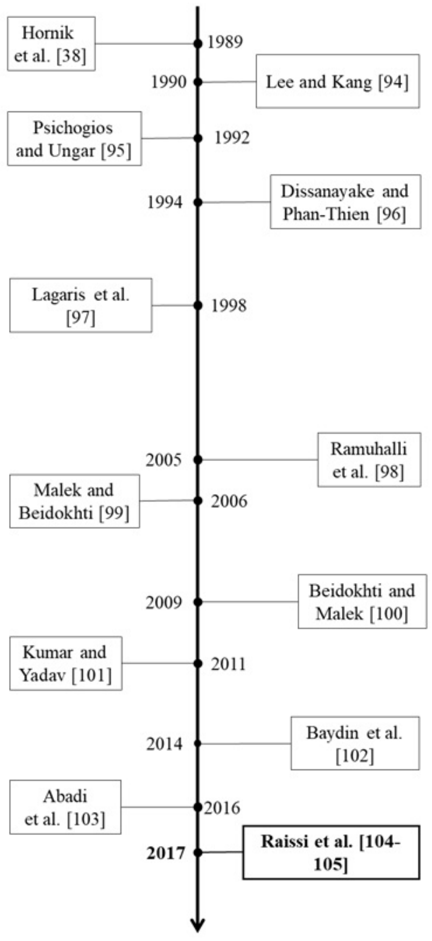

Fig. A.1. A timeline showing different approaches for prior knowledge integration into the artificial neural networks.

automatic differentiation (AD) was presented in 2014 [102], as well as TensorFlow in 2016 [103]. Based upon these studies, Raissi et al. [104, 105] proposed physics-informed neural network (PINN; *Vanilla* PINN) in two-part articles in 2017, followed by a combined version in 2019 [2].

Ikhyun Ryu received a B.S. degree in Mechanical Engineering from Karlsruhe Institute of Technology, Karlsruhe, Germany, in 2019. He then received a M.S. degree in Mechanical Engineering from Hanyang University, Seoul, South Korea, in 2022. He is now a Senior Research Engineer in PIDOTECH Inc. since 2022. His research interests include heat transfer, multidisciplinary design optimization, and Physics-Informed Neural Networks.

Gyu-Byung Park received a B.S. degree in Mechanical Engineering from Hanyang University, South Korea, in 2006. He then received a Ph.D. degree in Mechanical Engineering from Hanyang University, South Korea, in 2016. He is now a Senior Research Engineer in PIDOTECH Inc. since 2016. His research interests include Artificial Intelligence, Deep Learning, Design Optimization and Physics-Informed Neural Networks.

Yongbin Lee received a B.S. degree in Mechanical Engineering from Hanyang University, Seoul, South Korea, in 2002. He then received a M.S. degree in Mechanical Engineering from Hanyang University, Seoul, South Korea, in 2004. He then received a Ph.D. degree in Mechanical Engineering from Hanyang University, Seoul, South Korea, in 2009. He is now a Senior Research Engineer in PIDOTECH since 2012. His research interests include Machine Learning, Metamodeling, Design of Experiments, and Design Optimization.

Dong-Hoon Choi graduated from Seoul National University with a B.S. in mechanical engineering in 1975. He then graduated from KAIST in 1977 with an M.S. in Mechanical Engineering. He earned his Ph.D. in mechanical engineering from the University of Wisconsin-Madison. From 1986 to 2018, he served as a Professor of Mechanical Engineering at Hanyang University. He has been the CEO of PIDOTECH Inc. since 2003. His research interest includes AI-aided design optimization, multidisciplinary design optimization, surrogate-based design optimization, and AI applications for simulation and engineering design.