AI

MDPI

Review

Understanding Physics-Informed Neural Networks: Techniques, Applications, Trends, and Challenges

Amer Farea [1,2], Olli Yli-Harja [1,3] and Frank Emmert-Streib [1,*]

[1] Predictive Society and Data Analytics Lab, Faculty of Information Technology and Communication Sciences, Tampere University, 33720 Tampere, Finland; amer.farea@tuni.fi (A.F.); olli.yli-harja@tuni.fi (O.Y.-H.)
[2] Faculty of Engineering and Information Technology, Taiz University, Taiz P.O. Box 6803, Yemen
[3] Institute for Systems Biology, Seattle, WA 98195, USA
* Correspondence: frank.emmert-streib@tuni.fi

Abstract: Physics-informed neural networks (PINNs) represent a significant advancement at the intersection of machine learning and physical sciences, offering a powerful framework for solving complex problems governed by physical laws. This survey provides a comprehensive review of the current state of research on PINNs, highlighting their unique methodologies, applications, challenges, and future directions. We begin by introducing the fundamental concepts underlying neural networks and the motivation for integrating physics-based constraints. We then explore various PINN architectures and techniques for incorporating physical laws into neural network training, including approaches to solving partial differential equations (PDEs) and ordinary differential equations (ODEs). Additionally, we discuss the primary challenges faced in developing and applying PINNs, such as computational complexity, data scarcity, and the integration of complex physical laws. Finally, we identify promising future research directions. Overall, this survey seeks to provide a foundational understanding of PINNs within this rapidly evolving field.

Keywords: physics-informed neural networks; data-driven modeling; neural network architectures; inverse problems; ordinary differential equations; partial differential equations

Citation: Farea, A.; Yli-Harja, O.; Emmert-Streib, F. Understanding Physics-Informed Neural Networks: Techniques, Applications, Trends, and Challenges. *AI* **2024**, *5*, 1534–1557. https://doi.org/10.3390/ai5030074

Academic Editor: Giovanni Diraco

Received: 16 July 2024
Revised: 7 August 2024
Accepted: 19 August 2024
Published: 29 August 2024

1. Introduction

Neural networks (NNs), inspired by the biological structure of the human brain, have surged in popularity due to their ability to discern complex patterns and make predictions from data [1–3]. These models comprise interconnected nodes (neurons) organized into layers, each performing a simple mathematical operation. During a process known as training, NNs adjust their internal parameters to minimize the error between predicted and actual outputs, learning effectively from examples in a dataset. Over time, they have been successfully applied in fields such as computer vision, natural language processing, genetics, and cognitive science [4,5].

Physics-informed neural networks (PINNs) are a specialized type of neural network that integrates physics principles into their learning process [6–8]. Unlike traditional NNs that rely solely on data, PINNs incorporate domain-specific knowledge and physical laws to enhance their predictive capabilities, particularly in scientific and engineering contexts. By combining data-driven learning with physics-based constraints, PINNs offer a robust framework for solving complex problems governed by partial differential equations (PDEs), ordinary differential equations (ODEs), and other mathematical formulations [9–11]. This fusion enables the efficient and accurate modeling of physical phenomena, making PINNs well-suited for tasks like simulating fluid dynamics, predicting material properties, and solving inverse problems [12,13].

PINNs have emerged as a transformative approach at the intersection of machine learning and physical sciences, offering a novel paradigm for tackling complex problems governed by physical laws. By integrating principles from both fields, PINNs allow

researchers to leverage the expressive power of NNs while adhering to fundamental physical constraints.

Traditionally, machine learning and physics have operated independently, each addressing distinct challenges. Machine learning excels at uncovering complex patterns in data but often struggles to incorporate prior knowledge or enforce physical constraints [14,15]. Conversely, physics-based modeling, such as finite element methods or computational fluid dynamics, relies on explicit physical equations but may encounter scalability or generalization issues in complex real-world scenarios. PINNs bridge this gap by combining the flexibility of NNs with the rigor of physics principles [6]. Embedding physical laws into the learning process, PINNs provide a powerful framework for solving forward and inverse problems across various domains, including fluid dynamics, material science, and quantum mechanics. This integration enhances predictive accuracy, offers insights into physical phenomena, and facilitates the discovery of new scientific principles [16–18].

This paper is organized as follows: we start by providing background information (Section 2). Section 3 explores PINN architectures, techniques for integrating physical laws, and solving differential equations. In Section 5, we examine PINN applications. Section 6 discusses challenges and limitations. The Future Directions section (Section 7) outlines potential advances in algorithmic techniques, interdisciplinary collaborations, scalability, and addressing robustness issues. Finally, we provide concluding remarks in Section 8.

2. Background

Neural networks, a subset of machine learning models [19], have achieved remarkable success in various applications, ranging from computer vision [20–22], speech recognition [23], and natural language processing [24–27] to material science, fluid mechanics [28], genetics [29], cognitive science [30], genomics [31], and infrastructural health monitoring [32]. These models consist of multiple layers of interconnected neurons that process and learn from vast amounts of data. Through training utilizing techniques such as back-propagation and gradient descent, NNs iteratively adjust their weights to minimize prediction errors, thereby improving their performance.

Interestingly, the remarkable success of NNs in data-driven applications does not seamlessly extend to problems governed by complex physical laws. Many scientific and engineering problems, such as fluid dynamics, structural mechanics, and heat transfer, are described by PDEs that encapsulate the underlying physical principles. Unfortunately, traditional NNs do not inherently understand these physical principles but rely solely on data to learn accurate representations, and for this reason may produce physically inconsistent results. This limitation is particularly pronounced in scenarios where data are scarce or noisy.

To improve this shortcoming, PINNs have emerged as a powerful paradigm that integrates physical laws directly into the machine learning process. The core idea behind PINNs is to embed the governing equations of the physical system into the neural network's architecture or loss function. This is typically achieved by incorporating these equations into the loss function, which the network seeks to optimize during training. By doing so, PINNs ensure that the learned solutions not only fit the data but also adhere to the physical constraints imposed by the PDEs. This approach reduces the reliance on large datasets and leverages the strengths of both physics-based modeling and data-driven learning, resulting in models that are more accurate, generalizable, and interpretable.

The development of PINNs can be seen as part of a broader movement towards hybrid models that combine the deductive rigor of classical physics with the inductive power of machine learning. This fusion offers several advantages. First, it reduces the dependency on large datasets, as the physical laws provide a priori knowledge that guides the learning process. Second, it enhances the robustness and reliability of the models, ensuring that the predictions are physically consistent even in regions where data are sparse. Third, it opens up new avenues for solving inverse problems, where the goal is to infer unknown parameters or inputs from observed outputs, by leveraging both data and physical laws.

In this survey, we explore PINNs, focusing on their core methodologies and innovations. We review key applications across diverse fields, compare PINNs with traditional numerical methods, and discuss their advantages and challenges. We also examine recent trends and future directions, identifying potential areas for further research and application. This includes highlighting the benefits of PINNs over traditional methods and other machine learning approaches, as well as their limitations.

Related Work and Contribution

Previous reviews on PINNs can be found in [6,8,12,28,33,34]. However, these surveys are different in the follows ways. First, due to the origin of PINNs, many reviews focus on applications within physics, e.g., [6,12,28,35], and do not explore the full potential in general application domains. Second, reviews that do address non-physical applications focus only on one particular domain, e.g., [36–38]. Yet other reviews focus on bibliometric aspects of the literature about PINNs without going into methodological details [34] or provide only brief overviews [33,39].

In contrast, our survey provides a comprehensive yet balanced overview of the state-of-the-art research on PINNs. Specifically, our survey covers fundamental concepts, including an introduction to NNs and the motivations behind integrating physics-based constraints; methodologies, detailing various architectures and techniques used in PINNs for solving differential equations and inverse problems; applications, reviewing the diverse uses of PINNs across scientific domains; challenges and limitations, analyzing issues such as computational complexity, data scarcity, and the integration of complex physical laws; and future directions, exploring promising research avenues and emerging trends.

3. Methodologies

Deep learning has become widely adopted in scientific and engineering fields due to its ability to model complex relationships in various data types without requiring users to understand the system's physical parameters. Despite its versatility and effectiveness, challenges such as insufficient training data, lack of interpretability, and disregard for physical laws persist. To address these issues, enhanced methods have been developed, including physics-informed deep learning [7,11,40–43]. In this section, we discuss the methodologies employed in various studies related to PINNs. Our aim is to provide an overview of the diverse approaches utilized in the development, implementation, and evaluation of PINNs across different domains. From the choice of neural network architectures to the integration of physics-based constraints, each methodology offers unique insights into the fusion of machine learning and physics. By examining the methodologies in detail, we aim to uncover commonalities, challenges, and opportunities within the burgeoning field of PINNs.

The integration of prior knowledge into PINNs can be achieved through feature engineering, model construction, and regularization, as illustrated in Figure 1. The approach depends on the problem's nature and the type of neural network model being employed. By leveraging domain-specific knowledge and known physical principles, PINNs can achieve more accurate and interpretable predictions for various scientific and engineering applications.

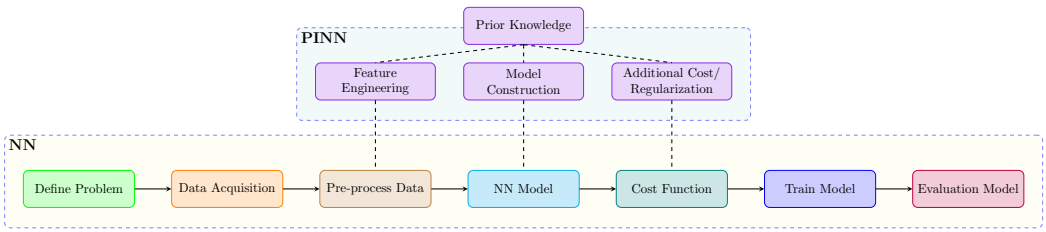

Figure 1. The general workflow for integrating prior knowledge into a neural network architecture to create a Physics-Informed Neural Network (PINN).

Different types of knowledge representation can be integrated into PINNs architecture to enhance their effectiveness [33,44–46].

3.1. Feature Engineering

Feature engineering plays a crucial role in structured data representation, leveraging prior knowledge from shared parameters in source data to enhance understanding of target data. For example, in fluid dynamics, features such as velocity, pressure, and temperature can significantly improve model learning and generalization. Incorporating additional features through physics-driven approaches, such as simulations, physical constraints, and governing equations, can be highly beneficial. These enhance learning by adding features derived from known physics equations or principles to the input data, facilitating more effective learning and generalization [47].

Structured data representation can also be integrated into model construction by designing neural network architectures that account for the data's underlying structure. For instance, Convolutional Neural Networks (CNNs) are suitable for grid-structured data like images, while Graph Neural Networks (GNNs) are designed for graph-structured data. This involves creating network structures that mirror the physical characteristics of the system or the data's nature. Examples include task-specific autoencoders, recurrent cells, or physics-inspired convolution filters, as outlined in the next section.

For problems involving temporal dependencies, feature engineering involves designing time-dependent features or incorporating lagged variables to capture temporal relationships. Domain-specific knowledge can guide feature selection. Recurrent Neural Networks (RNNs) are ideal for handling sequential data and modeling temporal evolution in PINNs. Regularization techniques like temporal smoothing ensure smooth transitions and consistency in predictions over time.

3.2. Model Construction

Neural networks include various architectures designed to integrate predefined cost functions into their learning process for accurate predictions. Their performance can be enhanced by incorporating prior knowledge to comply with physical constraints and governing laws when necessary. Several key architectures have emerged, each with distinct strengths [48].

Artificial Neural Networks (ANNs): A fully connected neural network [49,50], often referred to as a dense ANN, consists of interconnected layers, where each neuron in one layer is linked to every neuron in the next layer [51,52]. Mathematically, the output y of a single layer is computed as the activation function f applied to the weighted sum z of the inputs x with corresponding weights W and biases b, represented as follows:

$$z = Wx + b \tag{1}$$

$$y = f(z) \tag{2}$$

In the context of multiple layers, denoted by n, the process is iterated for each layer. Specifically, for the i-th layer, the output $y^{(i)}$ is computed as follows:

$$z^{(i)} = W^{(i)} y^{(i-1)} + b^{(i)} \tag{3}$$

$$y^{(i)} = f(z^{(i)}) \tag{4}$$

where $y^{(0)}$ represents the input vector x, and $W^{(i)}$ and $b^{(i)}$ denote the weight matrix and bias vector of the i-th layer, respectively. This recursive computation through layers enables the network to capture complex relationships within the data. Moreover, during training, the network aims to minimize a predefined cost function J, often represented as the mean squared error (MSE) for regression problems or cross-entropy for classification tasks. For example, in the case of MSE, the cost function can be written as follows:

$$J(W,b) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \tag{5}$$

where m is the number of training examples, y_i represents the true output, and $\hat{y}_i$ represents the predicted output.

ANNs are often used in PINNs for their simplicity and effectiveness in learning complex relationships; see [53–58].

Convolutional Neural Networks (CNNs): A CNN shares similarities with ANNs but includes convolution layers [59–61]. CNNs are particularly useful for problems involving grid-structured data, such as images or spatio-temporal data [59,62]. However, they can also accommodate numerical data with the utilization of a 1D CNN as the foundational model. The distinguishing characteristic of CNNs lies in their utilization of convolution filters, also known as kernels. These filters, square in shape, are applied to the input pixels through a process of convolution, traversing each square defined by a user-specified filter size and stride. By consistently employing the same filter across the input pixels, CNNs manage to maintain a concise set of hyperparameters [63].

A convolutional layer consists of multiple kernels that generate feature maps. Each neuron in a feature map is connected to a localized region (receptive field) in the previous layer. To obtain a feature map, the input is convolved with a kernel, followed by applying a nonlinear activation function [60].

Mathematically, the feature value $z^l_{i,j,k}$ at position (i,j) in the k-th feature map of the l-th layer is computed as follows:

$$z^l_{i,j,k} = \left(w^l_k\right)^T x^l_{i,j} + b^l_k \tag{6}$$

where w^l_k is the k-th filter's weight vector, b^l_k is its bias, and $x^l_{i,j}$ is the input patch at (i,j).

Following the convolution operation, an activation function $a(\cdot)$ is applied element-wise to introduce non-linearity:

$$a^l_{i,j,k} = a\left(z^l_{i,j,k}\right) \tag{7}$$

Subsequently, pooling layers are often employed to downsample the feature maps obtained from the convolutional layers, reducing dimensionality while preserving important information. A common pooling operation is max pooling, where the maximum value within a local region is retained.

$$y^l_{i,j,k} = \text{pool}\left(a^l_{m,n,k}\right), \quad \forall (m,n) \in R_{i,j} \tag{8}$$

where (m,n) represents the local neighborhood around (i,j), and $R_{i,j}$ is the region of the input map being pooled. Common pooling operations include average pooling and max pooling.

In CNNs, the training process typically involves minimizing a loss function $\mathcal{L}_{\text{total}}$, which combines a standard loss function $\mathcal{L}_{\text{std}}$ with a regularization term to prevent overfitting. This regularization term can include weight decay or dropout, among others. The total loss function is defined as follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{std}} + \lambda \mathcal{L}_{\text{reg}} \tag{9}$$

where λ is a hyperparameter controlling the regularization strength. This integrated approach allows for CNNs to effectively learn complex features from grid-structured data while mitigating the risk of overfitting during the training phase. In PINNs, CNNs can be employed when the problem involves spatial or spatio-temporal dependencies, see [41,64–70].

Recurrent Neural Networks (RNNs): RNNs are designed for handling time-series data [71,72]. This capability stems from the recursive arrangement of their hidden layers,

enabling them not only to generate outputs at specific time points but also to transmit hidden states to subsequent layers. Depending on the RNN variant, outputs at each time step may also be relayed forward. Conceptually, RNNs can be thought of as multiple instances of a standard ANN, with each instance communicating information to its successor [73–75]. This adaptation empowers RNNs to model data across individual time steps and the interconnections among them.

RNN architectures can vary to suit different problem types. Structurally, RNNs may operate in one-to-one, one-to-many, many-to-one, or many-to-many configurations [76,77]. A notable application of the many-to-many RNN structure is found in machine translation tasks. For instance, in a translation scenario, the encoder component of the network receives input words in English, while the decoder component generates corresponding translated words in French.

In dynamical systems, governing equations often describe the system's behavior over time, frequently taking the form of differential equations. These equations represent a valuable source of prior knowledge.

RNNs are designed to handle sequential data by maintaining an internal state or memory of past inputs. Mathematically, a recurrent neural network can be represented as follows:

$$\begin{aligned} h_t &= f(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \\ y_t &= g(W_{hy}h_t + b_y) \end{aligned} \tag{10}$$

where h_t is the hidden state at time t; x_t is the input at time t; W_{hh}, W_{xh}, W_{hy} are weight matrices; b_h and b_y are bias vectors; f is the activation function for the hidden layer; and g is the activation function for the output layer. For a given sequence of inputs $X = (x_1, x_2, \ldots, x_T)$, the output sequence $Y = (y_1, y_2, \ldots, y_T)$ is obtained by iterating the above equations through time steps $t = 1, 2, \ldots, T$.

The training of RNNs involves optimizing a cost function J that measures the discrepancy between the predicted outputs and the actual outputs. One common choice for the cost function is the mean squared error:

$$J = \frac{1}{T}\sum_{t=1}^{T} \|y_t - \hat{y}_t\|^2 \tag{11}$$

where $\hat{y}_t$ is the target output at time t.

In PINNs, RNNs can be employed for solving time-dependent problems governed by ODEs, such as predicting the behavior of dynamic systems over time; see [78–83].

Graph Neural Networks (GNNs): Deep learning models primarily focus on structured data like images and text [84,85]. However, data with irregular structures necessitate alternative processing methods. To tackle this challenge, GNNs have been developed, with Graph Convolutional Networks (GCNs) emerging as a prominent variant [84,86,87]. GCNs leverage the graph Fourier transform to extract features from graphs, similar to how CNNs utilize channels. They employ graph filters parameterized by the eigenvalues of a graph Laplacian matrix for feature extraction. The output layer of a GCN varies based on the task at hand; for node classification, a fully connected layer is often used.

The mathematical representation of GCNs can involve the following steps: GCNs may utilize a Fourier-like transformation to extract features from graph-structured data. The graph Fourier transform of a signal x on a graph G can be defined as follows:

$$\hat{x} = U^T x \tag{12}$$

where U is the matrix of eigenvectors of the graph Laplacian matrix.

Then, the GCNs employ graph filters parameterized by the eigenvalues of the graph Laplacian matrix. Given a graph signal x and a filter θ, the filtered signal $\tilde{x}$ is computed as follows:

$$\tilde{x} = \theta(\Lambda)x = U\theta(\Lambda)U^T x \tag{13}$$

where Λ is the diagonal matrix of eigenvalues of the graph Laplacian matrix.

The output layer of a GCN depends on the specific task. For example, for node classification, a fully connected layer followed by a softmax function can be used. The output y is computed as follows:

$$y = \text{softmax}(W\tilde{x} + b) \tag{14}$$

where W is the weight matrix and b is the bias vector.

The cost function for training a GCN depends on the specific task and can be formulated accordingly. For example, for node classification, a common choice is the cross-entropy loss function:

$$J(\theta) = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{i,k}\log(\hat{y}_{i,k}) \tag{15}$$

where N is the number of nodes, K is the number of classes, $y_{i,k}$ is the true label for node i and class k, and $\hat{y}_{i,k}$ is the predicted probability of node i belonging to class k.

In the context of PINNs, GNNs can be used to model complex physical systems characterized by interconnected components, such as molecular dynamics simulations or social network analysis; see [85,88–92].

Attention Mechanisms: Attention mechanisms allow for neural networks (NNs) to focus on specific parts of the input data, enabling them to weigh the importance of different features dynamically [93,94]. Mathematically, attention mechanisms can be represented as follows: consider a neural network f with parameter θ, and let x represent the input data. The attention mechanism $A(x)$ assigns weights to different parts of the input, allowing for the network to focus on the most relevant information. This attention mechanism can be formulated as follows:

$$A(x) = \text{softmax}(W_a \cdot \text{ReLU}(W_x \cdot x + b_x) + b_a) \tag{16}$$

where W_x, b_x, W_a, and b_a are learnable parameters, and softmax is the softmax function. The attention-weighted input $\tilde{x}$ is then computed as the element-wise product of the attention weights and the input:

$$\tilde{x} = A(x) \odot x \tag{17}$$

The attention-weighted input $\tilde{x}$ is then passed through the neural network f to obtain the output y:

$$y = f(\tilde{x}; \theta) \tag{18}$$

Finally, the cost function J of the PINN, which incorporates the attention mechanism, can be defined as follows:

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(y_i, \hat{y}_i) + \lambda R(\theta) \tag{19}$$

where N is the number of data points, $\mathcal{L}$ is the loss function, $\hat{y}_i$ is the ground truth label, and λ is the regularization parameter. The term $R(\theta)$ represents the regularization term, which penalizes large values of the network parameters θ to prevent overfitting.

In PINNs, attention mechanisms can enhance performance by selectively attending to relevant spatial or temporal features, particularly in problems with large or high-dimensional input spaces; see [95–99].

Generative models: Deep neural networks can be categorized into discriminative [100] and generative models [101] based on their function. Discriminative models predict the target variable given input variables, while generative models model the conditional probability of observable variables given the target. Two prominent generative models are variational autoencoders (VAEs) [102] and generative adversarial networks (GANs) [103].

Mathematically, a VAE can be represented as follows: Let X represent the input data, Z represent the latent space, and θ represent the parameters of the VAE. The encoder $q_\theta(Z|X)$ maps input data to the latent space distribution, and the decoder $p_\theta(X|Z)$ reconstructs the

input data from the latent space. The objective is to maximize the evidence lower bound (ELBO) given by

$$\mathcal{L}(\theta) = \mathbb{E}_{q_\theta(Z|X)}[\log p_\theta(X|Z)] - \mathrm{KL}(q_\theta(Z|X)||p(Z)) \tag{20}$$

where $p(Z)$ is the prior distribution over the latent space and KL denotes the Kullback–Leibler divergence.

A GAN can be represented as follows:

In a GAN, a generator network G produces samples from a prior distribution $p_{\text{data}}(X)$, and a discriminator network D distinguishes between real and generated samples. The objective of the generator is to minimize the following cost function:

$$\min_G \max_D V(D,G) = \mathbb{E}_{X\sim p_{\text{data}}(X)}[\log D(X)] + \mathbb{E}_{Z\sim p(Z)}[\log(1 - D(G(Z)))] \tag{21}$$

where $V(D,G)$ represents the value function of the minimax game.

In PINNs, autoencoders or VAEs can be employed to learn compact representations of physical systems, enabling efficient modeling and simulation for tasks such as data denoising, feature extraction, or uncertainty estimation; see [40,104–107].

Each of the preceding architectures offers unique advantages and is suited to different types of physical problems.

3.3. Incorporating Physical Laws as an Additional Cost Function

At the current state of PINN research, the methods discussed in this section are widely used to embed physical laws into neural networks. This ensures the resulting models are not only data-driven but also physically consistent and interpretable.

Incorporating physical laws into PINNs is crucial for ensuring that models accurately reflect underlying principles. Various techniques integrate these laws into the loss function, allowing for neural networks to capture intricate relationships while staying true to fundamental principles. The most frequent functional form of the loss is the Mean Squared Error (MSE) [7]; the p-norm is also used [108].

In this context, the cost function may include data fidelity and physics regularization terms that penalize deviations from known physical laws during training [109]. Alternatively, it might impose physical constraints based on conservation laws or constitutive relations as equality or inequality constraints on the neural network's output, ensuring that the predictions are physically plausible [110]. Additionally, the cost function can serve as a regularization term in the training objective to enforce physical consistency, penalizing deviations from physical laws while promoting smoothness and stability in the learned solutions [111,112].

Recent research has shown the feasibility of incorporating structured prior information into data-efficient and physics-informed learning systems. For instance, Gaussian process regression has been used to create tailored functional representations for specific linear operators, enabling precise solution inference and uncertainty assessments in mathematical physics [7,44,113].

The pioneering framework introduced by [7] incorporated physical laws into neural network learning processes, specifically for solving nonlinear PDEs in both forward and inverse problems. This integration of differential equations into the loss functions of a neural network guides the training process using physical constraints.

Building on this foundation, Raissi et al. introduced a deep learning framework called Hidden Fluid Mechanics (HFM), extending the concept of PINNs to complex systems governed by coupled and nonlinear PDEs [114]. These early contributions demonstrated the efficacy of PINNs in addressing forward and inverse problems, paving the way for broader adoption and further innovation.

Subsequent research has advanced PINNs' capabilities, include developing adaptive activation functions and multi-fidelity approaches to address PDE stiffness and improve

convergence rates [115]. Methods incorporating physical equations into deep learning model loss functions, such as those introduced by [41], enable training without labeled data, providing accurate predictions while respecting problem constraints and quantifying predictive uncertainty across diverse scenarios.

Hybrid Approaches combine elements of multiple techniques [6,34], such as physics-based loss functions and constraint-based methods, to simultaneously enforce physical constraints and data-driven objectives. Leveraging the complementary strengths of different techniques, hybrid approaches offer a versatile and effective framework for incorporating physical laws into PINNs.

Furthermore, advancements in optimization techniques, including physics-informed stochastic gradient descent and adaptive learning rates, have contributed to the robustness and efficiency of PINNs. In this context, Yang et al. [116] introduced the Bayesian physics-informed neural network (B-PINN), which integrates Bayesian neural networks (BNNs) [117] and PINNs to tackle both forward and inverse nonlinear problems involving PDEs and noisy data. B-PINNs leverage physical principles and noisy measurements within a Bayesian framework to provide predictions and evaluate uncertainty. Unlike PINNs, B-PINNs offer more accurate predictions and are better equipped to manage significant levels of noise by addressing overfitting. A systematic comparison between Hamiltonian Monte Carlo (HMC) [118] and variational inference (VI) [119] indicates a preference for HMC for posterior estimation. Furthermore, the substitution of the BNN in the prior with a truncated Karhunen–Loève (KL) expansion combined with HMC or a deep normalizing flow (DNF) model [120] shows potential but lacks scalability for high-dimensional problems.

4. Utility of PINNs

Differential equations are essential for understanding the dynamic behaviors of physical systems, as they describe how variables change relative to each other and enable predictions about future behavior. However, real-world systems often face challenges due to partially known differential equations:

1. Unknown Parameters: For example, in wind engineering [121], the equations of fluid dynamics are established, but coefficients related to turbulent flow are uncertain.
2. Unknown Functional Forms: In chemical engineering [122,123], the exact form of rate equations can be unclear due to uncertainties in reaction pathways.
3. Unknown Forms and Parameters: In battery state modeling [123,124], equivalent circuit models partially capture the current–voltage relationship, and key parameters like resistance and capacitance are unknown.

This partial knowledge hinders our understanding and control of these systems, making it essential to infer unknown components from observed data, a process known as system identification. This helps in predicting system states, informing control strategies, and enabling theoretical analysis.

Recently, Zhang et al. [125] proposed a strategy using PINN and symbolic regression to discover an unknown mathematical model in a system of ODEs. Though focused on Alzheimer's disease modeling, their approach shows potential for general equation discovery.

For solving differential equations using PINNs as shown below, various approaches exist. The overall process is illustrated in Figure 2, where the residuals of the differential equations are integrated into the loss function to enhance the accuracy of the neural network model. Here, ϵ denotes the acceptable margin of loss.

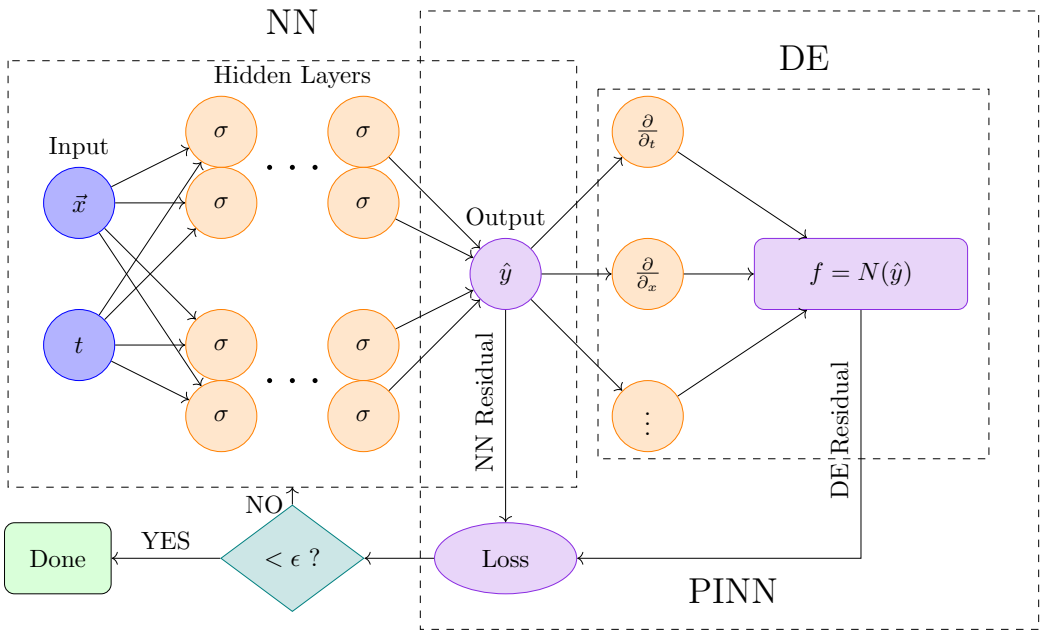

Figure 2. The process depicted involves integrating the residuals of the differential equations (DEs) with dynamics $N()$ into the loss function to improve the accuracy of the neural network (NN) model. Here, ϵ represents the acceptable margin of loss. In this context, the primary role of the neural network is to identify the optimal parameters that minimize the specified loss function while adhering to the initial and boundary conditions. Meanwhile, the supplementary cost function ensures that the constraints of the ODEs/PDEs are met, representing the physical information aspect of the neural network.

4.1. Solving PDEs and ODEs with PINNs

Physics-Informed Neural Networks (PINNs) offer a robust framework for solving Partial Differential Equations (PDEs) and Ordinary Differential Equations (ODEs), essential in modeling physical phenomena across various scientific and engineering domains. Several approaches exist for solving these equations with PINNs:

1. Finite Difference Methods: These discretize the spatial and temporal domains of PDEs and ODEs into a grid, approximating derivatives using finite differences. PINNs are trained to learn the solution directly from data at these grid points, leveraging neural networks' flexibility and scalability [126].
2. Collocation Method: This method enforces differential equations at discrete collocation points throughout the domain. PINNs minimize the residual of the PDEs or ODEs at these points, approximating the solution while satisfying the equations [127].
3. Boundary Integral Methods: These represent the solution to PDEs as an integral over the domain's boundary, reducing dimensionality and simplifying the numerical solution. PINNs trained to learn the boundary integral can solve PDEs with reduced computational cost and memory requirements [128].
4. Deep Galerkin Methods: These seek to minimize the residual of PDEs or ODEs over the entire domain. PINNs are trained to solve the equations in a strong sense by incorporating physical principles directly into the loss function [129].
5. Time-Stepping Methods: These discretize the temporal domain of time-dependent PDEs and ODEs into time steps, evolving the solution forward using iterative updates. PINNs learn the time evolution of the solution directly from data, utilizing neural networks' parallelism and scalability [130].

These approaches collectively provide versatile and efficient means of solving PDEs and ODEs with PINNs, enabling the development of accurate and scalable models for a broad spectrum of scientific and engineering applications.

4.2. Inverse Problems

Inverse problems occur in scientific and engineering disciplines when determining input parameters or conditions that result in specific observed behaviors or outputs. These problems are challenging because they require inferring hidden quantities from limited and noisy data [8,12].

Regularization: Such techniques are commonly used in inverse problems to impose constraints on the solution space and prevent overfitting. PINNs can be trained to minimize the discrepancy between predicted and observed data while simultaneously enforcing regularization constraints that ensure the solution remains physically plausible [111]. By incorporating prior knowledge about the underlying physics into the regularization process, PINNs can produce solutions that are both accurate and consistent with known physical principles [109].

Usually, regularization techniques in inverse problems are represented using an optimization framework. To give an example, let $\mathbf{x}$ represent the unknown parameter vector we aim to estimate, and let $\mathbf{y}_{\text{obs}}$ represent the observed data. In inverse problems, we typically have a model that relates the parameters $\mathbf{x}$ to the observed data $\mathbf{y}_{\text{pred}}$ through a function $\mathbf{F}$, i.e., $\mathbf{y}_{\text{pred}} = \mathbf{F}(\mathbf{x})$. A basic formulation of the inverse problem is to find the parameter vector $\mathbf{x}$ that minimizes the discrepancy between the observed and predicted data. This can be represented as follows:

$$\min_{\mathbf{x}} ||\mathbf{y}_{\text{obs}} - \mathbf{F}(\mathbf{x})||^2 \tag{22}$$

Regularization techniques add a penalty term to the objective function to control the complexity of the solution and prevent overfitting. This penalty term typically depends on some measure of the complexity of the parameter vector $\mathbf{x}$, denoted as $\Omega(\mathbf{x})$. A common form of regularization is Tikhonov regularization (also known as ridge regression), where the objective function becomes

$$\min_{\mathbf{x}} ||\mathbf{y}_{\text{obs}} - \mathbf{F}(\mathbf{x})||^2 + \lambda||\Omega(\mathbf{x})||^2 \tag{23}$$

Here, λ is a regularization parameter that controls the trade-off between fitting the data and the complexity of the solution. The term $||\Omega(\mathbf{x})||^2$ represents a measure of the norm of the parameter vector $\mathbf{x}$, which could be the L2-norm or another norm depending on the specific regularization technique used.

In the context of PINNs, the regularization term $\Omega(\mathbf{x})$ could incorporate prior knowledge about the physical principles governing the problem.

Bayesian inference: Such methods provide a probabilistic framework for solving inverse problems by estimating the posterior distribution of input parameters based on observed data [116]. This posterior distribution, $p(\theta|D)$, can be approximated using PINNs, which train a neural network to map input parameters to the posterior distribution [117]. Mathematically, we express this as follows:

$$p(\theta|D) = \frac{p(D|\theta) \cdot p(\theta)}{p(D)} \tag{24}$$

where $p(\theta|D)$ is the posterior distribution of the parameters given the data, $p(D|\theta)$ is the likelihood function, representing the probability of observing the data given the parameters, and $p(\theta)$ is the prior distribution, representing our initial belief about the parameters before observing any data.

The network's output, $\hat{p}(\theta|D)$, approximates the true posterior distribution. By incorporating physics-based priors, which encode known physical laws or constraints, PINNs enhance the accuracy and reliability of parameter estimates, even with limited and noisy data. This approach leverages domain knowledge during the inference process, producing more accurate results.

Data assimilation: By employing data assimilation techniques, observational data can be combined with numerical models to produce optimal estimates of the state variables and

parameters of a dynamical system [112]. PINNs can be used to assimilate observational data into physics-based models, effectively constraining the model predictions to be consistent with the observed data.

A basic mathematical representation for using data assimilation in inverse problems is formulated as follows: let $\mathbf{y}_{\text{obs}}$ denote the observed data, $\mathbf{x}$ represent the unknown state or parameters to be estimated, and $\mathbf{y}(\mathbf{x})$ denote the model predictions given the state or parameters $\mathbf{x}$.

In data assimilation, the goal is to find the optimal estimate $\mathbf{x}_{\text{est}}$ that minimizes the discrepancy between the observed data and the model predictions, often subject to additional constraints or regularization terms. This can be formulated as an optimization problem, typically solved using optimization techniques such as variational methods or Bayesian inference.

A common approach is to define a cost function that measures the misfit between the observed data and the model predictions, along with any additional regularization terms. The optimal estimate $\mathbf{x}_{\text{est}}$ is then obtained by minimizing this cost function:

$$\mathbf{x}_{\text{est}} = \arg\min_{\mathbf{x}} J(\mathbf{x}) \tag{25}$$

where $J(\mathbf{x})$ is the cost function, often defined as the sum of a data misfit term and a regularization term:

$$J(\mathbf{x}) = J_{\text{data}}(\mathbf{x}) + J_{\text{reg}}(\mathbf{x}) \tag{26}$$

The data misfit term, $J_{\text{data}}(\mathbf{x})$, quantifies the mismatch between the observed data and the model predictions:

$$J_{\text{data}}(\mathbf{x}) = \|\mathbf{y}_{\text{obs}} - \mathbf{y}(\mathbf{x})\|^2 \tag{27}$$

The regularization term, $J_{\text{reg}}(\mathbf{x})$, imposes additional constraints or penalties to ensure the solution is stable or adheres to prior knowledge:

$$J_{\text{reg}}(\mathbf{x}) = \text{regularization\_function}(\mathbf{x}) \tag{28}$$

The specific choice of regularization function depends on the problem and the desired properties of the solution.

Multi-physics methods: In many inverse problems, the underlying physical processes are governed by multiple interacting phenomena, making the problem inherently multi-physics [7,12,43,129]. PINNs can couple multiple physics-based models to solve the inverse problem in a unified framework. This involves incorporating multiple physics-based models into a system of coupled PDEs. By leveraging the expressive power of neural networks, PINNs can simultaneously estimate multiple input parameters or conditions from observed data, even in complex and highly coupled systems.

The mathematical representation of multi-physics coupling using PINNs involves incorporating multiple physics-based models into a unified framework. Let $f_i(u_i, x)$ denote the governing equation for the i-th physical process, where u_i represents the solution variable associated with that process and x denotes the spatial coordinates. The multi-physics coupling is achieved by simultaneously solving the following system of coupled PDEs:

$$\begin{aligned}
\frac{\partial u_1}{\partial t} &= f_1(u_1, x) + \sum_{j=2}^{N} \gamma_{1j} \cdot g_{1j}(u_j, x) \\
\frac{\partial u_2}{\partial t} &= f_2(u_2, x) + \sum_{j=1}^{N} \gamma_{2j} \cdot g_{2j}(u_j, x) \\
&\vdots \\
\frac{\partial u_N}{\partial t} &= f_N(u_N, x) + \sum_{j=1}^{N-1} \gamma_{Nj} \cdot g_{Nj}(u_j, x)
\end{aligned} \tag{29}$$

where N is the number of coupled physics processes, γ_{ij} represents the coupling coefficients, and $g_{ij}(u_j, x)$ describes the influence of the solution variable u_j from the j-th physics process on the i-th process. By leveraging the expressive power of neural networks, PINNs provide a flexible and efficient framework for solving such multi-physics problems.

Model Calibration and Uncertainty Quantification: Inverse problems often require not only estimating the input parameters but also quantifying uncertainty in the estimated solution. PINNs can be trained to produce not only point estimates of the input parameters but also probability distributions that quantify uncertainty in the estimated solution. By incorporating physics-based priors and regularization constraints into the training process, PINNs can produce more reliable estimates of the input parameters and a more accurate quantification of uncertainty in the estimated solution.

Let $\mathbf{y}$ denote the observed data or measurements, $\mathbf{m}$ the parameters of the model to be estimated, $\mathbf{f}(\mathbf{m})$ the forward model which maps parameters to predicted data, $\mathbf{e}$ the error or noise in the observations, and $\mathbf{u}$ the uncertainty associated with the estimated parameters.

Then, in a mathematical form, the model calibration and uncertainty quantification problem can be represented as follows:

1. Forward Model:
$$\mathbf{y} = \mathbf{f}(\mathbf{m}) + \mathbf{e} \tag{30}$$
2. Parameter Estimation:
$$\hat{\mathbf{m}} = \arg\min_{\mathbf{m}} ||\mathbf{y} - \mathbf{f}(\mathbf{m})||^2 \tag{31}$$
3. Uncertainty Quantification:
$$\mathbf{u} = \mathcal{F}(\mathbf{m}) \tag{32}$$

where $||\cdot||$ denotes a norm, such as the Euclidean norm, and $\mathcal{F}(\cdot)$ represents a function that quantifies uncertainty in the estimated parameters often through techniques like Bayesian inference or Monte Carlo methods.

4.3. Domain Decomposition

The incorporation of domain decomposition techniques represents another critical advancement in the evolution of PINNs. By partitioning the physical domain into smaller subdomains, these techniques facilitate parallel training and improve scalability, making it feasible to apply PINNs to large-scale problems. The introduction of neural operators, such as DeepONet [131], has further extended the PINN framework by enabling the learning of mappings between function spaces, thus allowing for the solution of families of PDEs rather than individual instances. The work discusses the potential of NNs in approximating nonlinear operators and proposes Deep Operator Networks (DeepONets) as a method to achieve accurate and efficient learning from relatively small datasets. DeepONets consist of two sub-networks: one for encoding input functions and another for encoding output function locations. Through systematic simulations of dynamic systems and PDEs, it is demonstrated that DeepONets significantly reduce generalization errors compared to fully connected networks. Theoretical analyses also reveal the dependence of approximation errors on the number of sensors and input function type, with computational results showing high-order error convergence rates, including polynomial rates and exponential convergence with respect to the training dataset size.

5. Applications

PINNs have shown remarkable potential across a spectrum of scientific and engineering disciplines. In this section, we highlight some applications from fluid dynamics, material science, quantum mechanics, and other fields. Table 1 shows an overview where the columns highlight different features. We would like to note that we also included PINNs for more general equations (GEs) such as models for predicting energy states [132,133] or drug responses [134,135].

Table 1. The application of PINN for tackling Forward Solution (FS) and Parameter Estimation (PE) in various contexts involving Ordinary Differential Equations (ODEs), Partial Differential Equations (PDEs), and Generalized Equations (GEs).

Application	Reference	FS	PE	ODEs	PDEs	GEs
Fluid Dynamics	[7]	✓	✓	–	✓	–
	[43]	✓	✓	–	✓	–
	[42]	✓	–	–	✓	–
	[114]	–	✓	–	✓	–
Material Science	[132]	✓	–	–	–	✓
	[133]	✓	✓	–	–	✓
	[136]	–	✓	–	✓	–
Structural Systems	[137]	✓	✓	–	✓	–
	[43]	✓	✓	–	✓	–
	[138]	✓	–	–	✓	–
	[139]	–	✓	–	✓	–
Quantum Mechanics	[7]	✓	✓	–	✓	–
	[140]	✓	–	✓	✓	–
	[141]	✓	–	–	✓	–
Geophysics	[142]	✓	–	–	✓	–
	[143]	–	✓	–	✓	–
Energy Systems	[121]	✓	–	–	–	✓
	[123]	–	✓	–	✓	–
	[144]	–	✓	✓	–	–
Oncology	[145]	✓	–	✓	–	–
	[134]	✓	–	–	–	✓
	[135]	✓	–	–	–	✓
	[146]	–	✓	✓	–	–

5.1. Fluid Dynamics

In fluid dynamics, a branch of physics examining fluid behavior under diverse conditions, PINNs have emerged as a robust tool for simulating and analyzing fluid phenomena [7,28,42,43,114]. Exploring applications in simulating turbulent flows, optimizing aerodynamic designs, predicting fluid-structure interactions, and more, PINNs amalgamate data-driven learning with physics-based modeling, offering a promising avenue for addressing complex fluid dynamics issues and advancing our understanding of fluid flow behavior.

5.2. Material Science

Moreover, material science [132,133,136], a multidisciplinary field probing material properties, structures, and behaviors, benefits greatly from PINNs. With applications in predicting material properties, optimizing material compositions and structures, and designing novel materials, PINNs accelerate material discovery and development. Combining data-driven learning with physics-based modeling, PINNs pave the way for innovations in material science, fostering advancements in manufacturing, electronics, energy storage, and healthcare.

5.3. Structural Systems

In structural systems, PINNs are applied to predict stress and strain distributions in complex structures [137], such as bridges, buildings, and mechanical components, under various loading conditions [43]. They also facilitate the detection and quantification of structural damage by interpreting sensor data and solving inverse problems. Within material science, PINNs enable the simulation of material behavior across different scales, from the atomic to the macroscopic levels, supporting the design of advanced materials

with tailored properties [139]. They are also instrumental in studying phase transitions and microstructural evolution in materials, providing insights into processes like solidification and grain growth [138].

5.4. Quantum Mechanics

In the realm of quantum mechanics, which elucidates phenomena at the atomic and subatomic levels, PINNs offer a valuable means to simulate and analyze quantum mechanical systems [7,140,141]. Enabling the simulation and analysis of complex quantum systems, PINNs facilitate the study of intricate quantum phenomena and the development of quantum technologies. From simulating quantum systems to predicting quantum properties, PINNs, through their fusion of data-driven learning and physics-based modeling, contribute significantly to the comprehension and utilization of quantum mechanics, driving innovations in quantum computing, cryptography, and sensing.

5.5. Geophysics

In geophysics, PINNs assist in interpreting seismic data to infer subsurface properties, aiding in oil and gas exploration and earthquake hazard assessment [143]. They are also used to model groundwater flow and contaminant transport, which support the management of water resources and environmental remediation efforts [142]. In biomedical engineering, PINNs are employed to simulate blood flow and cardiac mechanics, thereby contributing to the understanding and treatment of cardiovascular diseases [147]. Additionally, they are used to model tumor growth and spread, offering tools for predicting cancer progression and optimizing treatment strategies [148].

5.6. Energy Systems

In the field of energy systems, PINNs aid in simulating electrochemical processes in batteries, contributing to the development of more efficient and durable energy storage systems [123]. They are also used to model and optimize the performance of renewable energy systems, such as wind turbines [121] and power grid systems [144].

5.7. Oncology

The use of PINNs in cancer research has also shown significant promise. PINNs integrate physical laws and domain knowledge into models, which is particularly useful for understanding complex biological systems like cancer. They have been employed to model tumor growth by solving PDEs, providing accurate predictions of tumor behavior and enabling personalized treatment planning by estimating key parameters [145]. In drug response optimization [134,135], PINNs predict the pharmacokinetics and pharmacodynamics of anticancer drugs, helping to optimize dosing regimens and maximize therapeutic efficacy. For tumor-immune system interactions [146], PINNs model the dynamics of immune cell infiltration and the effects of immunotherapies, aiding in the design of new treatment strategies.

6. Challenges and Limitations

Although PINNs have promising capabilities, there are also challenges and limitations. In this section, we discuss some of the key challenges and constraints associated with the development and application of PINNs. These include data scarcity, computational complexity, integration of complex physical laws, and issues related to generalization and robustness.

6.1. Data-Related Issues

PINNs merge data-driven learning with physics-based modeling but face challenges with data availability, quality, and diversity, impacting model performance and reliability. Data scarcity limits the training of NNs, especially in underrepresented regions, leading to less accurate models. Data quality issues, such as noise and biases, also affect predictive

accuracy. Imbalanced data skew model training, while data heterogeneity requires robust preprocessing to ensure compatibility. Limited input ranges hinder extrapolation, and data biases affect generalization, necessitating meticulous data collection. Data interpretability remains challenging in high-dimensional spaces. Integrating PINNs with traditional NNs can mitigate these issues by functioning as an unsupervised method, eliminating the need for labeled data, and converting PDE problems into loss function optimization. The PINN algorithm embeds the mathematical model in the network, incorporating a residual term from the governing equation to constrain the solution space.

6.2. Computational Challenges

Large datasets pose significant computational challenges for PINNs, including high dimensionality, leading to an explosion of parameters and computational complexity, which requires substantial resources and can cause overfitting and slow convergence. Numerical instabilities often arise when solving PDEs or ODEs, particularly in stiff or ill-conditioned systems, necessitating careful numerical methods and regularization techniques. Scalability is another major issue, with memory limitations, communication overhead, and computational bottlenecks requiring distributed training and parallel computing. Hyperparameter tuning is complex and time-consuming, involving numerous parameters like network architecture and learning rates. Training times are considerable, especially for large datasets or complex problems, highlighting the need for more efficient algorithms. Additionally, interpretability is challenging, necessitating methods to understand and visualize predictions. Addressing these challenges requires advances in machine learning, numerical analysis, parallel computing, and domain-specific knowledge. Developing scalable algorithms, optimizing numerical methods, and leveraging hardware acceleration can help overcome these challenges and unlock the full potential of PINNs for complex scientific and engineering problems.

We would like to emphasize that such extensions are only meaningful if the performance of the PINN is improved. That means, in all cases, appropriate testing needs to be conducted to ensure quality, robustness, and reproducibility of the results.

6.3. Integration of Complex Physics

PINNs offer a promising framework for integrating complex physical laws into machine learning models, but several challenges need to be addressed for accuracy and reliability. Capturing nonlinear and multiscale phenomena is crucial, as many physical systems exhibit interactions across multiple spatial and temporal scales. Additionally, incorporating robust probabilistic modeling techniques is essential to handle uncertainty and variability due to factors like measurement noise and environmental fluctuations. Some physical systems exhibit chaotic or stochastic behavior, requiring specialized techniques like stochastic differential equations or probabilistic inference. Complex boundary conditions, such as non-homogeneous or time-varying conditions, must be accurately incorporated into PINN models for stability and accuracy. Multi-physics coupling, involving interactions between different physical domains, also presents a challenge and requires integrated models. Lastly, domain-specific knowledge is essential for identifying relevant physical principles, equations, parameters, and boundary conditions, guiding the development of accurate PINN models. Overcoming these challenges necessitates a multidisciplinary approach of combining advances in machine learning, numerical methods, and domain expertise, enabling the development of reliable models for complex physical systems.

6.4. Generalization and Robustness

Generalization and robustness are crucial for the performance and reliability of PINNs. Generalization ensures accurate predictions on unseen data, while robustness maintains performance despite perturbations and uncertainties in input data. Achieving these involves overcoming several challenges. Overfitting, where the model memorizes training data instead of learning general patterns, can be addressed with techniques like regu-

larization, dropout, and data augmentation. Conversely, underfitting, where the model is too simplistic to capture data patterns, requires more complex architectures and additional training data. Domain shift, where training and test data distributions differ, can degrade performance and is mitigated through domain adaptation and transfer learning. Out-of-distribution (OOD) detection is essential for identifying unreliable predictions, especially in critical applications, and involves methods like uncertainty estimation and anomaly detection. Transfer learning and domain adaptation enhance generalization by leveraging knowledge from related domains or tasks. Addressing these challenges requires algorithmic improvements, model regularization, data augmentation, and domain-specific expertise, ultimately enhancing the reliability, performance, and safety of PINN models across various applications.

7. Future Directions

As PINNs continue to evolve, several exciting directions for future research and development emerge. In this section, we explore some of the potential future directions for advancing the field of PINNs.

7.1. Algorithmic Advancements

Algorithmic advancements in neural networks (NNs) are crucial for the evolution and enhancement of Physics-Informed Neural Networks (PINNs). Key breakthroughs include novel training algorithms tailored to NNs, improving efficiency, scalability, and efficacy. Adaptive learning rate schedules optimize stability and convergence, especially with non-uniform gradients or noisy data. Advanced regularization techniques, incorporating domain-specific knowledge, enhance generalization and robustness. Uncertainty quantification methods improve prediction reliability, particularly in data-scarce environments. Self-supervised and semi-supervised learning with unlabeled or partially labeled data boost performance and generalization. Model compression and pruning streamline NN complexity, making them suitable for resource-constrained settings. Addressing adversarial robustness through robust optimization and defense mechanisms strengthens NNs against attacks, which is crucial for safety-critical applications. These advancements enhance PINNs' capabilities across scientific and engineering domains. By combining algorithmic innovation with domain-specific expertise, one can unlock new avenues for innovation and discovery with PINNs.

7.2. Interdisciplinary Collaborations

Interdisciplinary collaborations are crucial for advancing PINNs, fostering the exchange of ideas and collective problem-solving. These partnerships hold significant potential in various areas:

- Physics and Machine Learning: Physicists provide insights into physical principles, while machine learning specialists develop and optimize algorithms.
- Engineering and Data Science: Engineers contribute domain-specific knowledge in fields like fluid dynamics and structural mechanics, while data scientists handle data preprocessing, feature engineering, and model refinement.
- Medicine and Healthcare: Medical researchers and healthcare professionals work with machine learning experts to improve personalized medicine, disease diagnosis, and healthcare optimization using PINNs.
- Climate Science and Environmental Engineering: Climate scientists and environmental engineers collaborate with machine learning researchers to address climate change, environmental sustainability, and natural disaster mitigation.
- Materials Science, Nanotechnology, Robotics, and Autonomous Systems: In materials science, PINNs aid in discovering and optimizing new materials. In robotics, they enhance autonomous systems and control strategies. These interdisciplinary efforts leverage synergies, advancing innovation and knowledge across diverse fields.

7.3. Enhancements in Interpretability

Interpretability is crucial for NNs and PINNs, providing insights and fostering trust in their predictions, especially in scientific and engineering fields where transparency is essential. As PINNs find diverse applications, enhancing interpretability becomes vital. Key strategies include the following:

- Explainable Model Architectures: Use of sparse NNs and decision trees to clarify relationships between inputs and predictions.
- Feature Importance Analysis: Techniques like feature attribution and sensitivity analysis identify influential features and how input changes affect predictions.
- Visualization Techniques: Saliency maps and activation maximization help users understand model behavior and reasoning.
- Rule Extraction and Symbolic Reasoning: Provide concise representations of learned relationships for better comprehension.
- Domain-Specific Interpretability: Tailored to specific scientific and engineering contexts for relevant insights.
- Model-Agnostic Techniques: Surrogate models and global explanation methods facilitate understanding across different model types.

Enhancing interpretability in PINNs improves model transparency, reliability, and usability in real-world scenarios. It simplifies model understanding and debugging, allowing users to gain deeper insights into underlying physics and data relationships, driving scientific discovery and engineering innovation.

7.4. Digital Twins

Another promising application of PINNs is in digital twin research [149,150]. In this context, PINNs can be used for the parameter estimation of complex systems, providing models for problems in manufacturing, oncology, and economics. However, an extension is needed to enable online learning, allowing for continuous updates of parameters as new data become available over time. This extension bears similarities to meta-learning because two different forms of learning—inner learning for the PINN and outer learning for the updates—need to be simultaneously optimized.

7.5. Large-Scale and Real-Time Applications

Large-scale and real-time applications open new possibilities for PINNs, offering solutions to intricate problems. To fully utilize PINNs across scientific research, engineering design, and industry, improvements in scalability, efficiency, and deployment are crucial. Key strategies include developing scalable algorithms and architectures for large datasets, integrating with high-performance computing (HPC) for faster simulations, deploying in real-time decision support systems, incorporating online learning for adaptability, leveraging edge computing for localized inference, and ensuring robustness for safety-critical contexts. Focusing on these applications allows us to address complex challenges and foster innovation, highlighting the need for advancements in scalability, efficiency, and reliability to realize their full potential.

8. Conclusions

PINNs have emerged as a powerful framework for integrating physics-based modeling with data-driven learning that enable us to tackle complex scientific and engineering problems across diverse domains. In this survey, we explore the principles, applications, challenges, and future directions of PINNs, highlighting their potential to revolutionize scientific discovery, engineering design, and industrial innovation. The applications of PINNs span a wide range of domains, including fluid dynamics, material science, quantum mechanics, medicine, and climate research, demonstrating their versatility and effectiveness in addressing complex problems.

Looking ahead, the future of PINNs is promising, with exciting opportunities for innovation and discovery across a wide range of scientific and engineering domains. However,

further work is needed to develop systematic methods for optimizing network architectures for specific problems. Additionally, creating a taxonomy of approaches to different dynamic systems would facilitate easier application by others. Finally, extending PINNs into an online learning paradigm would enable their applicability in digital twin research.

Author Contributions: Conceptualization, A.F. and F.E.-S.; methodology, A.F. and F.E.-S.; writing—original draft preparation, A.F.; writing—review and editing, A.F., O.Y.-H., and F.E.-S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Gurney, K. *An Introduction to Neural Networks*; CRC Press: Boca Raton, FL, USA, 2018.
2. Crick, F. The recent excitement about neural networks. *Nature* **1989**, *337*, 129–132. [CrossRef]
3. Emmert-Streib, F. A heterosynaptic learning rule for neural networks. *Int. J. Mod. Phys. C* **2006**, *17*, 1501–1520. [CrossRef]
4. Bishop, C.M. Neural networks and their applications. *Rev. Sci. Instruments* **1994**, *65*, 1803–1832. [CrossRef]
5. Abiodun, O.I.; Jantan, A.; Omolara, A.E.; Dada, K.V.; Mohamed, N.A.; Arshad, H. State-of-the-art in artificial neural network applications: A survey. *Heliyon* **2018**, *4*, e00938. [CrossRef]
6. Cuomo, S.; Di Cola, V.S.; Giampaolo, F.; Rozza, G.; Raissi, M.; Piccialli, F. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *J. Sci. Comput.* **2022**, *92*, 88. [CrossRef]
7. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **2019**, *378*, 686–707. [CrossRef]
8. Karniadakis, G.E.; Kevrekidis, I.G.; Lu, L.; Perdikaris, P.; Wang, S.; Yang, L. Physics-informed machine learning. *Nat. Rev. Phys.* **2021**, *3*, 422–440. [CrossRef]
9. Mowlavi, S.; Nabi, S. Optimal control of PDEs using physics-informed neural networks. *J. Comput. Phys.* **2023**, *473*, 111731. [CrossRef]
10. Shin, Y.; Darbon, J.; Karniadakis, G.E. On the Convergence of Physics Informed Neural Networks for Linear Second-Order Elliptic and Parabolic Type PDEs. *Commun. Comput. Phys.* **2020**, *28*. [CrossRef]
11. Meng, X.; Li, Z.; Zhang, D.; Karniadakis, G.E. PPINN: Parareal physics-informed neural network for time-dependent PDEs. *Comput. Methods Appl. Mech. Eng.* **2020**, *370*, 113250. [CrossRef]
12. Cai, S.; Mao, Z.; Wang, Z.; Yin, M.; Karniadakis, G.E. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mech. Sin.* **2021**, *37*, 1727–1738. [CrossRef]
13. Chen, Y.; Lu, L.; Karniadakis, G.E.; Dal Negro, L. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Opt. Express* **2020**, *28*, 11618–11633. [CrossRef] [PubMed]
14. Jordan, M.I.; Mitchell, T.M. Machine learning: Trends, perspectives, and prospects. *Science* **2015**, *349*, 255–260. [CrossRef]
15. Mohri, M.; Rostamizadeh, A.; Talwalkar, A. *Foundations of Machine Learning*; MIT Press: Cambridge, MA, USA, 2018.
16. Yazdani, S.; Tahani, M. Data-driven discovery of turbulent flow equations using physics-informed neural networks. *Phys. Fluids* **2024**, *36*, 035107. [CrossRef]
17. Camporeale, E.; Wilkie, G.J.; Drozdov, A.Y.; Bortnik, J. Data-driven discovery of Fokker-Planck equation for the Earth's radiation belts electrons using Physics-Informed neural networks. *J. Geophys. Res. Space Phys.* **2022**, *127*, e2022JA030377. [CrossRef]
18. Chen, Z.; Liu, Y.; Sun, H. Physics-informed learning of governing equations from scarce data. *Nat. Commun.* **2021**, *12*, 6136. [CrossRef]
19. Emmert-Streib, F.; Moutari, S.; Dehmer, M. *Elements of Data Science, Machine Learning, and Artificial Intelligence Using R*; Springer: Berlin/Heidelberg, Germany.
20. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*. [CrossRef]
21. Sallam, A.A.; Amery, H.A.; Saeed, A.Y. Iris recognition system using deep learning techniques. *Int. J. Biom.* **2023**, *15*, 705–725. [CrossRef]
22. Sallam, A.A.; Mohammed, B.A.; Abdulbari, M. A Dorsal Hand Vein Recognition System based on Various Machine and Deep Learning Classification Techniques. In Proceedings of the IEEE 2023 3rd International Conference on Computing and Information Technology (ICCIT), Tabuk, Saudi Arabia, 13–14 September 2023; pp. 493–501.
23. Hinton, G.; Deng, L.; Yu, D.; Dahl, G.E.; Mohamed, A.r.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.N.; et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.* **2012**, *29*, 82–97. [CrossRef]

24. Farea, A.; Tripathi, S.; Glazko, G.; Emmert-Streib, F. Investigating the optimal number of topics by advanced text-mining techniques: Sustainable energy research. *Eng. Appl. Artif. Intell.* **2024**, *136*, 108877. [CrossRef]
25. Farea, A.; Emmert-Streib, F. Experimental Design of Extractive Question-Answering Systems:: Influence of Error Scores and Answer Length. *J. Artif. Intell. Res.* **2024**, *80*, 87–125. [CrossRef]
26. Sharma, T.; Farea, A.; Perera, N.; Emmert-Streib, F. Exploring COVID-related relationship extraction: Contrasting data sources and analyzing misinformation. *Heliyon* **2024**, *10*, e26973. [CrossRef] [PubMed]
27. Wu, Y.; Schuster, M.; Chen, Z.; Le, Q.V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv* **2016**, arXiv:1609.08144.
28. Brunton, S.L.; Noack, B.R.; Koumoutsakos, P. Machine learning for fluid mechanics. *Annu. Rev. Fluid Mech.* **2020**, *52*, 477–508. [CrossRef]
29. Libbrecht, M.W.; Noble, W.S. Machine learning applications in genetics and genomics. *Nat. Rev. Genet.* **2015**, *16*, 321–332. [CrossRef]
30. Lake, B.M.; Salakhutdinov, R.; Tenenbaum, J.B. Human-level concept learning through probabilistic program induction. *Science* **2015**, *350*, 1332–1338. [CrossRef]
31. Alipanahi, B.; Delong, A.; Weirauch, M.T.; Frey, B.J. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nat. Biotechnol.* **2015**, *33*, 831–838. [CrossRef] [PubMed]
32. Rafiei, M.H.; Adeli, H. A novel machine learning-based algorithm to detect damage in high-rise building structures. *Struct. Des. Tall Spec. Build.* **2017**, *26*, e1400. [CrossRef]
33. Kim, S.W.; Kim, I.; Lee, J.; Lee, S. Knowledge Integration into deep learning in dynamical systems: An overview and taxonomy. *J. Mech. Sci. Technol.* **2021**, *35*, 1331–1342. [CrossRef]
34. Lawal, Z.K.; Yassin, H.; Lai, D.T.C.; Che Idris, A. Physics-informed neural network (PINN) evolution and beyond: A systematic literature review and bibliometric analysis. *Big Data Cogn. Comput.* **2022**, *6*, 140. [CrossRef]
35. Sharma, P.; Chung, W.T.; Akoush, B.; Ihme, M. A review of physics-informed machine learning in fluid mechanics. *Energies* **2023**, *16*, 2343. [CrossRef]
36. Kashinath, K.; Mustafa, M.; Albert, A.; Wu, J.; Jiang, C.; Esmaeilzadeh, S.; Azizzadenesheli, K.; Wang, R.; Chattopadhyay, A.; Singh, A.; et al. Physics-informed machine learning: Case studies for weather and climate modelling. *Philos. Trans. R. Soc. A* **2021**, *379*, 20200093. [CrossRef] [PubMed]
37. Latrach, A.; Malki, M.L.; Morales, M.; Mehana, M.; Rabiei, M. A critical review of physics-informed machine learning applications in subsurface energy systems. *Geoenergy Sci. Eng.* **2024**, *239*, 212938. [CrossRef]
38. Huang, B.; Wang, J. Applications of physics-informed neural networks in power systems-a review. *IEEE Trans. Power Syst.* **2022**, *38*, 572–588. [CrossRef]
39. Pateras, J.; Rana, P.; Ghosh, P. A taxonomic survey of physics-informed machine learning. *Appl. Sci.* **2023**, *13*, 6892. [CrossRef]
40. Yang, Y.; Perdikaris, P. Adversarial uncertainty quantification in physics-informed neural networks. *J. Comput. Phys.* **2019**, *394*, 136–152. [CrossRef]
41. Zhu, Y.; Zabaras, N.; Koutsourelakis, P.S.; Perdikaris, P. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *J. Comput. Phys.* **2019**, *394*, 56–81. [CrossRef]
42. Sun, L.; Gao, H.; Pan, S.; Wang, J.X. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Comput. Methods Appl. Mech. Eng.* **2020**, *361*, 112732. [CrossRef]
43. Jagtap, A.D.; Kharazmi, E.; Karniadakis, G.E. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Comput. Methods Appl. Mech. Eng.* **2020**, *365*, 113028. [CrossRef]
44. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Inferring solutions of differential equations using noisy multi-fidelity data. *J. Comput. Phys.* **2017**, *335*, 736–746. [CrossRef]
45. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Machine learning of linear differential equations using Gaussian processes. *J. Comput. Phys.* **2017**, *348*, 683–693. [CrossRef]
46. Raissi, M.; Karniadakis, G.E. Hidden physics models: Machine learning of nonlinear partial differential equations. *J. Comput. Phys.* **2018**, *357*, 125–141. [CrossRef]
47. Di Lorenzo, D.; Champaney, V.; Marzin, J.; Farhat, C.; Chinesta, F. Physics informed and data-based augmented learning in structural health diagnosis. *Comput. Methods Appl. Mech. Eng.* **2023**, *414*, 116186. [CrossRef]
48. Emmert-Streib, F.; Yang, Z.; Feng, H.; Tripathi, S.; Dehmer, M. An introductory review of deep learning for prediction models with big data. *Front. Artif. Intell.* **2020**, *3*, 4. [CrossRef]
49. Agatonovic-Kustrin, S.; Beresford, R. Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *J. Pharm. Biomed. Anal.* **2000**, *22*, 717–727. [CrossRef] [PubMed]
50. Dongare, A.; Kharde, R.; Kachare, A.D. Introduction to artificial neural network. *Int. J. Eng. Innov. Technol. (IJEIT)* **2012**, *2*, 189–194.
51. Zou, J.; Han, Y.; So, S.S. Overview of artificial neural networks. In *Artificial Neural Networks: Methods and Applications*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 14–22.._2. [CrossRef]
52. Mehrotra, K.; Mohan, C.K.; Ranka, S. *Elements of Artificial Neural Networks*; MIT Press: Cambridge, MA, USA, 1997.
53. Daw, A.; Karpatne, A.; Watkins, W.D.; Read, J.S.; Kumar, V. Physics-guided neural networks (pgnn): An application in lake temperature modeling. In *Knowledge Guided Machine Learning*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2022; pp. 353–372.

54. Zhang, R.; Tao, H.; Wu, L.; Guan, Y. Transfer learning with neural networks for bearing fault diagnosis in changing working conditions. *IEEE Access* **2017**, *5*, 14347–14357. [CrossRef]
55. Pfrommer, J.; Zimmerling, C.; Liu, J.; Kärger, L.; Henning, F.; Beyerer, J. Optimisation of manufacturing process parameters using deep neural networks as surrogate models. *Procedia CiRP* **2018**, *72*, 426–431. [CrossRef]
56. Chao, M.A.; Kulkarni, C.; Goebel, K.; Fink, O. Fusing physics-based and deep learning models for prognostics. *Reliab. Eng. Syst. Saf.* **2022**, *217*, 107961. [CrossRef]
57. Wang, F.; Zhang, Q.J. Knowledge-based neural models for microwave design. *IEEE Trans. Microw. Theory Tech.* **1997**, *45*, 2333–2343. [CrossRef]
58. Yuan, F.G.; Zargar, S.A.; Chen, Q.; Wang, S. Machine learning for structural health monitoring: Challenges and opportunities. *Sens. Smart Struct. Technol. Civ. Mech. Aerosp. Syst. 2020* **2020**, *11379*, 1137903.
59. Li, Z.; Liu, F.; Yang, W.; Peng, S.; Zhou, J. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 6999–7019. [CrossRef] [PubMed]
60. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G.; Cai, J.; et al. Recent advances in convolutional neural networks. *Pattern Recognit.* **2018**, *77*, 354–377. [CrossRef]
61. Sallam, A.; Gaid, A.S.; Saif, W.Q.; Hana'a, A.; Abdulkareem, R.A.; Ahmed, K.J.; Saeed, A.Y.; Radman, A. Early detection of glaucoma using transfer learning from pre-trained cnn models. In Proceedings of the IEEE 2021 International Conference of Technology, Science and Administration (ICTSA), Taiz, Yemen, 22–24 March 2021; pp. 1–5.
62. Kattenborn, T.; Leitloff, J.; Schiefer, F.; Hinz, S. Review on Convolutional Neural Networks (CNN) in vegetation remote sensing. *ISPRS J. Photogramm. Remote Sens.* **2021**, *173*, 24–49. [CrossRef]
63. Xiang, Q.; Wang, X.; Lai, J.; Lei, L.; Song, Y.; He, J.; Li, R. Quadruplet depth-wise separable fusion convolution neural network for ballistic target recognition with limited samples. *Expert Syst. Appl.* **2024**, *235*, 121182. [CrossRef]
64. Sadoughi, M.; Hu, C. Physics-based convolutional neural network for fault diagnosis of rolling element bearings. *IEEE Sens. J.* **2019**, *19*, 4181–4192. [CrossRef]
65. De Bézenac, E.; Pajot, A.; Gallinari, P. Deep learning for physical processes: Incorporating prior scientific knowledge. *J. Stat. Mech. Theory Exp.* **2019**, *2019*, 124009. [CrossRef]
66. Zhang, R.; Liu, Y.; Sun, H. Physics-guided convolutional neural network (PhyCNN) for data-driven seismic response modeling. *Eng. Struct.* **2020**, *215*, 110704. [CrossRef]
67. Cohen, T.; Welling, M. Group equivariant convolutional networks. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 19–24 June 2016; pp. 2990–2999.
68. Dieleman, S.; De Fauw, J.; Kavukcuoglu, K. Exploiting cyclic symmetry in convolutional neural networks. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 19–24 June 2016; pp. 1889–1898.
69. Jia, X.; Willard, J.; Karpatne, A.; Read, J.; Zwart, J.; Steinbach, M.; Kumar, V. Physics guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles. In Proceedings of the 2019 SIAM International Conference on Data Mining, SIAM, Calgary, AB, Canada, 2–4 May 2019; pp. 558–566.
70. Worrall, D.E.; Garbin, S.J.; Turmukhambetov, D.; Brostow, G.J. Harmonic networks: Deep translation and rotation equivariance. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 5028–5037.
71. Sherstinsky, A. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Phys. D Nonlinear Phenom.* **2020**, *404*, 132306. [CrossRef]
72. Sutskever, I.; Martens, J.; Hinton, G.E. Generating text with recurrent neural networks. In Proceedings of the 28th International Conference on Machine Learning (ICML-11), Bellevue, WA, USA, 28 June–2 July 2011; pp. 1017–1024.
73. Medsker, L.R.; Jain, L. Recurrent neural networks. *Des. Appl.* **2001**, *5*, 2.
74. Pearlmutter. Learning state space trajectories in recurrent neural networks. In Proceedings of the IEEE International 1989 Joint Conference on Neural Networks, Washington, DC, USA, 18–22 June 1989; pp. 365–372.
75. Mandic, D.P.; Chambers, J. *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2001.
76. Das, S.; Tariq, A.; Santos, T.; Kantareddy, S.S.; Banerjee, I. Recurrent neural networks (RNNs): Architectures, training tricks, and introduction to influential research. In *Machine Learning for Brain Disorders*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 117–138.
77. Yu, Y.; Si, X.; Hu, C.; Zhang, J. A review of recurrent neural networks: LSTM cells and network architectures. *Neural Comput.* **2019**, *31*, 1235–1270. [CrossRef] [PubMed]
78. Nascimento, R.G.; Viana, F.A. Fleet prognosis with physics-informed recurrent neural networks. *arXiv* **2019**, arXiv:1901.05512.
79. Dourado, A.; Viana, F.A. Physics-informed neural networks for corrosion-fatigue prognosis. In Proceedings of the Annual Conference of the PHM Society, Paris, France, 2–5 May 2019; Volume 11.
80. Dourado, A.D.; Viana, F. Physics-informed neural networks for bias compensation in corrosion-fatigue. In Proceedings of the Aiaa Scitech 2020 Forum, Orlando, FL, USA, 6–10 January 2020; p. 1149.
81. Long, Y.; She, X.; Mukhopadhyay, S. Hybridnet: Integrating model-based and data-driven learning to predict evolution of dynamical systems. In Proceedings of the Conference on Robot Learning, PMLR, Zürich, Switzerland, 29–31 October 2018; pp. 551–560.

82. Yu, Y.; Yao, H.; Liu, Y. Structural dynamics simulation using a novel physics-guided machine learning method. *Eng. Appl. Artif. Intell.* **2020**, *96*, 103947. [CrossRef]
83. Lutter, M.; Ritter, C.; Peters, J. Deep lagrangian networks: Using physics as model prior for deep learning. *arXiv* **2019**, arXiv:1907.04490.
84. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Philip, S.Y. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 4–24. [CrossRef] [PubMed]
85. Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph neural networks: A review of methods and applications. *AI Open* **2020**, *1*, 57–81. [CrossRef]
86. Zhou, Y.; Zheng, H.; Huang, X.; Hao, S.; Li, D.; Zhao, J. Graph neural networks: Taxonomy, advances, and trends. *ACM Trans. Intell. Syst. Technol. (TIST)* **2022**, *13*, 1–54. [CrossRef]
87. Veličković, P. Everything is connected: Graph neural networks. *Curr. Opin. Struct. Biol.* **2023**, *79*, 102538. [CrossRef]
88. Ortega, A.; Frossard, P.; Kovačević, J.; Moura, J.M.; Vandergheynst, P. Graph signal processing: Overview, challenges, and applications. *Proc. IEEE* **2018**, *106*, 808–828. [CrossRef]
89. Seo, S.; Meng, C.; Liu, Y. Physics-aware difference graph networks for sparsely-observed dynamics. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
90. Seo, S.; Liu, Y. Differentiable physics-informed graph networks. *arXiv* **2019**, arXiv:1902.02950.
91. Zhang, G.; He, H.; Katabi, D. Circuit-GNN: Graph neural networks for distributed circuit design. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 7364–7373.
92. Mojallal, A.; Lotfifard, S. Multi-physics graphical model-based fault detection and isolation in wind turbines. *IEEE Trans. Smart Grid* **2017**, *9*, 5599–5612. [CrossRef]
93. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*. Available online: https://api.semanticscholar.org/CorpusID:13756489 (accessed on 18 August 2024).
94. Knyazev, B.; Taylor, G.W.; Amer, M. Understanding attention and generalization in graph neural networks. *Adv. Neural Inf. Process. Syst.* **2019**, *32*.
95. McClenny, L.; Braga-Neto, U. Self-adaptive physics-informed neural networks using a soft attention mechanism. *arXiv* **2020**, arXiv:2009.04544.
96. Rodriguez-Torrado, R.; Ruiz, P.; Cueto-Felgueroso, L.; Green, M.C.; Friesen, T.; Matringe, S.; Togelius, J. Physics-informed attention-based neural network for solving non-linear partial differential equations. *arXiv* **2021**, arXiv:2105.07898.
97. Rodriguez-Torrado, R.; Ruiz, P.; Cueto-Felgueroso, L.; Green, M.C.; Friesen, T.; Matringe, S.; Togelius, J. Physics-informed attention-based neural network for hyperbolic partial differential equations: Application to the Buckley–Leverett problem. *Sci. Rep.* **2022**, *12*, 7557. [CrossRef]
98. Jeddi, A.B.; Shafieezadeh, A. A physics-informed graph attention-based approach for power flow analysis. In Proceedings of the 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA), Pasadena, CA, USA, 13–16 December 2021; pp. 1634–1640.
99. Altaheri, H.; Muhammad, G.; Alsulaiman, M. Physics-informed attention temporal convolutional network for EEG-based motor imagery classification. *IEEE Trans. Ind. Inform.* **2022**, *19*, 2249–2258. [CrossRef]
100. Che, T.; Liu, X.; Li, S.; Ge, Y.; Zhang, R.; Xiong, C.; Bengio, Y. Deep verifier networks: Verification of deep discriminative models with deep generative models. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtually, 2–9 February 2021; Volume 35, pp. 7002–7010.
101. Salakhutdinov, R. Learning deep generative models. *Annu. Rev. Stat. Its Appl.* **2015**, *2*, 361–385. [CrossRef]
102. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114.
103. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. *Adv. Neural Inf. Process. Syst.* **2014**, *27*.
104. Warner, J.E.; Cuevas, J.; Bomarito, G.F.; Leser, P.E.; Leser, W.P. Inverse estimation of elastic modulus using physics-informed generative adversarial networks. *arXiv* **2020**, arXiv:2006.05791.
105. Yang, L.; Zhang, D.; Karniadakis, G.E. Physics-informed generative adversarial networks for stochastic differential equations. *SIAM J. Sci. Comput.* **2020**, *42*, A292–A317. [CrossRef]
106. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A.C. Improved training of wasserstein gans. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
107. Yang, Y.; Perdikaris, P. Physics-informed deep generative models. *arXiv* **2018**, arXiv:1812.03511.
108. Hammoud, M.A.E.R.; Titi, E.S.; Hoteit, I.; Knio, O. CDAnet: A Physics-Informed Deep Neural Network for Downscaling Fluid Flows. *J. Adv. Model. Earth Syst.* **2022**, *14*, e2022MS003051. [CrossRef]
109. Psaros, A.F.; Kawaguchi, K.; Karniadakis, G.E. Meta-learning PINN loss functions. *J. Comput. Phys.* **2022**, *458*, 111121. [CrossRef]
110. Soto, Á.M.; Cervantes, A.; Soler, M. Physics-informed neural networks for high-resolution weather reconstruction from sparse weather stations. *Open Res. Eur.* **2024**, *4*, 99. [CrossRef]
111. Davini, D.; Samineni, B.; Thomas, B.; Tran, A.H.; Zhu, C.; Ha, K.; Dasika, G.; White, L. Using physics-informed regularization to improve extrapolation capabilities of neural networks. In Proceedings of the Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS 2021), Vancouver, BC, Canada, 13 December 2021.

112. He, Q.; Barajas-Solano, D.; Tartakovsky, G.; Tartakovsky, A.M. Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. *Adv. Water Resour.* **2020**, *141*, 103610. [CrossRef]
113. Owhadi, H. Bayesian numerical homogenization. *Multiscale Model. Simul.* **2015**, *13*, 812–828. [CrossRef]
114. Raissi, M.; Yazdani, A.; Karniadakis, G.E. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science* **2020**, *367*, 1026–1030. [CrossRef]
115. Meng, X.; Karniadakis, G.E. A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems. *J. Comput. Phys.* **2020**, *401*, 109020. [CrossRef]
116. Yang, L.; Meng, X.; Karniadakis, G.E. B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. *J. Comput. Phys.* **2021**, *425*, 109913. [CrossRef]
117. Neal, R.M. *Bayesian Learning for Neural Networks*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012; Volume 118.
118. Neal, R.M. MCMC using Hamiltonian dynamics. *Handb. Markov Chain Monte Carlo* **2011**, *2*, 2.
119. Graves, A. Practical variational inference for neural networks. *Adv. Neural Inf. Process. Syst.* **2011**, *24*.
120. Rezende, D.; Mohamed, S. Variational inference with normalizing flows. In Proceedings of the International Conference on Machine Learning, PMLR, Lille, France, 7–9 July 2015; pp. 1530–1538.
121. Yucesan, Y.A.; Viana, F.A. A physics-informed neural network for wind turbine main bearing fatigue. *Int. J. Progn. Health Manag.* **2020**, *11*. [CrossRef]
122. Fedorov, A.; Perechodjuk, A.; Linke, D. Kinetics-constrained neural ordinary differential equations: Artificial neural network models tailored for small data to boost kinetic model development. *Chem. Eng. J.* **2023**, *477*, 146869. [CrossRef]
123. Pang, H.; Wu, L.; Liu, J.; Liu, X.; Liu, K. Physics-informed neural network approach for heat generation rate estimation of lithium-ion battery under various driving conditions. *J. Energy Chem.* **2023**, *78*, 1–12. [CrossRef]
124. Wang, Y.; Xiong, C.; Wang, Y.; Xu, P.; Ju, C.; Shi, J.; Yang, G.; Chu, J. Temperature state prediction for lithium-ion batteries based on improved physics informed neural networks. *J. Energy Storage* **2023**, *73*, 108863. [CrossRef]
125. Zhang, Z.; Zou, Z.; Kuhl, E.; Karniadakis, G.E. Discovering a reaction–diffusion model for Alzheimer's disease by combining PINNs with symbolic regression. *Comput. Methods Appl. Mech. Eng.* **2024**, *419*, 116647. [CrossRef]
126. Xiang, Z.; Peng, W.; Zhou, W.; Yao, W. Hybrid finite difference with the physics-informed neural network for solving PDE in complex geometries. *arXiv* **2022**, arXiv:2202.07926.
127. Hou, J.; Li, Y.; Ying, S. Enhancing PINNs for solving PDEs via adaptive collocation point movement and adaptive loss weighting. *Nonlinear Dyn.* **2023**, *111*, 15233–15261. [CrossRef]
128. Sun, J.; Liu, Y.; Wang, Y.; Yao, Z.; Zheng, X. BINN: A deep learning approach for computational mechanics problems based on boundary integral equations. *Comput. Methods Appl. Mech. Eng.* **2023**, *410*, 116012. [CrossRef]
129. Gao, H.; Zahr, M.J.; Wang, J.X. Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems. *Comput. Methods Appl. Mech. Eng.* **2022**, *390*, 114502. [CrossRef]
130. Chen, H.; Wu, R.; Grinspun, E.; Zheng, C.; Chen, P.Y. Implicit neural spatial representations for time-dependent PDEs. In Proceedings of the International Conference on Machine Learning, PMLR, Honolulu, HI, USA, 23–29 July 2023; pp. 5162–5177.
131. Lu, L.; Jin, P.; Karniadakis, G.E. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv* **2019**, arXiv:1910.03193.
132. Pun, G.P.; Batra, R.; Ramprasad, R.; Mishin, Y. Physically informed artificial neural networks for atomistic modeling of materials. *Nat. Commun.* **2019**, *10*, 2339. [CrossRef] [PubMed]
133. Mishin, Y. Machine-learning interatomic potentials for materials science. *Acta Mater.* **2021**, *214*, 116980. [CrossRef]
134. Sagingalieva, A.; Kordzanganeh, M.; Kenbayev, N.; Kosichkina, D.; Tomashuk, T.; Melnikov, A. Hybrid quantum neural network for drug response prediction. *Cancers* **2023**, *15*, 2705. [CrossRef]
135. Shin, J.; Piao, Y.; Bang, D.; Kim, S.; Jo, K. DRPreter: Interpretable anticancer drug response prediction using knowledge-guided graph neural networks and transformer. *Int. J. Mol. Sci.* **2022**, *23*, 13919. [CrossRef]
136. Zhang, E.; Dao, M.; Karniadakis, G.E.; Suresh, S. Analyses of internal structures and defects in materials using physics-informed neural networks. *Sci. Adv.* **2022**, *8*, eabk0644. [CrossRef]
137. Bolandi, H.; Sreekumar, G.; Li, X.; Lajnef, N.; Boddeti, V.N. Physics informed neural network for dynamic stress prediction. *Appl. Intell.* **2023**, *53*, 26313–26328. [CrossRef]
138. Zhu, Q.; Liu, Z.; Yan, J. Machine learning for metal additive manufacturing: Predicting temperature and melt pool fluid dynamics using physics-informed neural networks. *Comput. Mech.* **2021**, *67*, 619–635. [CrossRef]
139. Chaffart, D.; Yuan, Y.; Ricardez-Sandoval, L.A. Multiscale Physics-Informed Neural Network Framework to Capture Stochastic Thin-Film Deposition. *J. Phys. Chem. C* **2024**, *128*, 3733–3750. [CrossRef]
140. Desai, S.; Mattheakis, M.; Joy, H.; Protopapas, P.; Roberts, S. One-shot transfer learning of physics-informed neural networks. *arXiv* **2021**, arXiv:2110.11286.
141. Jin, H.; Mattheakis, M.; Protopapas, P. Physics-informed neural networks for quantum eigenvalue problems. In Proceedings of the IEEE 2022 International Joint Conference on Neural Networks (IJCNN), Padua, Italy, 18–23 July 2022; pp. 1–8.
142. Meray, A.; Wang, L.; Kurihana, T.; Mastilovic, I.; Praveen, S.; Xu, Z.; Memarzadeh, M.; Lavin, A.; Wainwright, H. Physics-informed surrogate modeling for supporting climate resilience at groundwater contamination sites. *Comput. Geosci.* **2024**, *183*, 105508. [CrossRef]

143. Waheed, U.B.; Alkhalifah, T.; Haghighat, E.; Song, C.; Virieux, J. PINNtomo: Seismic tomography using physics-informed neural networks. *arXiv* **2021**, arXiv:2104.01588.
144. Lakshminarayana, S.; Sthapit, S.; Maple, C. Application of physics-informed machine learning techniques for power grid parameter estimation. *Sustainability* **2022**, *14*, 2051. [CrossRef]
145. Rodrigues, J.A. Using Physics-Informed Neural Networks (PINNs) for Tumor Cell Growth Modeling. *Mathematics* **2024**, *12*, 1195. [CrossRef]
146. Raeisi, E.; Yavuz, M.; Khosravifarsani, M.; Fadaei, Y. Mathematical modeling of interactions between colon cancer and immune system with a deep learning algorithm. *Eur. Phys. J. Plus* **2024**, *139*, 1–16. [CrossRef]
147. Arzani, A.; Wang, J.X.; Sacks, M.S.; Shadden, S.C. Machine learning for cardiovascular biomechanics modeling: Challenges and beyond. *Ann. Biomed. Eng.* **2022**, *50*, 615–627. [CrossRef]
148. Perez-Raya, I.; Kandlikar, S.G. Thermal modeling of patient-specific breast cancer with physics-based artificial intelligence. *ASME J. Heat Mass Transf.* **2023**, *145*, 031201. [CrossRef]
149. Semeraro, C.; Lezoche, M.; Panetto, H.; Dassisti, M. Digital twin paradigm: A systematic literature review. *Comput. Ind.* **2021**, *130*, 103469. [CrossRef]
150. Emmert-Streib, F. Defining a digital twin: A data science-based unification. *Mach. Learn. Knowl. Extr.* **2023**, *5*, 1036–1054. [CrossRef]