Proceedings of the ASME 2019
International Design Engineering Technical Conferences
and Computers and Information in Engineering Conference
IDETC/CIE2019
August 18-21, 2019, Anaheim, CA, USA

IDETC2019-98115

MULTI-FIDELITY PHYSICS-CONSTRAINED NEURAL NETWORK AND ITS APPLICATION IN MATERIALS MODELING

Dehao Liu and Yan Wang[1]
Woodruff School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA

ABSTRACT

Training machine learning tools such as neural networks requires the availability of sizable data, which can be difficult for engineering and scientific applications where experiments or simulations are expensive. In this work, a novel multi-fidelity physics-constrained neural network is proposed to reduce the required amount of training data, where physical knowledge is applied to constrain neural networks, and multi-fidelity networks are constructed to improve training efficiency. A low-cost low-fidelity physics-constrained neural network is used as the baseline model, whereas a limited amount of data from a high-fidelity simulation is used to train a second neural network to predict the difference between the two models. The proposed framework is demonstrated with two-dimensional heat transfer and phase transition problems, which are fundamental in materials modeling. Physics is described by partial differential equations. With the same set of training data, the prediction error of physics-constrained neural network can be one order of magnitude lower than that of a classical artificial neural network without physical constraints. The accuracy of the prediction is comparable to those from direct numerical solutions of equations.

Keywords: machine learning; multi-fidelity model; physics-constrained neural networks; materials modeling; partial differential equations

1. INTRODUCTION

Machine learning (ML) tools, exemplified by the convolutional neural network and its derivatives, have demonstrated success in diverse fields. However, they are very data-hungry during training and can easily fail in applications where data are scarce and expensive to collect. The root cause is the "curse of dimensionality" in training the ML tools. As ML tools need to capture more detailed patterns or sensitive features, more complex modeling structures need to be introduced with more parameters and degrees of freedom. As a result, training algorithms need to explore and exploit in a very high-dimensional parameter space to search for optimal parameters. When the dimension increases, the volume of parameter spaces increases exponentially, so does the required amount of training data to cover the space and ensure the convergence of training. When the size of the training data set is small, overfitting can occur. That is, the training results in a spurious relationship that looks deceptively good but has low generality outside the labeled data range.

In various engineering and scientific applications, the cost of obtaining a large amount of data from high-fidelity simulations or experiments can be prohibitive. Data sparsity is not helpful to construct meaningful predictive ML models. Therefore, it is still a challenge for current state-of-the-art ML techniques to be applied in the domains of engineering or physical sciences. In the engineering domain, establishing high-dimensional process-structure-property relationships for either product or process design is the essential task. In engineering and scientific communities, human intelligence or knowledge has been embodied as physical laws or models based on centuries of data and knowledge accumulation. Giving up the available physical knowledge and purely relying on data-driven ML tools to identify the cause-effect relationships in physical sciences and engineering can be regarded as reinventing the wheel. Nevertheless, ML provides tools for systematic searching and exploring nonlinear and nonconvex relationships, which is much more efficient than ad hoc discovery. It is believed that training ML tools based on prior knowledge of physics can help navigate the high-dimensional parameter space with a small amount of training data.

[1] Contact author: yan.wang@me.gatech.edu

It is envisioned that the efficiency of training ML tools under the constraint of physical knowledge can be improved with small sample sizes. The physical laws or models can guide the searching and optimization procedures [1]. Generally, many physical laws are mathematically described as the relationships between physical quantities in the forms of ordinary differential equations (ODEs) or partial differential equations (PDEs). Some important and useful physical laws include but are not limited to conservation laws, laws of classical mechanics and thermodynamics. These physical laws have become the milestones of knowledge discovery in various scientific and engineering domains. Based upon physical laws or principles, various physics-based modeling and simulation techniques have been proposed to predict the behaviors of physical systems.

Incorporating physical meanings and physical knowledge in artificial neural networks (ANNs) has been studied from different perspectives. The first approach is to customize ANNs and incorporate physical meanings in the architecture. It has been demonstrated that ANN models can be applied to solve some special forms of optimization. For example, quadratic programming problems can be converted to linear complementarity problems and solved iteratively by projection neural networks [2,3]. Some efforts have been made for incorporating prior knowledge into ANNs in order to improve the training efficiency or prediction accuracy. Here, training efficiency means the convergence speed. For instance, prior knowledge can be applied as preprocessing tools to filter training data [4,5], or embedded as some analytical input-output functions in additional layers of ANNs [6], to improve the training efficiency. Prior knowledge can also be expressed as rules and interpreted with weights and basis functions in the ANN architecture, which could be further refined using training data [7,8]. Similarly, finite-element neural networks (FENNs) [9,10] can be constructed by transforming a finite element model to a neural network, where the weights of a FENN have physical meanings of material properties and can be computed in advance without training. FENNs have been used to obtain the solutions of differential equations for both forward and inverse problems. The major challenge of incorporating physical meanings into the ANN architecture is the complexity of customized networks. For instance, the number of weights in FENNs is related to the number of nodes, which could be very large for some high-dimensional problems with complex geometry.

The second approach to incorporate physical knowledge is treating it as constraints so that they can guide the training process. For instance, prior knowledge can be embedded into ANNs as architectural constraints and connection weight constraints to improve the training efficiency [11]. In addition to functional values, the information of derivatives has also been incorporated as prior knowledge for support vector regression [12]. ANNs have been used to approximate the solutions of PDEs. By transforming the original PDEs into their weighted residual forms, the prior knowledge of model forms and boundary values can be incorporated as penalty functions during the training of ANNs [13]. Similarly, the original model forms and boundary conditions, rather than their weighted residual forms, can be directly embedded as regularization terms into the objective function during the training process [14]. A regularization parameter has been introduced to control the trade-off between data fitting and knowledge-based regularization [15]. It has been shown that regularized ANNs such as multi-layer perceptron (MLP) and radial basis function (RBF) neural networks can help obtain the solutions of ODEs and PDEs with higher accuracy and lower memory requirement than traditional numerical methods [16]. The initial and boundary conditions can also be incorporated as the regularization terms to improve the efficiency of ANN training. For instance, a trial solution is formulated such that it contains the information of both boundary conditions and the model form [17,18]. However, it may be difficult to find trial solutions for boundary value problems that are defined on irregular boundaries. To tackle this problem, a MLP-RBF synergy model [19] was further proposed, where the first part of the trial solution was replaced by the RBF neural network so that the boundary conditions on irregular boundaries can be satisfied. Another way to handle arbitrary irregular boundaries is introducing a length factor [20] into the second part of the trial solution. As a measure of distance from the boundary, the length factor returns zero on the boundary and nonzero inside the boundary so that the first part of the trial solution is unaffected. Similarly, regularized ANNs were applied to approximate the solutions of ODEs [21], and a comparison was conducted between the performance of four different ANNs to solve ODEs [22]. Instead of regularization, information about boundary conditions can be explicitly used as equality constraints between the weights in ANNs such that a constrained backpropagation training can be taken [23–25]. The effectiveness of regularization during the ML training has been demonstrated in the above work. However, the training efficiency is still limited in high-dimensional problems, where the sampling of solutions from PDEs or ODEs can be costly.

In this paper, a multi-fidelity framework for the physics-constrained neural network (PCNN) is proposed to help construct high-dimensional surrogate models more efficiently. Here, PCNNs are constructed to approximate and predict the solutions of PDEs. Some solutions from simulations serve as the training data. The prior knowledge of PDEs, as well as the initial and boundary conditions, are applied to guide the training process of PCNNs with reduced searching space. The multi-fidelity concept is introduced to further reduce the required amount of training data. By combining a low-fidelity physics-constrained neural network (LF-PCNN) and a high-fidelity physics-constrained neural network (HF-PCNN), a multi-fidelity physics-constrained neural network (MF-PCNN) can be created with a lower training cost and higher prediction accuracy. The LF-PCNN is trained with low-fidelity simulation results, whereas the HF-PCNN is trained from high-fidelity simulations. The MF-PCNN is constructed by combining the predictions from the LF-PCNN and the difference between the LF-PCNN and HF-PCNN predictions. The advantage of the MF-PCNN is that the overall amount of training data can be reduced in order to achieve the similar level of accuracy by using the HF-PCNN

alone. In this paper, two examples are used to demonstrate the efficiency of the MF-PCNN framework. One example is the prediction of the temperature field in a heat transfer problem, and the other is the prediction of the phase field in a phase transition. It is shown that a MF-PCNN can be constructed with very limited simulation data to achieve a good accuracy of prediction.

In the remainder of this paper, the training of PCNNs, the construction of MF-PCNNs, and the setup of the computational scheme are described in Section 2. The computational results of the examples are shown in Section 3.

2. METHODOLOGY

In MF-PCNNs, the training data for LF-PCNNs and HF-PCNNs can be obtained from the analytical or numerical solutions of PDEs, e.g. from finite-element method (FEM). During the training, the prior knowledge about the form of PDEs or boundary values is added as the regularization terms in the loss function. The knowledge constraints provide guidance to the searching direction for optimization. The MF-PCNN is constructed based on the information from the LF-PCNN as well as the additional information that the HF-PCNN provides. The cost of obtaining high-fidelity information is higher than that of low-fidelity one. Therefore, the allocation of computational resources between high- and low-fidelity simulations can help reduce the overall training cost.

2.1 Training of PCNNs

Generally, a wide range of physical phenomena and dynamics can be described by PDEs, including heat transfer, advection-diffusion process, fluid dynamics, and others. Let us consider a time-dependent parametrized PDE with the general form

$$P\left(u, \frac{\partial u}{\partial t}, \frac{\partial u}{\partial \mathbf{x}}, \frac{\partial^2 u}{\partial t^2}, \frac{\partial^2 u}{\partial \mathbf{x}^2}, \dots\right) = f(t,\mathbf{x}),\ t \in [0,T],\ \mathbf{x} \in \Omega, \quad (1)$$

where $u(t,\mathbf{x})$ is the hidden solution to be found, $f(t,\mathbf{x})$ is a source or sink term, t is the time, $\mathbf{x} = (x_1, x_2, \dots, x_n)$ is the spatial vector, and $\Omega \in \mathbb{R}^n$ denotes the definition domain. This general PDE is subject to initial conditions (ICs)

$$I\left(u, \frac{\partial u}{\partial t}, \frac{\partial^2 u}{\partial t^2}, \dots\right) = g(\mathbf{x}),\ t = 0,\ \mathbf{x} \in \Omega, \quad (2)$$

and boundary conditions (BCs)

$$S\left(u, \frac{\partial u}{\partial t}, \frac{\partial u}{\partial \mathbf{x}}, \frac{\partial^2 u}{\partial t^2}, \frac{\partial^2 u}{\partial \mathbf{x}^2}, \dots\right) = h(t,\mathbf{x}),\ t \in [0,T],\ \mathbf{x} \in \partial\Omega, \quad (3)$$

where $\partial\Omega$ is the boundary of the definition domain. A more compact form of the above initial-boundary value problem can be written as

$$\mathbf{D}[u(t,\mathbf{x})] = f(t,\mathbf{x}),\ t \in [0,T],\ \mathbf{x} \in \Omega, \quad (4)$$

$$\mathbf{\Lambda}[u(0,\mathbf{x})] = g(\mathbf{x}),\ t = 0,\ \mathbf{x} \in \Omega, \quad (5)$$

$$\mathbf{\Gamma}[u(t,\mathbf{x})] = h(t,\mathbf{x}),\ t \in [0,T],\ \mathbf{x} \in \partial\Omega, \quad (6)$$

where $\mathbf{D}[\cdot]$, $\mathbf{\Lambda}[\cdot]$, and $\mathbf{\Gamma}[\cdot]$ are differential operators. For example, the three-dimensional (3D) heat equation without the source term corresponds to $\mathbf{D}[u(t,\mathbf{x})] = u_t - \alpha(u_{xx} + u_{yy} + u_{zz}) = 0$, where α is the thermal diffusivity, and the subscripts represent the partial derivatives with respect to either time or space.

In this work, the MLP architecture is used as a demonstration, which includes one input layer for $(t,\mathbf{x})$, multiple hidden layers, and one output layer for $U(t,\mathbf{x})$ to approximate the true solution $u(t,\mathbf{x})$. The neurons are connected with those in the neighbor layers, and the weights represent the strength of connections. The output from the hidden layer to the following layer is calculated as

$$y_i = \varphi\left(\sum w_{ij}\theta_j + b_i\right), \quad (7)$$

where w_{ij} is the weight of the connection between neuron j in the previous layer and neuron i in the current layer, θ_j is the j-th input value from the previous layer, and b_i is the bias for the neuron i in the current layer. φ is a nonlinear activation function, which can be sigmoid, tanh, rectified linear unit, or others.

The weights of a PCNN can be learned by minimizing the mean squared loss or total cost function

$$E = \lambda_T E_T + \lambda_P E_P + \lambda_I E_I + \lambda_s E_s, \quad (8)$$

where

$$E_T = \frac{1}{N_T}\sum_{i=1}^{N_T} |U(t_i^T, \boldsymbol{x}_i^T) - T(t_i^T, \boldsymbol{x}_i^T)|^2$$

is the loss caused by the discrepancy between the training data $T(\cdot)$ and the PCNN model prediction $U(\cdot)$, $\left\{t_i^{(\cdot)}, \mathbf{x}_i^{(\cdot)}\right\}$ denotes the sampling points in the defined domain, and $N_{(\cdot)}$ denotes the number of sampling points. Similarly,

$$E_P = \frac{1}{N_P}\sum_{i=1}^{N_P} |\mathbf{D}[U(t_i^P, \mathbf{x}_i^P)] - f(t_i^P, \mathbf{x}_i^P)|^2,$$

$$E_I = \frac{1}{N_I}\sum_{i=1}^{N_I} |\mathbf{\Lambda}[U(t_i^I, \mathbf{x}_i^I)] - g(\mathbf{x}_i^I)|^2,$$

and

$$E_S = \frac{1}{N_S}\sum_{i=1}^{N_S} \left|\mathbf{\Gamma}[U(t_i^S, \mathbf{x}_i^S)] - h(t_i^S, \mathbf{x}_i^S)\right|^2$$

are the losses caused by the violations of the model, initial conditions, and boundary conditions as the physical constraints from Eqs. (4)-(6). The constraint on weights of different losses is given as

$$\lambda_T + \lambda_P + \lambda_I + \lambda_s = 1. \quad (9)$$

The relative importance of prior knowledge can be adjusted by changing the weights of physical constraints λ_P, λ_I and λ_S. If the total loss function only includes the training loss E_T, then this is the traditional pure data-driven ANN to solve the initial-boundary value problem. By adding physical losses E_P, E_I and E_S as the regularization terms, the prior physical knowledge can help to reduce the size of searching space and provide guidance for the searching directions in training.

2.2 Construction of MF-PCNNs

The LF-PCNN and HF-PCNN must be trained first before the MF-PCNN is constructed. In this work, the fidelities are determined by the resolutions of FEM simulations given the same density of physical constraints. To be more specific, low-fidelity simulations are used to construct the LF-PCNN during a long time period $t \in [0,T]$, whereas high-resolution simulations are applied for the HF-PCNN during a short time period $t \in [0,T_0]$ $(T_0 < T)$.

After the LF-PCNN and HF-PCNN are trained, the difference between the predictions of the LF-PCNN $U_L(t,\mathbf{x})$ and HF-PCNN $U_H(t,\mathbf{x})$ is calculated as

$$\delta(t,\mathbf{x}) = U_H(t,\mathbf{x}) - U_L(t,\mathbf{x}),\ t \in [0,T_0],\ \mathbf{x} \in \Omega. \quad (10)$$

Then another ANN called difference artificial neural network (DANN) is constructed to predict the difference between the LF-PCNN and HF-PCNN, denoted as $U_\delta(t,\mathbf{x})$, during a longer time period $t \in [0,T]$. The weights of the DANN can be learned by using the observed difference $\delta(t,\mathbf{x})$ as the training data to minimize the mean squared error loss

$$E_\delta = \frac{1}{N_\delta}\sum_{i=1}^{N_\delta}|U_\delta(t_i,\boldsymbol{x}_i) - \delta(t_i,\boldsymbol{x}_i)|^2,\ t \in [0,T_0],\ \boldsymbol{x} \in \Omega, \quad (11)$$

where N_δ is the number of sampling points for the DANN. It is assumed that the evolution of the difference between the LF-PCNN and HF-PCNN during a longer time period $t \in [0,T]$ can be predicted by the DANN using the observed difference $\delta(t,\mathbf{x})$ as the training data during the short time period $t \in [0,T_0]$. Then the MF-PCNN is a combination of the LF-PCNN and DANN. The prediction from the MF-PCNN during the time period $t \in [0,T]$ is given by

$$U_M(t,\mathbf{x}) = U_L(t,\mathbf{x}) + U_\delta(t,\mathbf{x}),\ t \in [0,T],\ \mathbf{x} \in \Omega. \quad (12)$$

2.3 Experimental setup of the proposed MF-PCNN

The construction and training of the MF-PCNN are accomplished by using Tensorflow [26], which is an open-source Python library for machine learning. The partial derivatives of the ANNs are calculated based on the chain rules using the automatic differentiation [27]. Automatic differentiation is different from the numerical differentiation such as the method of finite difference. By applying the chain rules repeatedly, the derivatives of arbitrary order can be computed automatically and accurately to a working precision.

Two examples are applied to demonstrate the proposed MF-PCNN framework. The first example is a heat transfer problem where the evolution of the two-dimensional (2D) temperature distribution is modeled with the heat equation. The heat transfer example is used to demonstrate the effectiveness of the PCNN and test different weighting schemes of the total loss function. The second example is the phase transition problem where the evolution of the 2D phase field is modeled with the Allen-Cahn equation. The phase transition example is utilized to demonstrate the efficiency of the MF-PCNN framework.

The details of the computational setup for different ML models in the heat transfer and the phase transition examples are listed in TABLE 1 and TABLE 2, respectively. The ANNs, LF-PCNNs, and HF-PCNNs have the same structure of 30-20-30-20. That is, each of the networks has 4 layers. There are 30 neurons in the first and third layer, and 20 neurons in the second and last layer. The structures of the DANNs are simpler to avoid overfitting, which are 5-5-5-5 and 10-10-10-10. Two Gaussian process (GP) surrogate models with the RBF kernel are also constructed to predict the difference between the LF-PCNN and HF-PCNN for comparison purpose. The tanh function is used as the activation function. All of the loss functions of neural networks are minimized by using a gradient-based optimization algorithm called Adam [28] for the consideration of efficiency.

The training data for the ANNs, LF-PCNNs, and HF-PCNNs come from the FEM solutions of COMSOL, whereas the training data for the DANNs and GPs come from the observed differences between the predictions of the LF-PCNNs and HF-PCNNs during the short time period $t \in [0,T_0]$. The sampling strategy is uniform in both temporal and spatial dimensions for the convenience of comparison with the FEM solutions. Other sampling strategies such as random sampling, orthogonal sampling, and Latin Hypercube sampling can also be adopted.

TABLE 1 and TABLE 2 list the sizes of training data sets for the heat transfer and phase transition examples respectively. For instance, the amount of training data for the ANNs is 21×6×6, which means that there are 21 sampling points in the temporal dimension, 6 sampling points in the *x* direction of the spatial domain, and 6 sampling points in the *y* direction of the spatial domain. For the PCNNs in the heat transfer example, the number of physical constraints is 41×11×11. That is, there are 41 sampling points in the temporal dimension, 11 sampling points in the *x* direction and 11 sampling points in the *y* dimension of the spatial domain. The time period represents the size of the sampling space in the temporal dimension. In the heat transfer example, three different weighting schemes (PCNN1, PCNN2, and PCNN3) are compared. In the phase transition example, two HF-PCNNs (HF-PCNN1 and HF-PCNN2) are trained. The HF-PCNN1 is trained during the time period $t \in [0, 0.2]$, whereas the HF-PCNN2 is trained during two time periods, $t \in [0, 0.2]$ and $t \in [0.8, 1]$. Therefore, the amount of training data and the number of physical constraints for the HF-PCNN2 are twice those of the HF-PCNN1. The observed difference between the predictions of the LF-PCNN and HF-PCNN1 is served as the training data for the DANN1, DANN2, and GP1. Similarly, the observed difference between the predictions of the LF-PCNN and HF-PCNN2 is served as the training data for the DANN3, DANN4, and GP2. For ANNs, LF-PCNNs, and HF-PCNNs, the training of a neural network stops when the total loss *E* is lower than a threshold value 0.01. Similarly, the training of a DANN stops when the loss function E_δ is below 0.01.

TABLE 1: The setup for different ML models in the heat transfer example

ML model	Structure	Amount of training data (t×x×y)	Number of physical constraints (t×x×y)	Time period/s
ANN	30-20-30-20	21×6×6	0	[0, 1]
PCNN1, PCNN2, PCNN3	30-20-30-20	21×6×6	41×11×11	[0, 1]

TABLE 2: The setup for different ML models in the phase transition example

ML model	Structure	Amount of training data ($t\times x\times y$)	Number of physical constraints ($t\times x\times y$)	Time period/s
ANN	30-20-30-20	21×6×6	0	[0, 1]
LF-PCNN	30-20-30-20	21×6×6	21×11×11	[0, 1]
HF-PCNN1	30-20-30-20	9×21×21	5×11×11	[0, 0.2]
HF-PCNN2	30-20-30-20	18×21×21	10×11×11	[0, 0.2], [0.8, 1]
DANN1	5-5-5-5	9×26×26	0	[0, 0.2]
DANN2	10-10-10-10	9×26×26	0	[0, 0.2]
DANN3	5-5-5-5	18×26×26	0	[0, 0.2], [0.8, 1]
DANN4	10-10-10-10	18×26×26	0	[0, 0.2], [0.8, 1]
GP1	RBF kernel	9×26×26	0	[0, 0.2]
GP2	RBF kernel	18×26×26	0	[0, 0.2], [0.8, 1]

3. EXPERIMENTAL RESULTS

In this section, the results for the heat transfer and phase transition examples are shown. The heat transfer example is used to demonstrate the effectiveness of the PCNN and test different weighting schemes of the total loss function. A convergence analysis for the ANN and the PCNN is also conducted. The phase transition problem is to demonstrate the efficiency of the MF-PCNN framework.

3.1 Heat equation

The evolution of temperature distributions can be modeled by parabolic PDEs. The heat equation describes the diffusion process of energy, which is important in modeling microstructure evolution during phase transition. The 2D heat equation with the zero Neumann boundary condition used in this example is

$$\begin{cases} u_t - 0.01\left(u_{xx}+u_{yy}\right)=0, \;\; t,x,y\in[0,1], \\ u(0,x,y)=0.5[sin(4\pi x)+sin(4\pi y)], \\ u_x(t,0,y)=0, \\ u_x(t,1,y)=0, \\ u_y(t,x,0)=0, \\ u_y(t,x,1)=0. \end{cases}, \quad (13)$$

where u is the 2D temperature field.

The goal of training a neural network is to ensure the prediction $U(t,x,y)$ from the neural network can approximate the true solution $u(t,x,y)$ from FEM simulations with the desired accuracy. Here, the physical loss is

$$E_P=\frac{1}{N_P}\sum_{i=1}^{N_P}\left|\begin{matrix} U_t(t_i^P,x_i^P,y_i^P) \\ -0.01U_{xx}(t_i^P,x_i^P,y_i^P) \\ -0.01U_{yy}(t_i^P,x_i^P,y_i^P)\end{matrix}\right|^2. \quad (14)$$

The initial loss is given by

$$E_I=\frac{1}{N_I}\sum_{i=1}^{N_I}\left|\begin{matrix} U(0,x_i^I,y_i^I) \\ -0.5[sin(4\pi x_i^I)+sin(4\pi y_i^I)]\end{matrix}\right|^2. \quad (15)$$

The boundary loss is

$$E_S=\frac{1}{N_S}\sum_{i=1}^{N_S}\left[\begin{matrix} \left|U_x(t_i^S,0,y_i^S)\right|^2+\left|U_x(t_i^S,1,y_i^S)\right|^2 \\ +\left|U_y(t_i^S,x_i^S,0)\right|^2+\left|U_y(t_i^S,x_i^S,1)\right|^2\end{matrix}\right]. \quad (16)$$

To assess the sensitivity of weights, three weighting schemes of the total loss function are tested and compared with each other. In the PCNN1, the weights are equal and fixed in the total loss function

$$E=0.25(E_T+E_P+E_I+E_s). \quad (17)$$

In the PCNN2, the weights are unequal and fixed in the total loss function

$$E=0.125(E_T+2E_P+4E_I+E_s). \quad (18)$$

In the PCNN3, the weights are adaptive during the training, which are proportional to the percentages of individual losses in the total loss function

$$E=\frac{E_T^2+E_P^2+E_I^2+E_S^2}{E_T+E_P+E_I+E_s}. \quad (19)$$

Assigning higher weights to the physical constraints indicates that prior knowledge will be more influential in the training process. When the training data is sparse, increasing the number of physical constraints can help improve the training efficiency. In addition, the weights of physical constraints need to be large enough in order to ensure the training efficiency and prediction accuracy. When the weights of physical constraints are assigned, it is also necessary to consider the balance among different losses such that the reduction speeds of the four errors are comparable. The ideal case is that the four losses are reduced at the same speed so that the overall reduction speed of the total loss is maximized.

Here, the training data come from the FEM solutions. FIGURE 1 shows the original FEM solution of the temperature field, as well as the predictions by the traditional ANN, the equally-weighted PCNN1, the unequally-weighted PCNN2, and the adaptively-weighted PCNN3 at $t = 1$, respectively. The errors of the predicted temperature fields compared with the original FEM solution for different neural networks at $t = 1$ are shown in FIGURE 2. Here, the prediction error is the absolute difference between the prediction from a neural network and the FEM solution. The dots in the figures indicate the evaluation points. There are 26×26 evaluation points in the 2D domain. It is seen

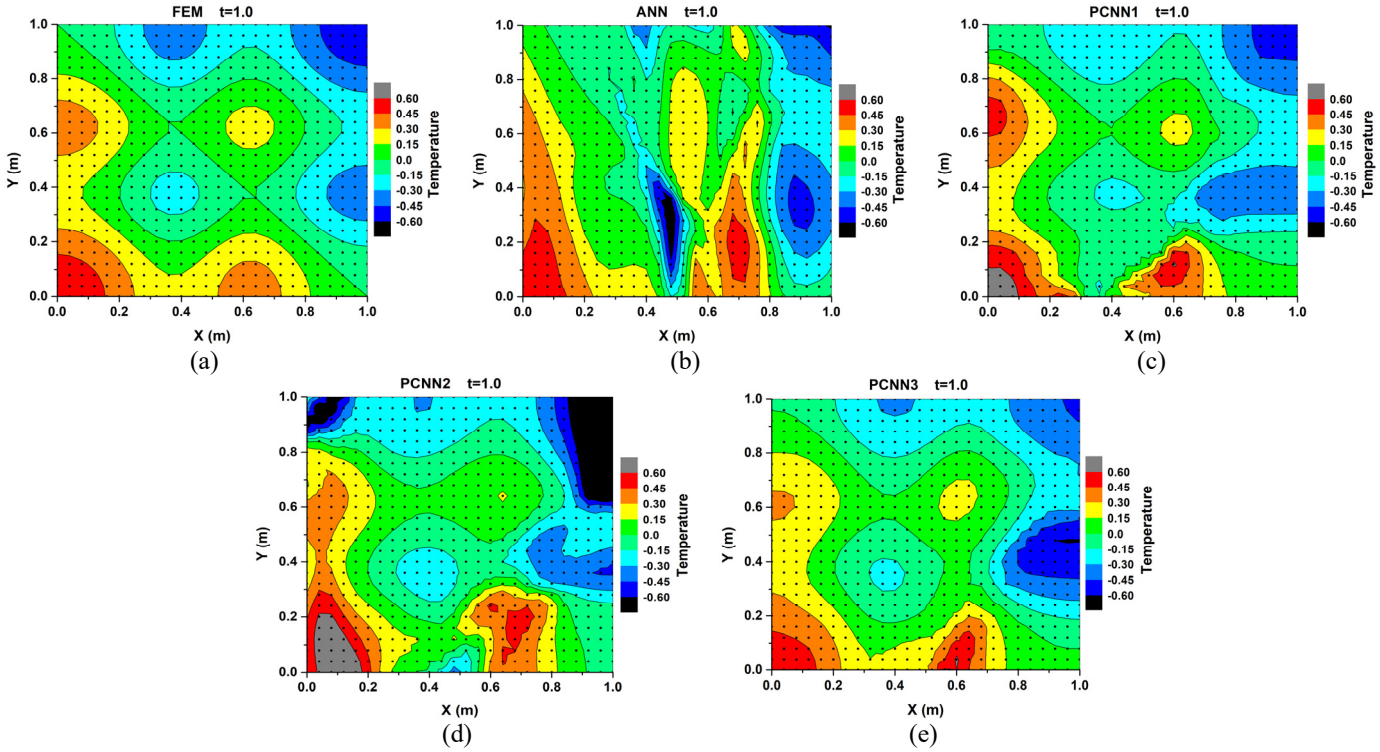


FIGURE 1: The predicted temperature fields from different models at t = 1: (a) original FEM solution, (b) traditional ANN, (c) equally-weighted PCNN1, (d) unequally-weighted PCNN2, and (e) adaptively-weighted PCNN3.

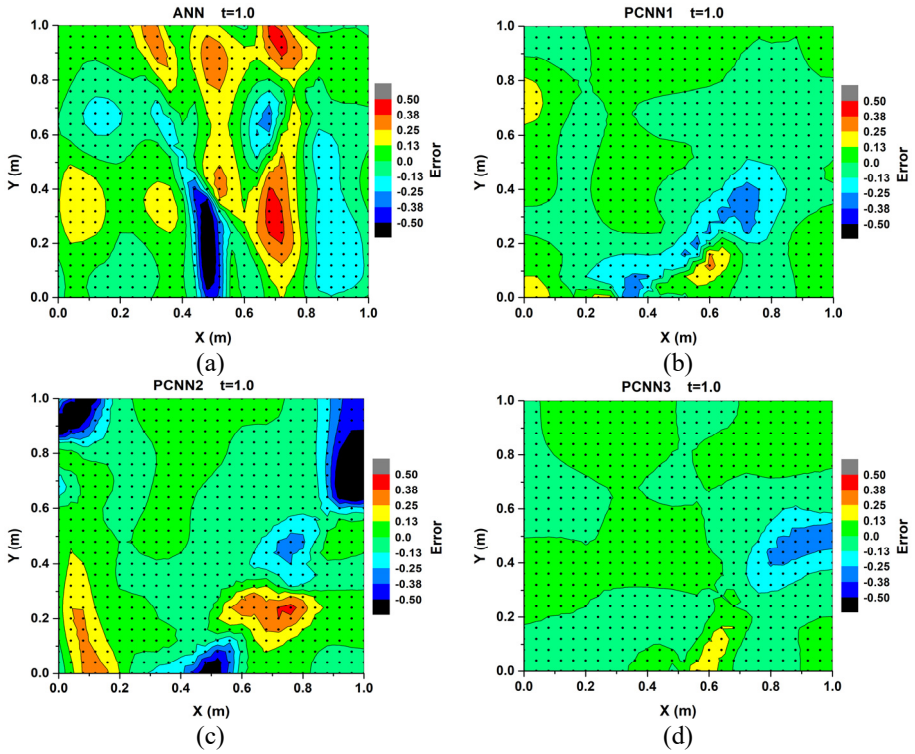
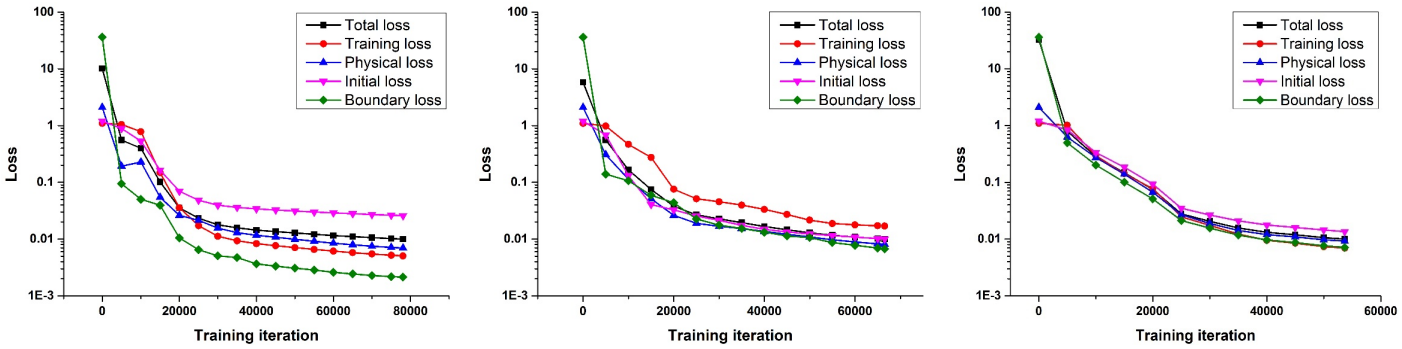


FIGURE 2: The errors of the predicted temperature fields compared to the FEM solution at t = 1: (a) traditional ANN, (b) equally-weighted PCNN1, (c) unequally-weighted PCNN2, and (d) adaptively-weighted PCNN3.



(a) (b) (c)

FIGURE 3: The learning curves for different PCNNs: (a) the equally-weighted PCNN1, (b) the unequally-weighted PCNN2, and (c) the adaptively-weighted PCNN3.

that the prediction from the ANN is less accurate than the three PCNNs, because of the small training data set. The error is especially large in the area around saddle points. Notice that the training data for the ANN and PCNNs come from the same LF simulations. With physical constraints added as regularization terms, the prediction errors of the PCNNs are reduced significantly.

The learning curves for different PCNNs are shown in FIGURE 3. For the three different PCNNs, all losses monotonically decrease during the training. However, the difference between the convergence speeds of individual losses varies with the different weighting schemes. For the equally-weighted PCNN1, as shown in FIGURE 3(a), the initial loss is one order of magnitude larger than the boundary loss, meaning that the difference between the convergence speeds of individual losses is large. Therefore, it takes a longer time for the PCNN1 to converge. For the unequally-weighted PCNN2, the weights of physical constraints are higher in order to increase the influence of prior knowledge. As a result, the different losses are within the same order of magnitude, as shown in FIGURE 3(b). As for the adaptively-weighted PCNN3, the weights are dynamically adjusted based on the percentages of individual losses in the total loss function. As shown in FIGURE 3(c), the different losses converged at the same speed and are well-balanced. The training time is the shortest among the three cases.

The quantitative comparison of training time and the mean squared error (MSE) of prediction for four neural networks is listed in TABLE 3. All MSEs of prediction for the PCNN1 and PCNN3 are almost one order of magnitude lower than that for the ANN. As a result of stronger enforcement for the physical constraints, the prediction accuracy of the PCNN2 is higher than that of the PCNN1 at $t = 0$. However, the MSE of prediction at $t = 1$ for the PCNN2 is larger than that of the PCNN1. This could be caused by the in-balance between different losses in the PCNN2. As shown in FIGURE 3(b), the training loss is still larger than the threshold value 0.01 when the training is finished, although the total loss as the weighted average has reached the threshold. The adaptively-weighted PCNN3 has all individual losses well-balanced and has the highest prediction accuracy. The PCNN3 also has the least training time among the three PCNNs. Notice that the computational time for training the PCNNs is much longer than that for the ANN, because additional information from physical knowledge is used in the training. In engineering applications, simulations, especially high-fidelity ones, are computationally expensive. Therefore, the simulation results as the training data are sparse. However, prior knowledge can be obtained without expensive computation, which can be regarded as the supplemental data for training.

The convergence speeds of the ANN and the adaptively-weighted PCNN3 with respect to the amount of training data are compared in FIGURE 4. It is shown that the required amount of training data to reach certain accuracy level of prediction at time $t = 1$ can be reduced by adding physical constraints. Here, the number of physical constraints of the PCNN3 is $21 \times 6 \times 6 = 756$. The prediction MSEs at $t = 1$ of both ANN and PCNN decrease when the training data size increases. The advantage of PCNN over ANN is obvious when the training data size is small. When the training data size is less than 400, the prediction accuracy can have nearly one order of magnitude difference. To reach the same accuracy level of 0.01, the ANN requires about 900 training data points, whereas the PCNN only needs about 300 training data points. As the training data size increases, the difference of prediction accuracy between the ANN and PCNN gradually reduces.

TABLE 3: Quantitative comparison for different neural networks to solve the heat equation

Neural network	Training time (second)	MSE of prediction at $t = 0$	MSE of prediction at $t = 1$
ANN	8.66	0.1998	0.0293
PCNN1	1475.40	0.0225	0.0079
PCNN2	1259.91	0.0125	0.0350
PCNN3	1019.07	0.0139	0.0055

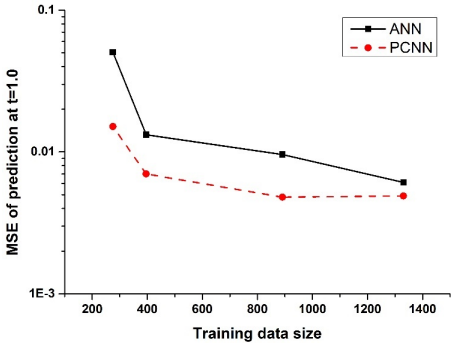


FIGURE 4: Convergence analysis for the ANN and the PCNN3.

3.2 Allen-Cahn equation

The second example is the Allen-Cahn equation, which is a nonlinear reaction-diffusion equation that describes the process of phase transition such as grain growth and spinodal decomposition. It has become the foundational model for the interface diffusion in the phase-field method, which is developed to study phase transitions and interfacial dynamics in materials science. The purpose of this example is to demonstrate the proposed MF-PCNN framework. The Allen-Cahn equation with periodic boundary condition in this example is

$$\begin{cases} u_t - 0.001\left(u_{xx} + u_{yy}\right) = u - u^3, \ \ t, x, y \in [0,1], \\ u(0,x,y) = 0.5\left[sin(4\pi x) + sin(4\pi y)\right], \\ u(t,0,y) = u(t,1,y), \\ u_x(t,0,y) = u_x(t,1,y), \\ u(t,x,0) = u(t,x,1), \\ u_y(t,x,0) = u_y(t,x,1). \end{cases} \quad ,(20)$$

where a non-conserved variable u is the order parameter or phase field.

Based on the results of the previous example, the weights of the physical constraints are adaptively adjusted as in Eq. (19). The physical loss is given by

$$E_P = \frac{1}{N_P}\sum_{i=1}^{N_P}\left|\begin{matrix} U_t\left(t_i^P, x_i^P, y_i^P\right) \\ -0.001U_{xx}\left(t_i^P, x_i^P, y_i^P\right) \\ -0.001U_{yy}\left(t_i^P, x_i^P, y_i^P\right) \\ -U\left(t_i^P, x_i^P, y_i^P\right) + U^3\left(t_i^P, x_i^P, y_i^P\right) \end{matrix}\right|^2. \quad (21)$$

The initial loss is given by

$$E_I = \frac{1}{N_I}\sum_{i=1}^{N_I}\left|\begin{matrix} U\left(0, x_i^I, y_i^I\right) \\ -0.5\left[sin\left(4\pi x_i^I\right) + sin\left(4\pi y_i^I\right)\right] \end{matrix}\right|^2. \quad (22)$$

The boundary loss is given by

$$E_S = \frac{1}{N_S}\sum_{i=1}^{N_S}\left[\begin{matrix} \left|U\left(t_i^S, 0, y_i^S\right) - U\left(t_i^S, 1, y_i^S\right)\right|^2 \\ +\left|U_x\left(t_i^S, 0, y_i^S\right) - U_x\left(t_i^S, 1, y_i^S\right)\right|^2 \\ +\left|U(t_i^S, x_i^S, 0) - U(t_i^S, x_i^S, 1)\right|^2 \\ +\left|U_y(t_i^S, x_i^S, 0) - U_y(t_i^S, x_i^S, 1)\right|^2 \end{matrix}\right]. \quad (23)$$

As shown in Eq. (12), the prediction of a MF-PCNN is a combination of the LF-PCNN prediction and the difference predicted by a ML model (DANN or GP). First, a low-cost LF-PCNN is trained during the time period $t \in [0,1]$ and then used as the baseline model. In addition, two high-cost HF-PCNNs (HF-PCNN1 and HF-PCNN2) are constructed. As shown in TABLE 2, the HF-PCNN1 is trained with data for the time period $t \in [0, 0.2]$, whereas the HF-PCNN2 is trained with the data for two time periods, $t \in [0, 0.2]$ and $t \in [0.8, 1]$. Then DANNs and GPs are trained to predict the differences between the LF-PCNN and HP-PCNN predictions during the time period $t \in [0, 1]$. The observed difference between the predictions of the LF-PCNN and HF-PCNN1 during the time period $t \in [0, 0.2]$ serves as the training data for the DANN1, DANN2, and GP1. The network structure of DANN2 is more complex than DANN1. Similarly, the observed difference between the predictions of the LF-PCNN and HF-PCNN2 for two time periods, $t \in [0, 0.2]$ and $t \in [0.8, 1]$, serves as the training data for the DANN3, DANN4, and GP2. Finally, the prediction of the MF-PCNN is the sum of the LF-PCNN prediction and the predicted difference by DANNs or GPs. In this example, the mean square of the difference between the LF simulation and HF simulation is 0.0001 during the time period $t \in [0, 0.2]$. However, since a coarser mesh and larger time step is used in the LF simulation, errors are accumulated over time. Then, the mean square of the difference between the LF simulation and HF simulation becomes 0.0029 during the time period $t \in [0.8, 1.0]$. Therefore, LF simulations are less accurate than HF simulations in the later stage. It is necessary and useful to adopt the MF-PCNN framework to fully utilize the training data with different fidelity.

The predictions of the phase field from different models, including traditional ANN, LF-PCNN, multi-fidelity models (combinations of LF-PCNN and DANNs, as well as LF-PCNN and GPs), at time $t = 0.5$ are shown in FIGURE 5. It is seen that the traditional ANN has larger prediction errors than PCNNs, especially at the saddle points where the true values are zeros. Adding physical constraints can significantly reduce the prediction errors, as in the LF-PCNN. At some saddle points, the phase field predicted by the LF-PCNN is still larger than zero, as shown in FIGURE 5(c). Compared to the LF-PCNN, the prediction errors of MF-PCNNs can be further reduced by adding the prediction of the difference from DANNs or GPs. As shown in FIGURE 5(d-i), the phase field predicted by the MF-PCNNs is almost zero at all saddle points. The difference between the predictions of the LF-PCNN and HF-PCNN can be captured by DANNs or GPs very well.

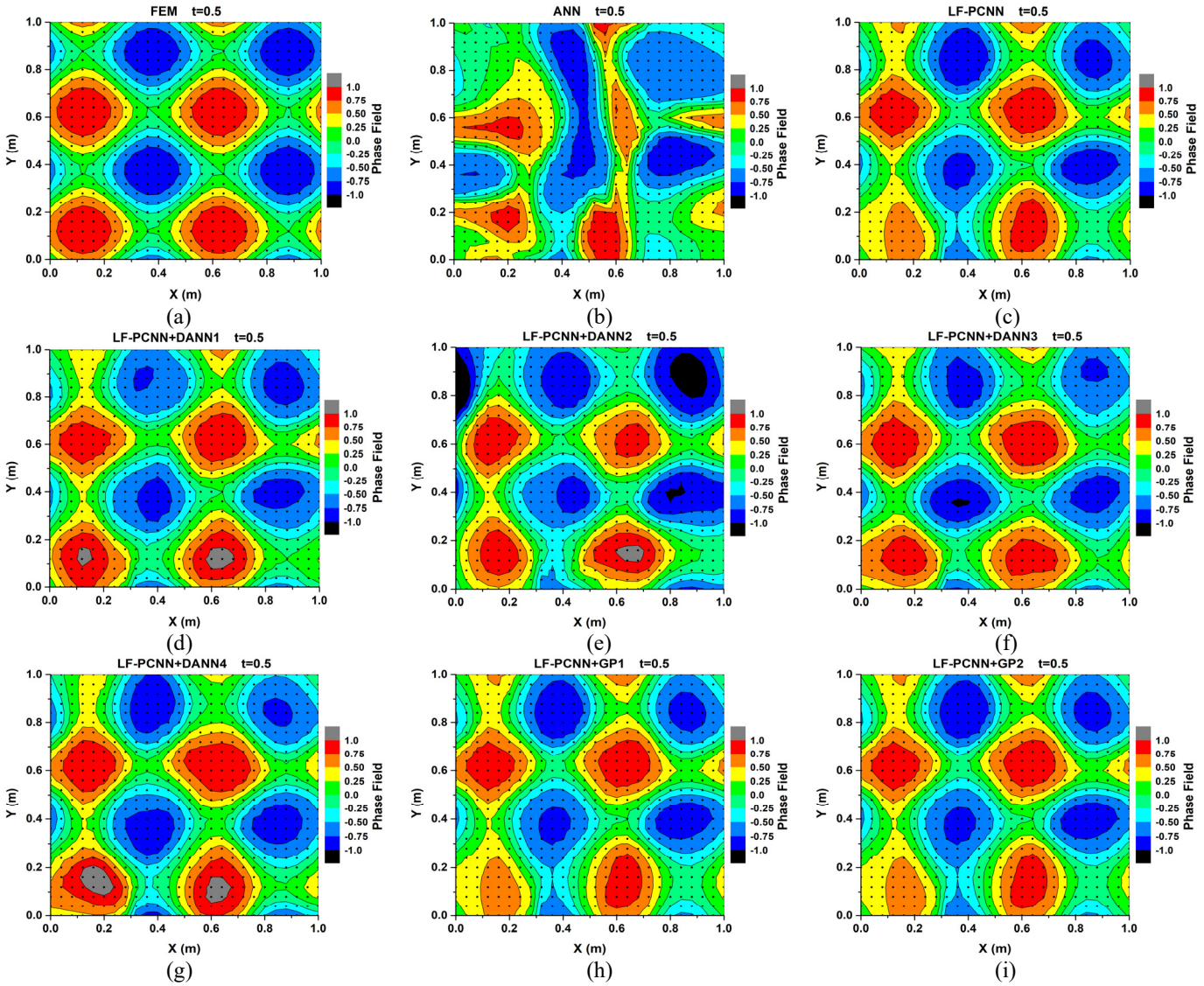


FIGURE 5: The predicted phase fields from different models at $t = 0.5$.

The quantitative comparisons of training time and the MSE of prediction for different ML models to solve the Allen-Cahn equation are listed in TABLE 4, where a MF-PCNN is composed of a LF-PCNN and a ML model to predict the difference. For example, MF-PCNN1 = LF-PCNN+DANN1 means that the MF-PCNN1 is a combination of the LF-PCNN and DANN1. The total training time of a MF-PCNN is the sum of training times for the LF-PCNN, HF-PCNN, and the difference ML model (DANN or GP). It is noted that the prediction of the HF-PCNN1 is used in the training of the MF-PCNN1, MF-PCNN2, and MF-PCNN3, whereas the prediction of the HF-PCNN2 is used in the training of the rest of the MF-PCNNs. Therefore, the training times of the MF-PCNN4, MF-PCNN5, and MF-PCNN6 are longer because of more training data and physical constraints. The training time of the MF-PCNNs with GPs is longer than that of the MF-PCNNs with DANNs because GPs are computationally more expensive.

The MSEs of predictions at different simulated time periods for different ML models are shown in FIGURE 6. In general, the MSE of prediction increases over time for different ML models except the MF-PCNN3 and MF-PCNN6. Since the prediction of the phase field relies on the previous predictions, the error will be accumulated over time. It is noted that the time period $t \in [1, 2]$ is outside the time range $t \in [0, 1]$ of LF training data for the LF-PCNN. Therefore, the error for extrapolation is larger, which is a common issue for most ML models. Nevertheless, the MSEs of extrapolation for the LF-PCNN, MF-PCNN1 and MF-PCNN4 are one order of magnitude lower than that of the ANN.

TABLE 4: Quantitative comparison between different ML models to solve the Allen-Cahn equation

ML model	Training time (second)	MSE of prediction at $t = 0.5$	MSE of prediction at $t = 1.5$
ANN	7.93	0.2215	0.8866
LF-PCNN	774.32	0.0258	0.0684
MF-PCNN1=LF-PCNN+DANN1	774.32+324.37+79.52=1178.21	0.0133	0.0521
MF-PCNN2=LF-PCNN+DANN2	774.32+324.37+25.19=1123.88	0.0753	0.8508
MF-PCNN3=LF-PCNN+GP1	774.32+324.37+1433.66=2532.35	0.0258	0.0684
MF-PCNN4=LF-PCNN+DANN3	774.32+3095.68+100.38=3970.38	0.0114	0.0399
MF-PCNN5=LF-PCNN+DANN4	774.32+3095.68+58.01=3928.01	0.0173	0.1926
MF-PCNN6=LF-PCNN+GP2	774.32+3095.68+10730.40=14600.40	0.0258	0.0684

The MSE of prediction from the MF-PCNN1 is significantly lower than that of the LF-PCNN for $t \in [0, 1]$. The difference between the MSEs however decreases for $t \in [1, 2]$. Furthermore, the MSE of prediction at $t = 0.5$ for the MF-PCNN1 is decreased by about 50%, compared with that of the LF-PCNN. As for the MF-PCNN2, its MSE of prediction is higher than that of LF-PCNN when $t > 0.5$. The MSE of prediction for the MF-PCNN2 is almost the same as that of the ANN when $t > 0.75$. The increased MSE for the MF-PCNN2 is caused by overfitting since the DANN2 has more neurons than the DANN1. The MSE of prediction at $t = 0$ for the MF-PCNN3 is slightly lower than that of the LF-PCNN. However, the MSE of prediction at $t = 0.25$ for the MF-PCNN3 is larger that of the LF-PCNN. The MSE of prediction for the MF-PCNN3 becomes the same as that of the LF-PCNN when $t > 0.5$, which means that the GP1 fails to predict the differences and its output is zero. Notice that $t = 0.5$ is outside the time range $t \in [0, 0.2]$ of the HF training data for the HF-PCNN1. The prediction is based on extrapolation. The errors indicate that DANNs are more robust than GPs for extrapolation.

With more training data and physical constraints, the HF-PCNN2 has two sampling spaces in the temporal dimension, which are [0, 0.2] and [0.8, 1]. The observed difference between the predictions of the LF-PCNN and HF-PCNN2 is served as the training data for the DANN3, DANN4, and GP2. Therefore, the prediction of the difference between the LF-PCNN and HF-PCNN at t = 0.5 has become an interpolation problem. Compared to the MF-PCNN1, the MSE of prediction for the MF-PCNN4 is the lowest among all ML models for the most of the time. With more training data, the MSE of prediction for the MF-PCNN5 is lower than that of the MF-PCNN2. However, the MSE of prediction for the MF-PCNN5 becomes higher than that of the MF-PCNN4 when $t > 0.25$ because of the overfitting. Compared to the MF-PCNN3, the MSE of prediction for the MF-PCNN6 is reduced with more training data when $t \in [0.5, 1.25]$. However, the MSE of prediction for the MF-PCNN6 becomes the same as that of the LF-PCNN when $t > 1.25$, indicating the failure of prediction by GP2.

Among all ML models in this work, the MF-PCNN1 is the best one in comprehensive performance since it has a relatively low training time and very good accuracy. The good generalization of the MF-PCNN1 comes from the simpler neural network structure of the DANN1.

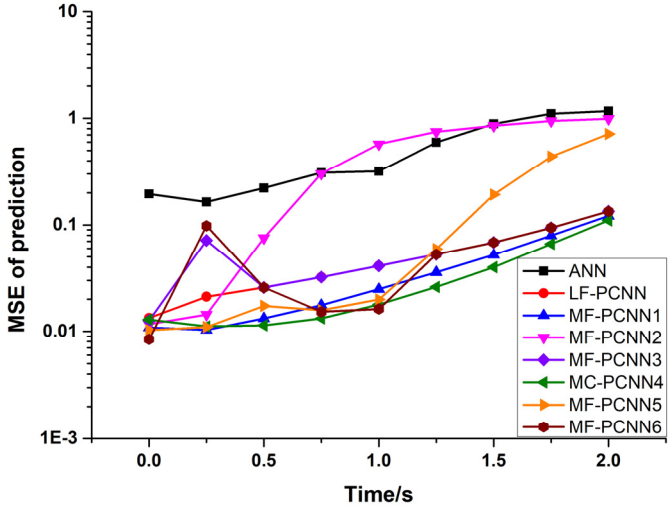


FIGURE 6: The change of MSE of prediction for different ML models.

4. CONCLUSION

In this work, a new scheme of multi-fidelity physics-constrained neural networks is proposed to improve the efficiency of training in neural networks by reducing the required amount of training data and incorporating physical knowledge as constraints. Neural networks with two (or more) levels of fidelities are combined to improve the prediction accuracy. Low-fidelity networks predict the general trend, whereas high-fidelity networks model local details and fluctuations. For the concern of training cost, low-fidelity networks can be trained with low-fidelity data, and the prediction accuracy can be further improved with supplementary high-fidelity data. Thus, the training efficiency is improved from two aspects. The first one is the

guidance from the physical knowledge, and the second one is a more cost-effective data collection and sampling strategy.

The physical knowledge can be easily added as the regularization terms into the total loss functions in neural networks. The physical constraints then can help reduce the searching space and guide the searching direction during the training. The proposed formulation is generic and can be extended to other machine learning approaches, where regularization can be similarly applied.

The proposed scheme is demonstrated with two examples of materials modeling. The PCNN is effective for these two different types of PDEs with different boundary conditions. The classical ANN with small training data sets tends to have large prediction errors. By adding physical constraints, the prediction accuracy of the PCNN can be one order of magnitude higher than the one from the classical ANN. Even with limited training data, the prediction of the PCNN is comparable with the original FEM solution. The weights associated with physical constraints can be adjusted to reflect the importance of the prior knowledge. They also affect the prediction accuracy. It is demonstrated that the adaptive weighting scheme results in higher prediction accuracy and shorter training time because the different losses in the total cost function are well balanced and have a similar convergence speed. The convergence analysis shows that the required amount of training data can be reduced by adding more physical constraints. Based on the computational results, DANNs are more capable than GPs to do the extrapolation of the difference between the LF-PCNN and HF-PCNN.

The developed MF-PCNN is an efficient approach to predict unknown relationships by combining the information from physical knowledge and available data. The training efficiency can be significantly improved if the training data from numerical simulations with different fidelities are utilized to construct MF-PCNNs. The training data are not limited to numerical simulation results only. They can also come from experimental measurements. The costs of experimental measurements can also be incorporated into the multi-fidelity scheme, where cost-effective sampling strategies can be taken.

The potential improvement of the current PCNN could be replacing the ANN to be the Recurrent Neural Network (RNN), such as long short-term memory (LSTM) neural network. Unlike feedforward neural networks, RNNs can use their internal state to process sequences of inputs, which may be more appropriate to solve time-dependent problems.

The proposed scheme should not be regarded as the replacement of classical numerical simulation methods (e.g. finite element and spectral methods) for solving partial differential equations. Rather, it enhances the efficiency of engineering design when high-fidelity simulations need to be run repetitively to obtain samples for design optimization. The number of samples for optimization for high-dimensional problem usually is very large. The machine learning approach therefore only shows its advantage for complex problems with high-dimensional searching space with the cost of training justified. The proposed scheme has the potential of making machine learning useful for real-world engineering applications where data sparsity is a common issue.

ACKNOWLEDGEMENTS

This work is supported in part by the George W. Woodruff Faculty Fellowship at Georgia Institute of Technology.

REFERENCES

[1] Karpatne, A., Atluri, G., Faghmous, J. H., Steinbach, M., Banerjee, A., Ganguly, A., Shekhar, S., Samatova, N., and Kumar, V., 2017, "Theory-Guided Data Science: A New Paradigm for Scientific Discovery from Data," IEEE Trans. Knowl. Data Eng., **29**(10), pp. 2318–2331.

[2] Li-Zhi, L., and Hou-Duo, Q., 1999, "A Neural Network for the Linear Complementarity Problem," Math. Comput. Model., **29**(3), pp. 9–18.

[3] Xia, Y., Leung, H., and Wang, J., 2002, "A Projection Neural Network and Its Application to Constrained Optimization Problems," IEEE Trans. Circuits Syst. I Fundam. Theory Appl., **49**(4), pp. 447–458.

[4] Thompson, M. L., and Kramer, M. A., 1994, "Modeling Chemical Processes Using Prior Knowledge and Neural Networks," AIChE J., **40**(8), pp. 1328–1340.

[5] Watson, P. M., Gupta, K. C., and Mahajan, R. L., 1998, "Development of Knowledge Based Artificial Neural Network Models for Microwave Components," 1998 IEEE MTT-S Int. Microw. Symp. Dig. (Cat. No.98CH36192), **1**, pp. 9–12.

[6] Wang, F., 1997, "Knowledge-Based Neural Models for Microwave Design," IEEE Trans. Microw. Theory Tech., **45**(12 PART 2), pp. 2333–2343.

[7] Tresp, V., Hollatz, J., and Ahmad, S., 1993, "Network Structuring and Training Using Rule-Based Knowledge," Adv. Neural Inf. Process. Syst. 5, pp. 871–878.

[8] Towell, G. G., and Shavlik, J. W., 1994, "Knowledge-Based Artificial Neural Networks," Artif. Intell., **70**(1–2), pp. 119–165.

[9] Ramuhalli, P., Udpa, L., and Udpa, S. S., 2005, "Finite-Element Neural Networks for Solving Differential Equations," IEEE Trans. Neural Networks, **16**(6), pp. 1381–1392.

[10] Xu, C., Wang, C., Ji, F., and Yuan, X., 2012, "Finite-Element Neural Network-Based Solving 3-D Differential Equations in Mfl," IEEE Trans. Magn., **48**(12), pp. 4747–4756.

[11] Han, F., and Huang, D. S., 2008, "A New Constrained Learning Algorithm for Function Approximation by Encoding a Priori Information into Feedforward Neural Networks," Neural Comput. Appl., **17**(5–6), pp. 433–439.

[12] Lauer, F., and Bloch, G., 2008, "Incorporating Prior Knowledge in Support Vector Regression," Mach. Learn., **70**(1), pp. 89–118.

[13] Dissanayake, M. W. M. G., and Phan-Thien, N., 1994, "Neural-network-based Approximations for Solving

Partial Differential Equations," Commun. Numer. Methods Eng., **10**(3), pp. 195–201.
[14] Raissi, M., Perdikaris, P., and Karniadakis, G. E., 2018, "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations," J. Comput. Phys., **378**, pp. 686–707.
[15] de Cursi, J. E. S., and Koscianski, A., 2007, "Physically Constrained Neural Network Models for Simulation," Adv. Innov. Syst. Comput. Sci. Softw. Eng., pp. 567–572.
[16] Shirvany, Y., Hayati, M., and Moradian, R., 2009, "Multilayer Perceptron Neural Networks with Novel Unsupervised Training Method for Numerical Solution of the Partial Differential Equations," Appl. Soft Comput. J., **9**(1), pp. 20–29.
[17] Lagaris, I. E., Likas, A., and Fotiadis, D. I., 1998, "Artificial Neural Networks for Solving Ordinary and Partial Differential Equations," IEEE Trans. Neural Networks, **9**(5), pp. 987–1000.
[18] Shekari Beidokhti, R., and Malek, A., 2009, "Solving Initial-Boundary Value Problems for Systems of Partial Differential Equations Using Neural Networks and Optimization Techniques," J. Franklin Inst., **346**(9), pp. 898–913.
[19] Lagaris, I. E., Likas, A. C., and Papageorgiou, D. G., 2000, "Neural-Network Methods for Boundary Value Problems with Irregular Boundaries," IEEE Trans. Neural Networks, **11**(5), pp. 1041–1049.
[20] McFall, K. S., and Mahan, J. R., 2009, "Artificial Neural Network Method for Solution of Boundary Value Problems With Exact Satisfaction of Arbitrary Boundary Conditions," IEEE Trans. Neural Networks, **20**(8), pp. 1221–1233.
[21] Malek, A., and Shekari Beidokhti, R., 2006, "Numerical Solution for High Order Differential Equations Using a Hybrid Neural Network-Optimization Method," Appl. Math. Comput., **183**(1), pp. 260–271.
[22] Bellamine, F., Almansoori, A., and Elkamel, A., 2015, "Modeling of Complex Dynamic Systems Using Differential Neural Networks with the Incorporation of a Priori Knowledge," Appl. Math. Comput., **266**, pp. 515–526.
[23] Ferrari, S., and Jensenius, M., 2008, "A Constrained Optimization Approach to Preserving Prior Knowledge during Incremental Training," IEEE Trans. Neural Networks, **19**(6), pp. 996–1009.
[24] Di Muro, G., and Ferrari, S., 2008, "A Constrained-Optimization Approach to Training Neural Networks for Smooth Function Approximation and System Identification," Proc. Int. Jt. Conf. Neural Networks, pp. 2353–2359.
[25] Rudd, K., Muro, G. Di, and Ferrari, S., 2014, "A Constrained Backpropagation Approach for the Adaptive Solution of Partial Differential Equations," IEEE Trans. Neural Networks Learn. Syst., **25**(3), pp. 571–584.
[26] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S., and Corrado, A. Davis, J. Dean, M. Devin, et al., 2016, "TensorFlow: A System for Large-Scale Machine Learning.," 12th USENIX Symp. Oper. Syst. Des. Implement.
[27] Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M., 2015, "Automatic Differentiation in Machine Learning: A Survey," J. Mach. Learn. Res., **18**, pp. 1–43.
[28] Lee, D., and Myung, K., 2017, "Read My Lips, Login to the Virtual World," 2017 IEEE Int. Conf. Consum. Electron. ICCE 2017, pp. 434–435.