Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

Multi-Fidelity Physics-Constrained Neural Network and Its Application in Materials Modeling

Dehao Liu, Yan Wang*

Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA

*Corresponding author. Email: yan.wang@me.gatech.edu Tel: +1-404-894-4714.

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 physics-constrained neural network 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, phase transition, and dendritic growth 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 the 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

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

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 many 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. For instance, recently a convolutional deep belief network [1] was used to extract microstructural features from material images. Generative adversarial networks were applied to generate synthetic images of material microstructures [2]. However, these traditional neural networks are data-hungry, which required large data sets (e.g. microstructural images) to train. Data sparsity is the bottleneck for applying the state-of-the-art ML techniques in the domains of engineering, where establishing high-dimensional process-structure-property relationships for either product or process design is the essential task. Transfer learning [3] is an approach to reduce the required amount of training data for a target domain from the model previously or concurrently trained in a source domain where data available, by reusing the source data (instance transfer) [4,5], previously trained features or basis functions (feature representation transfer) [6,7], or previously trained parameters

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

(parameter transfer) [8,9]. It has been applied in engineering, e.g., by training neural networks based on simulation data first and subsequently adjusting them to experimental data [10]. Transfer learning can be successful without much training in the target domain if the source and target domains are very similar or the same. However, the premise is that there are enough data available to train the model in the source domain, as it is still a pure data-driven approach. 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.

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 [11]. 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 used to predict the behaviors of physical systems. If trained properly, ML tools can be applied to predict physical properties of systems much more efficiently than traditional simulations [12]. Traditional simulation-

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

based design optimization process, which usually requires a large number of iterative simulation runs during the search, can be potentially accelerated by using the ML predictions instead.

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 [13,14]. Some efforts have been made for incorporating prior knowledge into ANNs in order to improve the training efficiency or prediction accuracy. Here, the training efficiency means the convergence speed. For instance, prior knowledge can be applied as preprocessing tools to filter training data [15,16], or embedded as some analytical input-output functions in additional layers of ANNs [17,18], 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 [19,20]. Similarly, finite-element neural networks (FENNs) [21,22] 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 [23]. In addition to functional values, the information of derivatives has also been incorporated as prior knowledge for support vector regression [24]. 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 [25]. 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 [26]. A regularization parameter has been introduced to control the trade-off between data fitting and knowledge-based regularization [27]. 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 [28]. 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 [29,30]. 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 [31] 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 [32] 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 [33], and a comparison was conducted between the performance of four different ANNs to solve ODEs [34]. 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 [35–37]. The effectiveness of regularization during

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

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 work, a multi-fidelity physics-constrained neural network is proposed. The concept of multi-fidelity has been explored extensively in surrogate modeling, particularly Gaussian process (GP) or co-kriging. By integrating the data from high-fidelity (HF) and low-fidelity (LF) simulations, the tradeoff between efficiency and accuracy for metamodeling can be made [38–40]. Multi-fidelity Gaussian process modeling has also been applied in design optimization [41–43]. Here, the concept of multi-fidelity is introduced to ANNs for the first time. The two major contributions of this paper to ML methodology are as follows. First, a new scheme of multi-fidelity physics-constrained neural network (PCNN) is proposed to reduce the training cost by simultaneously incorporating physical knowledge as constraints and using data with different fidelities. Second, a new adaptive weighting scheme is proposed for regularization to control the convergence of individual losses associated with training data and different types of physical constraints so that a balance between data and physical knowledge is achieved.

Here, it is demonstrated that PCNNs can be constructed to approximate the solutions of PDEs to predict the dynamic properties of systems. Some solutions from the simulations serve as the training data. The prior knowledge of PDEs, including 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 cost to obtain 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 than traditional ANNs. The LF-PCNN is trained with low-fidelity simulation results, whereas the HF-PCNN is trained from high-fidelity simulations. Then another ANN called discrepancy artificial neural network (DANN) is trained based on the difference between the LF-PCNN and HF-PCNN

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

predictions. The MF-PCNN is constructed by combining the predictions from the LF-PCNN and DANN. The advantage of the MF-PCNN is that the overall computational cost to obtain training data can be reduced by using the data with different fidelities. In this paper, three examples are used to demonstrate the MF-PCNN framework. The first example is the prediction of the temperature field in a heat transfer problem. The second example is the prediction of the phase field in a phase transition process. The third example is to predict the dendritic growth during solidification based on multiphysics simulations. It is shown that the MF-PCNN can be constructed with a limited amount of simulation data but 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.

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

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

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

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)

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

and boundary conditions (BCs)

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

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

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

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

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

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 derivative with respect to either time or space.

In this work, the MLP architecture is used as a demonstration, which includes one input layer $(t, \mathbf{x})$, multiple hidden layers, and one output layer $U(t, \mathbf{x})$ 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

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

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

$$E = \lambda_T E_T + \lambda_P E_P + \lambda_I E_I + \lambda_s E_s, \tag{8}$$

where

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

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,

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

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

and

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

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 the weights of different losses is given as

$$\lambda_T + \lambda_P + \lambda_I + \lambda_s = 1. \tag{9}$$

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. It will be shown in Section 3.1 that assigning different weights will affect the speed of training. An adaptive scheme to assign the weights is proposed here so that the overall loss is calculated as

$$E = \frac{E_T^2+E_P^2+E_I^2+E_S^2}{E_T+E_P+E_I+E_S}. \tag{10}$$

for each iteration during the training process. That is, the weights are proportional to individual losses from data and physical constraints respectively. 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

$$\delta(t, \mathbf{x}) = U_H(t, \mathbf{x}) - U_L(t, \mathbf{x}), \ t \in [0, T_0], \ \mathbf{x} \in \Omega. \tag{11}$$

Then the DANN is constructed to predict the discrepancy 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 discrepancy $\delta(t, \mathbf{x})$ as the training data to minimize the mean squared error loss

$$E_\delta = \frac{1}{N_\delta}\sum_{i=1}^{N_\delta}|U_\delta(t_i,\mathbf{x}_i) - \delta(t_i,\mathbf{x}_i)|^2,\ t \in [0,T_0],\ \mathbf{x} \in \Omega, \tag{12}$$

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 discrepancy $\delta(t,\mathbf{x})$ 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

$$U_M(t,\mathbf{x}) = U_L(t,\mathbf{x}) + U_\delta(t,\mathbf{x}),\ t \in [0,T],\ \mathbf{x} \in \Omega. \tag{13}$$

2.3 Experimental setup of the proposed MF-PCNN

The construction and training of the MF-PCNN are accomplished by using Tensorflow [44], 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 [45]. 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.

Three 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 proposed adaptive 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 MF-PCNN framework. The third example is the dendritic growth during solidification where heat transfer and phase transition are tightly coupled. The purpose is to demonstrate the applicability of the proposed MF-PCNN framework for complex multiphysics problems in materials design.

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

The details of the computational setup for different ML models in the heat transfer, phase transition and dendritic growth example are listed in Table 1, Table 2 and Table 3, 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 neural network architecture was identified by conducting some simple sensitivity studies. Finding the optimal architecture requires some systematic searching and sampling procedures, which can be done in future work. The structures of the tested DANNs are 5-5-5-5 and 10-10-10-10, which are simpler in order to avoid overfitting. For comparison purpose, two GP surrogate models with the RBF kernel are also constructed to predict the difference between the LF-PCNN and HF-PCNN. Only one run of the optimizer is performed from the RBF kernel's initial parameters. The noise level of the RBF kernel is set to be *alpha* = 0.1 to prevent overfitting. The hyperbolic tangent (tanh) function is used as the activation function. All of the loss functions in neural networks are minimized by using a gradient-based optimization algorithm called Adam [46].

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 discrepancy between the predictions of the LF-PCNNs and HF-PCNNs during the short time period $t \in [0, T_0]$. All FEM simulations are finished in less than one minute for these 2D problems. The training data and physical constraints for the first two examples are sampled uniformly in both temporal and spatial dimensions. Random sampling is used to obtain the LF and HF training data for the dendritic growth example.

Notice that in a multi-fidelity modeling framework, the LF data can come from LF models with lower resolutions, reduced-order models, models with simplified geometry, and others where the computational cost is lower than HF models. In this work, LF data were taken from the FEM simulations with low

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

resolutions. The proposed MF-PCNN does not require a fixed hierarchy of fidelities over the whole range of input parameters. That is, the LF and HF data do not form a nested hierarchy for both spatial and time domains.

2.3.1 Example 1: heat transfer

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

$$\begin{cases} u_t - 0.01\left(u_{xx} + u_{yy}\right) = 0, \;\; t, x, y \in [0,1], \\ u(0, x, y) = 0.5[sin(4\pi x) + sin(4\pi y)], \\ u_x(t, 0, y) = 0, \\ u_x(t, 1, y) = 0, \\ u_y(t, x, 0) = 0, \\ u_y(t, x, 1) = 0. \end{cases} , \tag{14}$$

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. In the total loss function defined by Eq. (8), the training loss here is given by

$$E_T = \frac{1}{N_T} \sum_{i=1}^{N_T} \left| U(t_i^T, x_i^T, y_i^T) - T(t_i^T, x_i^T, y_i^T) \right|^2. \tag{15}$$

The physical loss is

$$E_P = \frac{1}{N_P} \sum_{i=1}^{N_P} \left| U_t(t_i^P, x_i^P, y_i^P) - 0.01\left[U_{xx}(t_i^P, x_i^P, y_i^P) + U_{yy}(t_i^P, x_i^P, y_i^P)\right] \right|^2. \tag{16}$$

The initial loss is given by

$$E_I = \frac{1}{N_I} \sum_{i=1}^{N_I} \left| U(0, x_i^I, y_i^I) - 0.5[sin(4\pi x_i^I) + sin(4\pi y_i^I)] \right|^2. \tag{17}$$

The boundary loss is

$$E_S = \frac{1}{N_S}\sum_{i=1}^{N_S}\left[\left|U_x\left(t_i^S,0,y_i^S\right)\right|^2 + \left|U_x\left(t_i^S,1,y_i^S\right)\right|^2 + \left|U_y\left(t_i^S,x_i^S,0\right)\right|^2 + \left|U_y\left(t_i^S,x_i^S,1\right)\right|^2\right]. \quad (18)$$

As shown in Table 1, the amount of training data for the heat transfer example is $N_T = 21 \times 6 \times 6$, which means that there are 21 sampling points in the temporal dimension or time period, 6 sampling points in the x direction, and 6 sampling points in the y direction of the spatial domain. The simulation domain is $x, y \in [0,1]$ and time period is $t \in [0,1]$. The training data for the heat transfer example are from the FEM simulation where the grid spacing is $\Delta x = 0.2$ and the time step is $\Delta t = 0.05$. For the PCNNs in the heat transfer example, the number of physical constraints is $41 \times 11 \times 11 = 4961$. The grid spacing is $\Delta x = 0.1$ and the time step is $\Delta t = 0.025$ for physical constraints. The numbers of sampling points corresponding to the physical loss, initial loss, and boundary loss are $N_P = 3240$, $N_I = 121$ and $N_S = 1600$ respectively, which sum up to 4961. In the heat transfer example, three different weighting schemes (PCNN1, PCNN2, and PCNN3) are compared. The training of ANN and PCNNs stops when the total loss E is lower than a threshold value 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]

2.3.2 Example 2: phase transition

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 Allen-Cahn equation with periodic boundary condition in this example is

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

$$\begin{cases} u_t - 0.001(u_{xx} + u_{yy}) = u - u^3, \ \ t, x, y \in [0,1], \\ u(0, x, y) = 0.5[sin(4\pi x) + sin(4\pi y)], \\ u(t, 0, y) = u(t, 1, y), \\ u_x(t, 0, y) = u_x(t, 1, y), \\ u(t, x, 0) = u(t, x, 1), \\ u_y(t, x, 0) = u_y(t, x, 1). \end{cases} , \tag{19}$$

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. (10). The training loss is given by

$$E_T = \frac{1}{N_T} \sum_{i=1}^{N_T} \left| U(t_i^T, x_i^T, y_i^T) - T(t_i^T, x_i^T, y_i^T) \right|^2. \tag{20}$$

The physical loss is given by

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

The initial loss is given by

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

The boundary loss is given by

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

In the phase transition example, two HF-PCNNs (HF-PCNN1 and HF-PCNN2) are trained as shown in Table 2. The simulation domain is $x, y \in [0,1]$ and time period is $t \in [0,1]$. The training data for the ANN and LF-PCNN are from the LF simulation where the grid spacing is $\Delta x = 0.2$ and the time step is $\Delta t = 0.05$. The training data for the HF-PCNNs are from the HF simulation where the grid spacing is $\Delta x = 0.05$ and the time step is $\Delta t = 0.025$. Therefore, the training data for the HF-PCNNs is more accurate and expensive than for the ANN and LF-PCNN. 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 of those for the HF-PCNN1. The observed discrepancy between the predictions of the LF-PCNN and HF-PCNN1 serves as the training data for the DANN1, DANN2, and GP1. Here, the amount of training data for the DANN1, DANN2, and GP1 is $9 \times 26 \times 26$, which means that the grid spacing is $\Delta x = 0.04$ and the time step is $\Delta t = 0.025$. The difference between the HF and LF simulation data is not used as the training data for the discrepancy function because they may not be measured at the same location or time step. That is, since the data are not in a nested hierarchy, the observed discrepancy is obtained from the neural network predictions. Similarly, the observed discrepancy between the predictions of the LF-PCNN and HF-PCNN2 serves as the training data for the DANN3, DANN4, and GP2. In this work, the difference between the HF simulation data and the prediction of LF-PCNN is not used as the training data for the discrepancy function. This is because that more accurate predictions can be obtained by adding physical constraints into the training of HF-PCNNs. Besides, the trained HF-PCNNs can provide more training data to the DANNs or GPs so that the constructed MF-PCNNs are more general and have better prediction accuracy. 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 2. The setup for different ML models in the phase transition 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]
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]

2.3.3 Example 3: dendritic growth

The third example is dendritic growth during solidification, where heat transfer and phase transition are coupled with each other. In this multiphysics problem, the heat equation and the Allen-Cahn equation need to be solved simultaneously to predict the evolution of dendritic growth. The coupled PDEs and corresponding boundary conditions for the dendritic growth example are

$$\begin{cases} 0.001p_t - 0.0001(p_{xx} + p_{yy}) = p(1-p)\left[p - 0.5 + \frac{0.9}{\pi}\tan^{-1}(10q_e - 10q)\right] \\ p(0,x,y) = exp\left(-\frac{x^2+y^2}{0.04}\right) \\ p_x(t,-2.5,y) = p_x(t,2.5,y) = p_y(t,x,-2.5) = p_y(t,x,2.5) = 0 \\ 0.001(q_t - q_{xx} + q_{yy}) = 0.001Kp_t \\ q(0,x,y) = 0 \\ q_x(t,-2.5,y) = q_x(t,2.5,y) = q_y(t,x,-2.5) = q_y(t,x,2.5) = 0 \\ t \in [0,1],\ x,y \in [-2.5,2.5] \end{cases}, \qquad (24)$$

where p is the phase field and q is the temperature field. The liquidus temperature q_e and latent heat K are materials dependent and are the design variables in materials design. That is, we need to find the best material compositions corresponding to the optimal values of these two variables so that the desirable dendritic growth behavior can be obtained. The evolutions of dendrites are sensitive to the liquidus temperature and latent heat of the materials [47]. The dendritic growth (which can be quantified by the growth speed, symmetry, secondary arm spacing, and other descriptors) affects the final mechanical, thermal, and other properties of solid crystals. Therefore, efficient ML predictions of dendritic growths help establish the process-structure-property relationship for materials design. In this simplified example, all variables in the coupled PDEs are dimensionless. A scaling factor of 0.001 is multiplied at both sides of the heat equation so that the magnitudes of the Allen-Cahn equation and the heat equation are in the same scale. The normalization procedure ensures the fast convergence of PCNNs.

In order to apply the proposed MF-PCNN framework to design optimization, the design variables q_e and K need to be included in the inputs of the PCNN. To be more specific, the input for the PCNN is

(t, x, y, q_e, K). The training data for phase field $P_T(t, x, y, q_e, K)$ and temperature field $Q_T(t, x, y, q_e, K)$ come from FEM simulations. Then the outputs of the PCNN are $P(t, x, y, q_e, K)$ and $Q(t, x, y, q_e, K)$, which approximate the true phase field p and temperature field q respectively. The training of the PCNN for design optimization will require multiple sets of design variables and FEM simulation runs. In this example, we only demonstrate the feasibility of multiphysics prediction. Only two sets of design variables and FEM simulation data corresponding to two samples of dendritic growth are used for the training of MF-PCNNs. For each dendritic growth sample, the design variables q_e and K have constant values only, and two MF-PCNNs with different number of physical constraints are tested. The complete framework of materials design based on the MF-PCNN will be demonstrated in future work.

Similar to the previous two examples, loss functions are formulated based on the PDEs as well as the initial and boundary conditions. The adaptive weighting scheme in Eq. (10) is used to maximize the overall reduction speed of the total loss. The prediction of the MF-PCNN is a combination of the LF-PCNN prediction and the DANN as in Eq. (13). The LF-PCNN is trained with coarse simulation data during the time period $t \in [0, 1]$, and the HF-PCNN is trained with the denser simulation data during the time period $t \in [0, 0.2]$. The observed discrepancy between the predictions of the LF-PCNN and HF-PCNN during the time period $t \in [0, 0.2]$ serves as the training data for the DANN.

In the dendritic growth example, the simulation domain is $x, y \in [-2.5, 2.5]$ and time period is $t \in [0,1]$. As listed in Table 3, the training data for the LF-PCNNs are sampled randomly from the FEM simulation, where the grid spacing is $\Delta x = 0.01$ and the time step is $\Delta t = 0.1$. The training data for the HF-PCNNs are sampled randomly from the FEM simulation, where the grid spacing is $\Delta x = 0.01$ and the time step is $\Delta t = 0.05$. More training data are used for the training of the HF-PCNNs to increase the prediction accuracy. Compared to the LF-PCNN1 and HF-PCNN1, the LF-PCNN2 and HF-PCNN2 have more physical constraints involved. The training of the LF-PCNNs and HF-PCNNs stops when the total

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

loss E is lower than a threshold value of 0.0001. Similarly, the training of the DANNs stops when the loss function E_δ is below 0.005. The threshold values are lower than those in the previous examples because more accurate predictions are needed to observe the complex dendritic shape.

Table 3. The setup for different ML models in the dendritic growth example

ML model	Structure	Amount of training data (t×x×y)	Number of physical constraints (t×x×y)	Time period/s
LF-PCNN1	30-20-30-20	2861 (random, $\Delta t = 0.1$)	21×21×21	[0, 1]
HF-PCNN1	30-20-30-20	3901 (random, $\Delta t = 0.05$)	5×21×21	[0, 0.2]
DANN1	5-5-5-5	9×51×51	0	[0, 0.2]
LF-PCNN2	30-20-30-20	2861 (random, $\Delta t = 0.1$)	11×41×41	[0, 1]
HF-PCNN2	30-20-30-20	3901 (random, $\Delta t = 0.05$)	3×41×41	[0, 0.2]
DANN2	5-5-5-5	9×51×51	0	[0, 0.2]

3. EXPERIMENTAL RESULTS

In this section, the results for the heat transfer, phase transition, and dendritic growth examples are shown. The heat transfer example is used to demonstrate the effectiveness of the proposed adaptive weighting scheme for the total loss function. A convergence analysis for the ANN and the PCNN is also conducted. The phase transition example is to demonstrate the performance of the MF-PCNN framework. The dendritic growth example is used to demonstrate the applicability of the proposed MF-PCNN framework for complex multiphysics problems in materials design.

3.1 Heat transfer example

To assess the sensitivity of weights, three weighting schemes of the total loss function are tested and compared with each other. In the PCNN1 listed in Table 1, the weights are equal and fixed in the total loss function

$$E = 0.25(E_T + E_P + E_I + E_s). \tag{25}$$

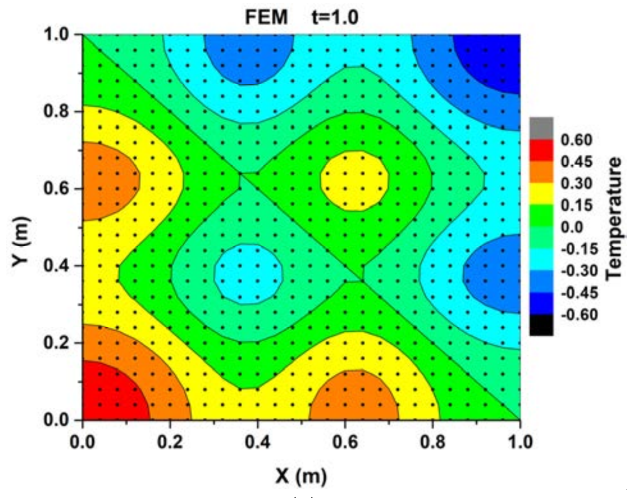
In the PCNN2, the weights are unequal and fixed in the total loss function

$$E = 0.125(E_T + 2E_P + 4E_I + E_s). \tag{26}$$

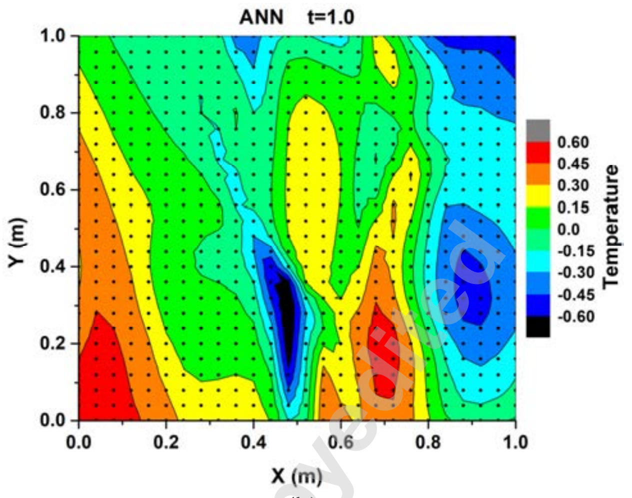
In the PCNN3, the weights are adaptive during the training as shown in Eq. (10). Assigning larger weights to the physical constraints indicates that prior knowledge will be more influential in the training process. When the training data are sparse, increasing the number of physical constraints can help 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 so 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. Fig. 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 Fig. 2, respectively. Here, the prediction error is the absolute difference between the prediction from the neural network and the FEM solution. The dots in the figures indicate the sampling positions in the 2D domain, where a total of 26×26 samples are taken. It is seen 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 the physical constraints added as regularization terms, the prediction errors of the PCNNs are reduced significantly.

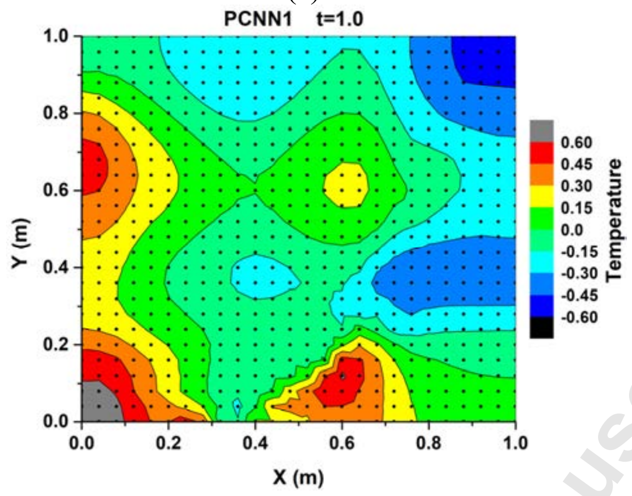
Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME



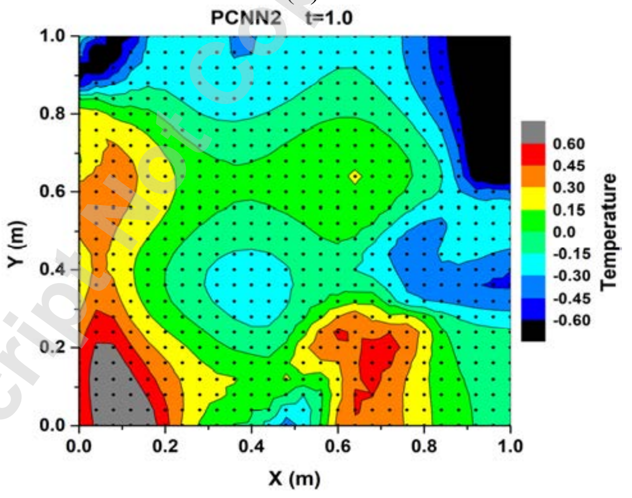
(a)



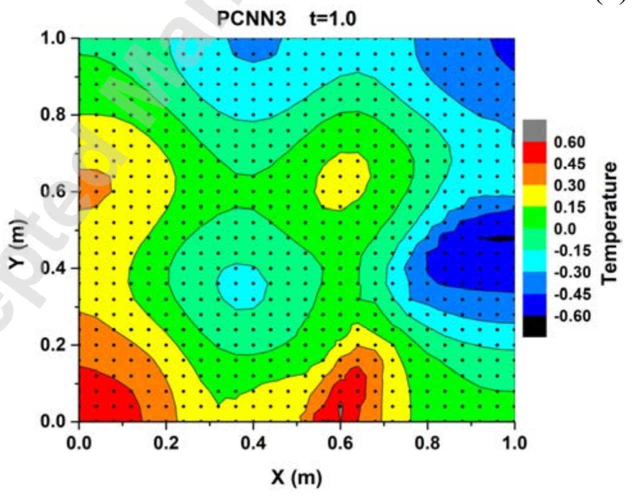
(b)



(c)



(d)



(e)

Fig. 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.

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

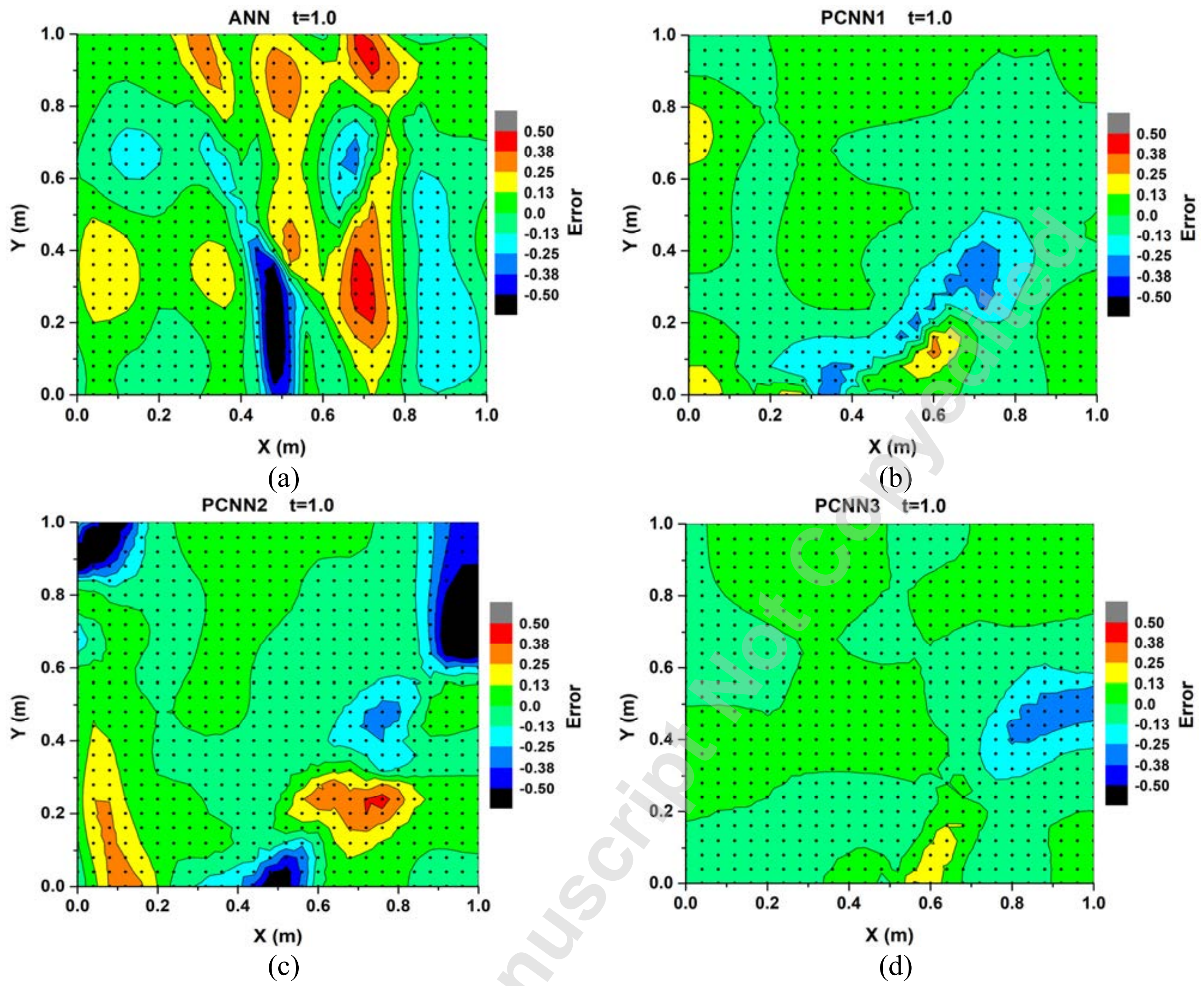
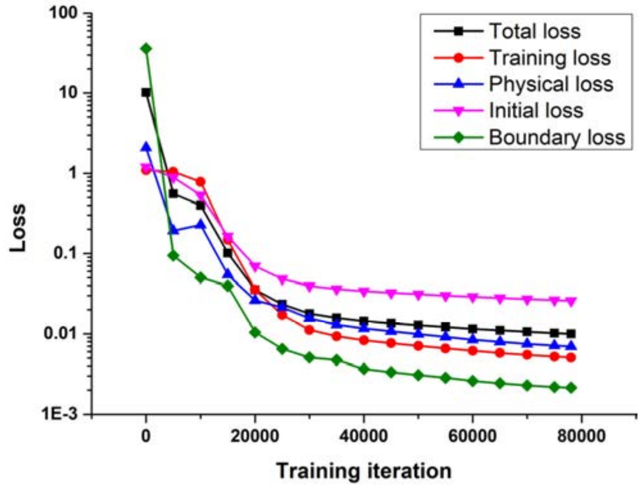


Fig. 2. The errors of the predicted temperature fields compared to the FEM solution at $t = 1$: (a) traditional ANN, (b) equally-weighted PCNN1, (c) unequally-weighted PCNN2, and (d) adaptively-weighted PCNN3.

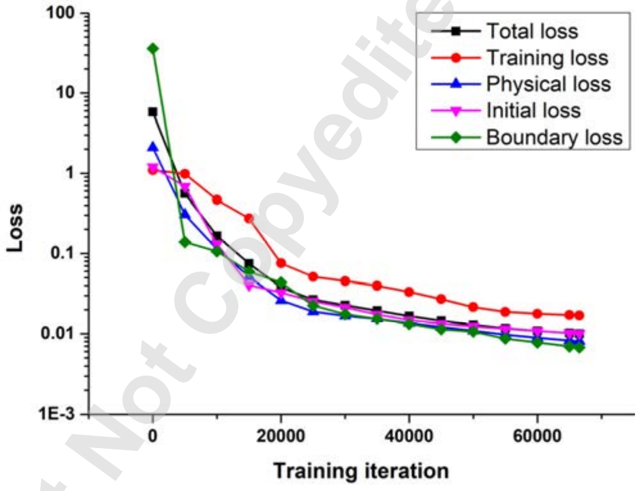
The learning curves for different PCNNs are shown in Fig. 3. For the three different PCNNs, all losses monotonically decrease during the training. However, the difference between the convergence speeds of individual losses varies with the different weighting schemes. For the equally-weighted PCNN1, as shown in Fig. 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

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

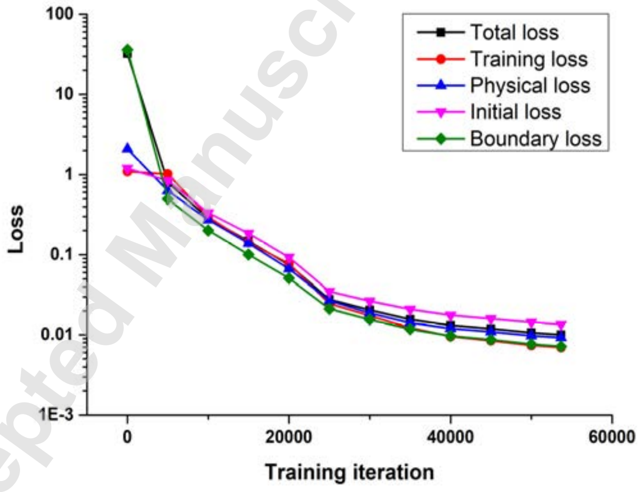
same order of magnitude, as shown in Fig. 3 (b). As for the adaptively-weighted PCNN3, the weights are dynamically adjusted based on the percentages of individual losses in the total loss function. As shown in Fig. 3 (c), the different losses converged at the same speed and are well-balanced. The training time is the shortest among the three cases.



(a)



(b)



(c)

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

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

The quantitative comparisons of training time and the mean squared error (MSE) of prediction for the four neural networks are listed in Table 4. 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 Fig. 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.

Table 4. 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

The convergence speeds of the ANN and the adaptively-weighted PCNN3 with respect to the amount of training data are compared in Fig. 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 the physical constraints. Here, the number of physical constraints of the PCNN3 is $21\times6\times6 = 756$. 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

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

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.

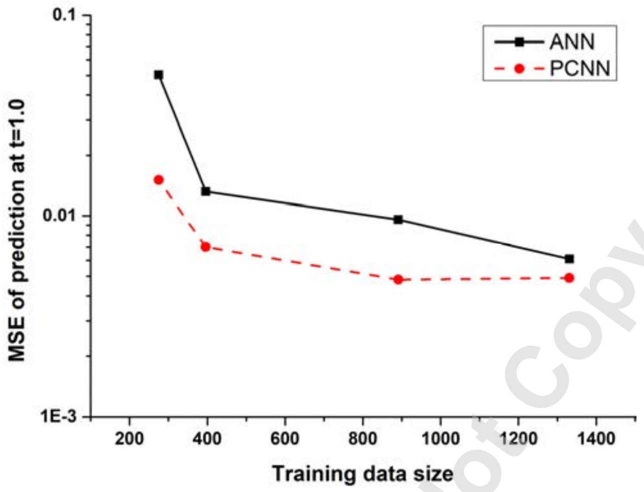


Fig. 4. Convergence analysis for the ANN and the PCNN3.

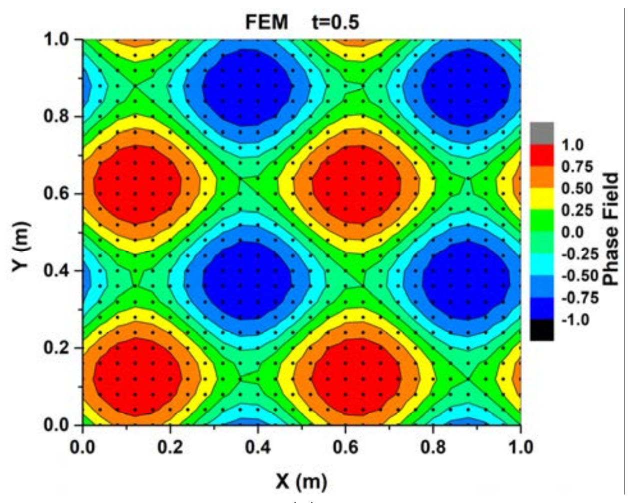
3.2 Phase transition example

As shown in Eq. (13), the prediction of a MF-PCNN is a combination of the LF-PCNN prediction and the discrepancy 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 discrepancy between the LF-PCNN and HP-PCNN predictions during the time period $t \in [0, 1]$. The observed discrepancy 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 discrepancy 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.

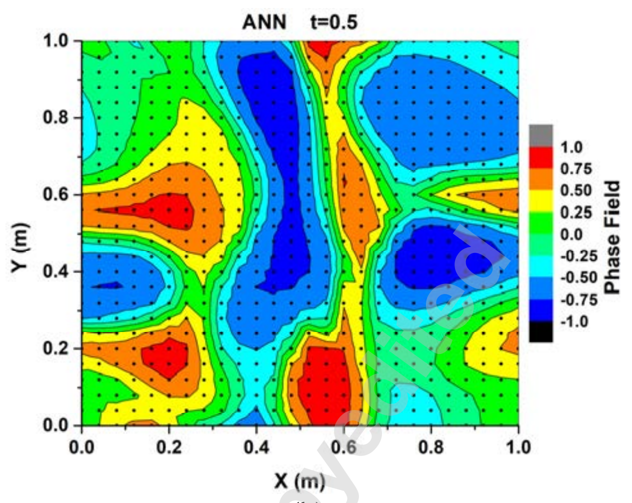
Finally, the prediction of the MF-PCNN is the sum of the LF-PCNN prediction and the predicted discrepancy by DANNs or GPs. In this example, the mean square of the difference between the LF simulation and HF simulation is 0.0001 during the time period $t \in [0, 0.2]$. However, since a coarser mesh and 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 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 Fig. 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 Fig. 5(c). Compared to the LF-PCNN, the prediction errors of MF-PCNNs can be further reduced by adding the discrepancy prediction from DANNs or GPs. As shown in Fig. 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.

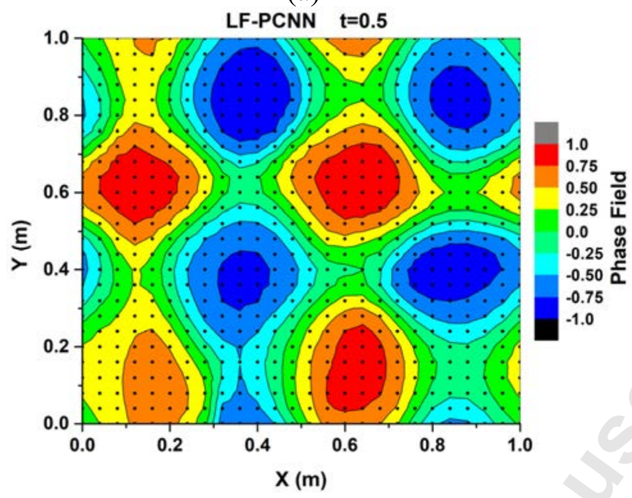
Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME



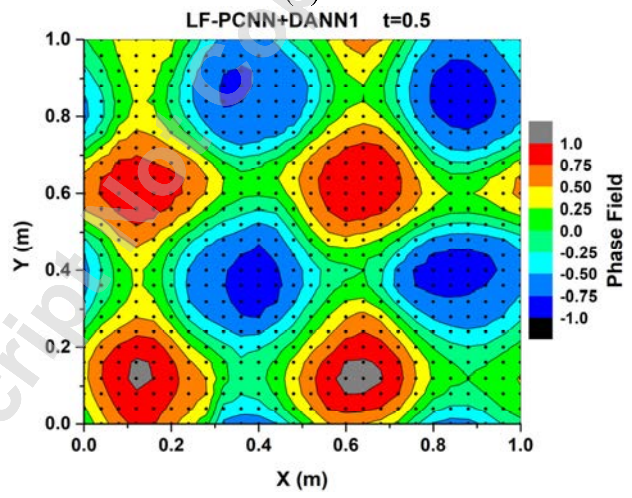
(a)



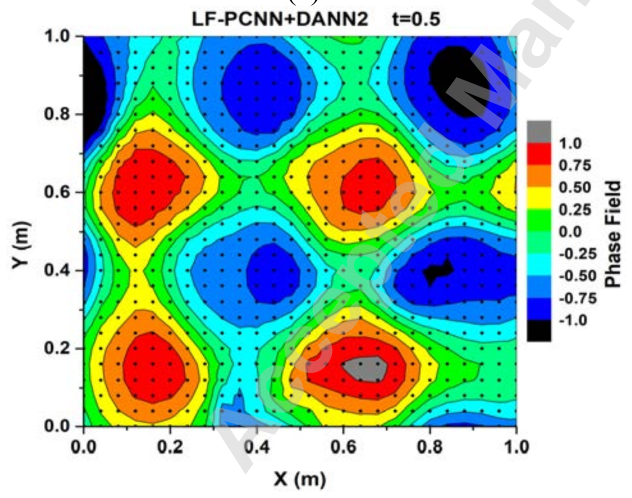
(b)



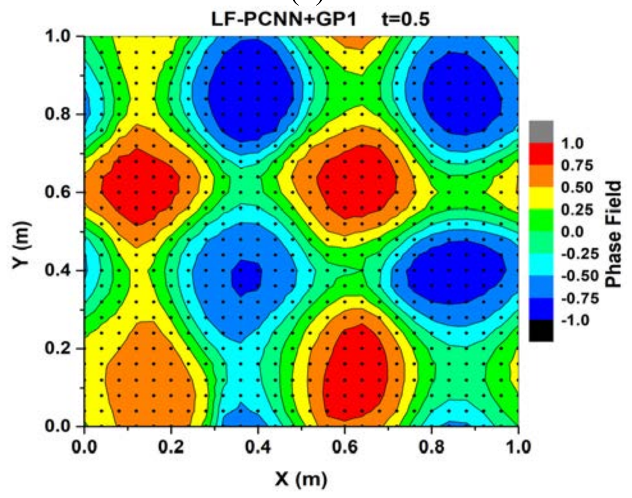
(c)



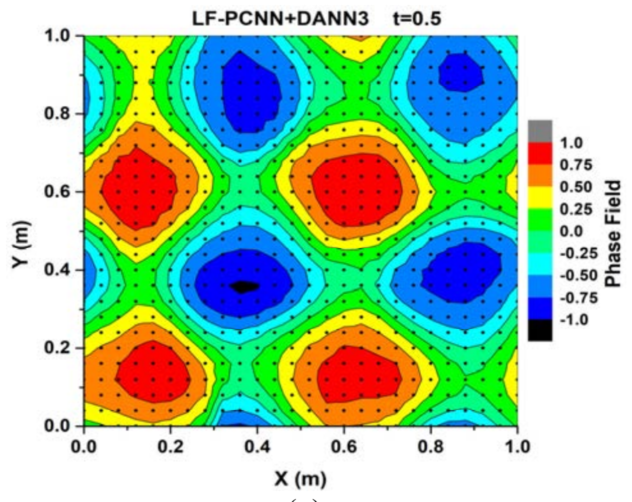
(d)



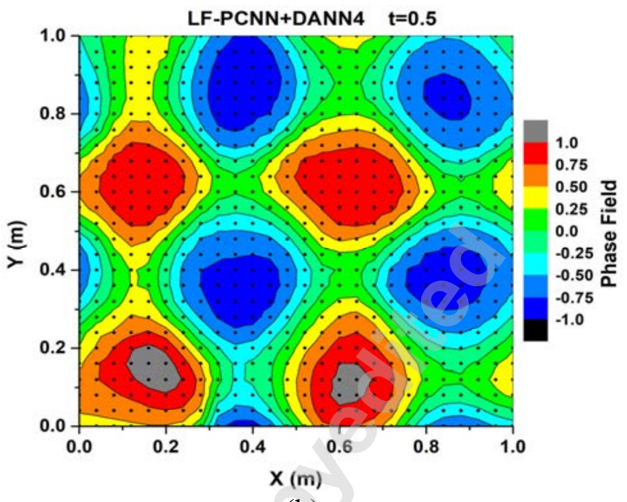
(e)



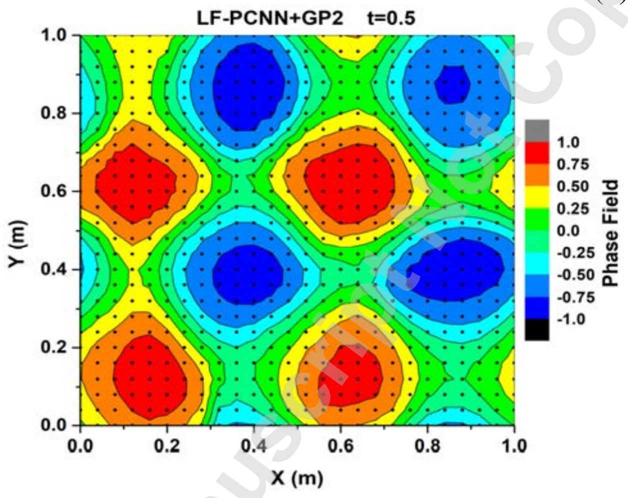
(f)



(g)



(h)



(i)

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

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

The quantitative comparisons of training time and the MSEs of prediction for different ML models to solve the Allen-Cahn equation are listed in Table 5, where a MF-PCNN is composed of a LF-PCNN and a ML model to predict the discrepancy. 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 discrepancy model (DANN or GP). The total training time of MF-PCNN1 is 774.32+324.37+79.52 = 1178.21 s, where the training times of the LF-PCNN, HF-PCNN1, and DANN1 are 774.32, 324.37, and 79.52 s, respectively. It is noted that the prediction of the HF-PCNN1 is used for the training of the MF-PCNN1, MF-PCNN2, and MF-PCNN3, whereas the prediction of the HF-PCNN2 is used for 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. It is noted that the training time of the GP1 is comparable to the DANNs, whereas the training time of the GP2 is one magnitude longer than the DANNs. The standard GP is computationally more expensive because the computation of inverse covariance matrices is involved. Therefore, the standard GP has a cubic time complexity $O(n^3)$ [48], whereas the standard ANN has a linear time complexity $O(n)$, where n is the number of training data. Nevertheless, the GPs predict not only mean values but also the associated variances. Therefore, they are valuable in uncertainty quantification and robust optimization [41–43].

The MSEs of predictions at different time periods for different ML models are compared in Fig. 6. In general, the MSE of prediction increases over time for different ML models. Since the prediction of the phase field at one time step relies on the predictions from previous steps, the error will be accumulated over time. It is also 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

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

ML models. Nevertheless, the MSEs of extrapolation for the LF-PCNN, MF-PCNN1, MF-PCNN3, MF-PCNN4 and MF-PCNN6 are one order of magnitude lower than that of the ANN.

Table 5. Quantitative comparison between different ML models in the phase transition example

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+62.58=1161.27	0.0218	0.0587
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+1498.41=5368.41	0.0129	0.0648

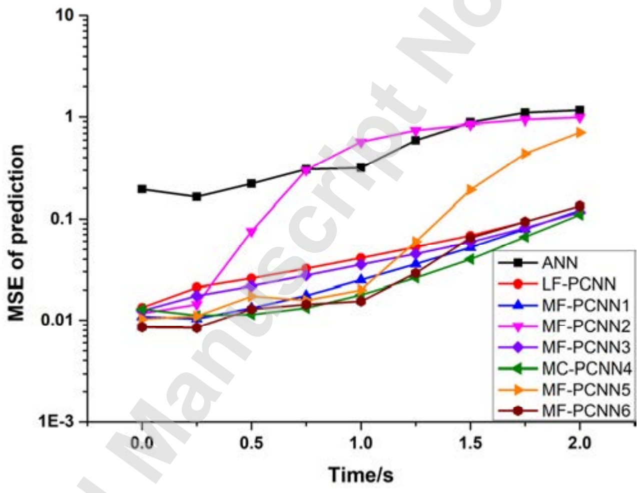


Fig. 6. The change of MSE of prediction for different ML models.

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 to 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 most likely caused by overfitting since the DANN2 has more neurons than the

DANN1. The MSE of prediction for the MF-PCNN3 is slightly lower than that of the LF-PCNN but higher than that of MF-PCNN1 for $t \in [0, 2]$. Notice that $t = 0.5$ is outside the time range $t \in [0, 0.2]$ of the HF training data for the HF-PCNN1, and the prediction is based on extrapolation. The errors indicate that DANN1 is better than GP1 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 discrepancy 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 discrepancy 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 $t \in [0.0, 1.5]$. Therefore, the MF-PCNN6 has a lower MSE of prediction than that of MF-PCNN3 at the cost of longer training time. However, the MSE of prediction for the MF-PCNN6 becomes the same as that of the LF-PCNN when $t > 1.75$, indicating the failure of prediction by GP2.

Among all ML models in this example, the MF-PCNN1 has the best performance since it has a relatively low training time and very good accuracy. The good performance of the MF-PCNN1 is mostly due to the simpler neural network structure of the DANN1.

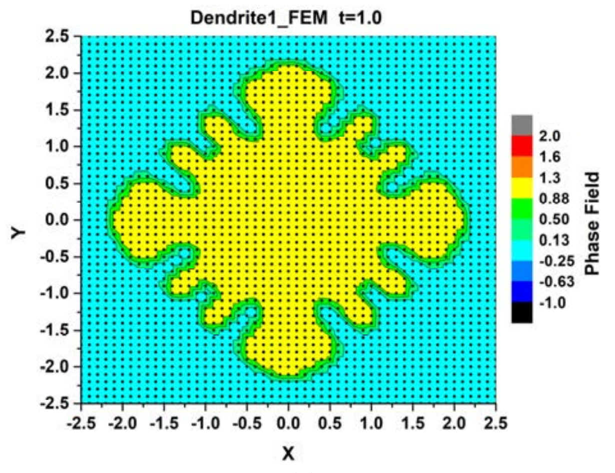
3.3 Dendritic growth example

Here, instead of showing the complete design optimization procedure for iterative predictions and searching, we only show the evolutions of dendritic growth predicted by MF-PCNNs with two different sets of design variables. For the first design, the liquidus temperature is $q_e = 1$ and latent heat is $K = 2$.

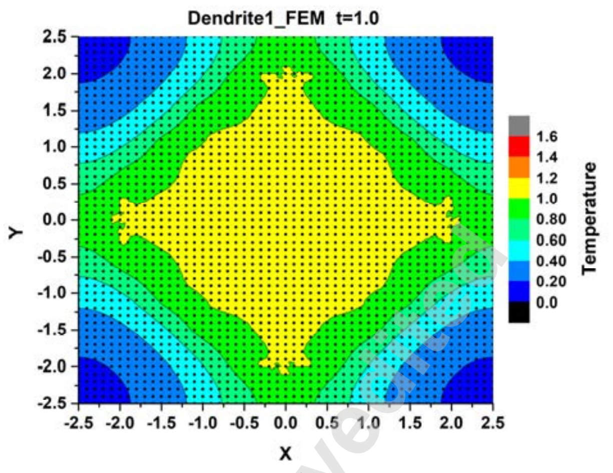
Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

The predicted phase fields and temperature fields from FEM and the MF-PCNNs at t = 1.0 are shown in Fig. 7. The settings for the two MF-PCNNs are compared in Table 3. There are 51×51 sampling points in the 2D domain. For the second design, the liquidus temperature is $q_e = 1.4$ and latent heat is $K = 2.8$. The predicted phase fields and temperature fields from FEM and the MF-PCNNs at t = 1.0 are shown in Fig. 8. For both cases, the predicted shapes of the primary arms from the MF-PCNNs are similar to the FEM simulation result, whereas the predicted secondary arms deviate from the simulation. Since secondary arms contain very thin and delicate features, more training data or physical constraints are needed to predict the secondary arms more accurately. The number of neurons also needs to be increased. For both dendrites, the predicted temperature fields from the MF-PCNNs correspond to the FEM simulation result very well, given that gradients in the temperature field is smaller than the phase field. The MSEs of prediction from the MF-PCNN for dendritic growth at t = 1.0 are shown in Table 6. It is shown that the MSE of prediction for the phase field is at least one order of magnitude larger than that for the temperature field in both cases. Compared to the MF-PCNN1, the MF-PCNN2 has more physical constraints. Therefore, the MSE of prediction for phase field from the MF-PCNN2 is lower than that from the MF-PCNN1. In the future work, the architecture of MF-PCNNs needs to be optimized for such multiphysics examples, which will be largely determined by the resolutions of predicted fields.

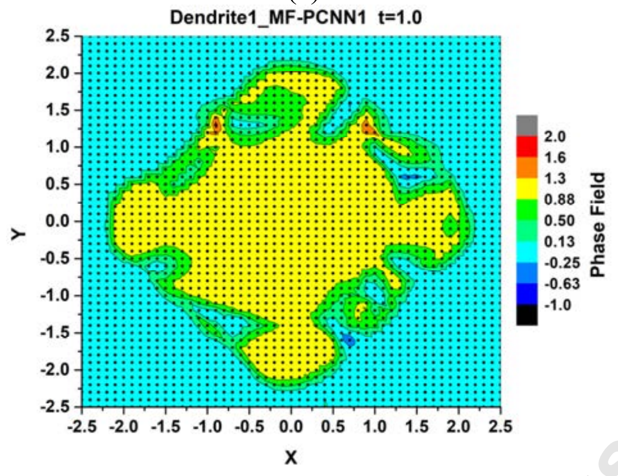
Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME



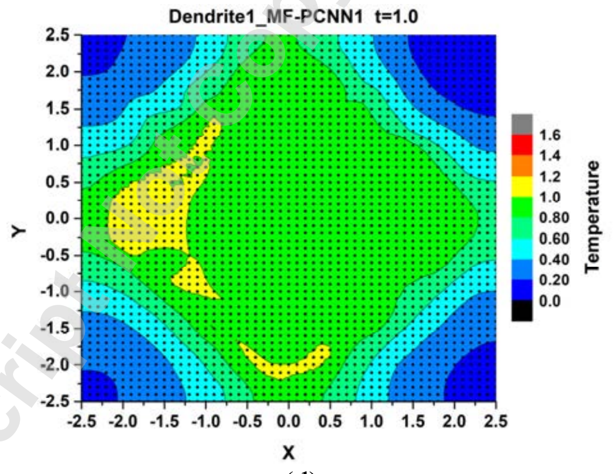
(a)



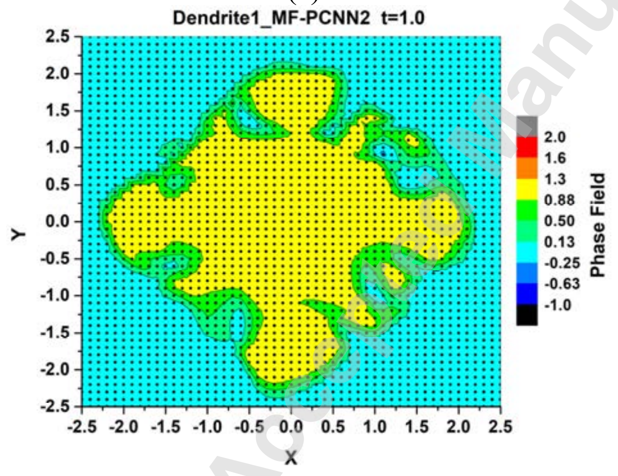
(b)



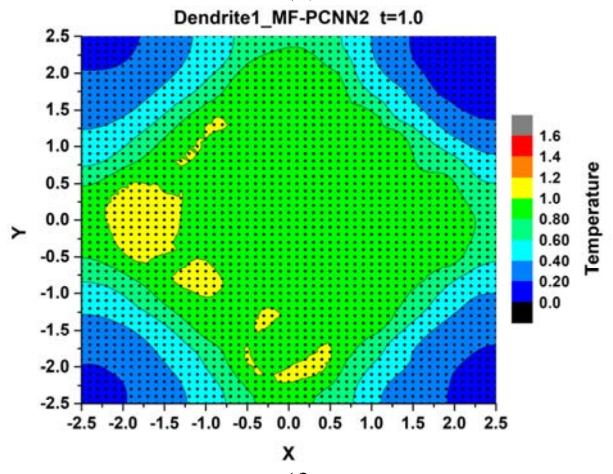
(c)



(d)



(e)



(f)

Fig. 7. The predicted phase fields and temperature fields from different models for the first material option ($q_e = 1$; $K = 2$) at $t = 1.0$. Phase fields are shown in (a), (c), and (e). Temperature fields are shown in (b), (d), and (f).

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

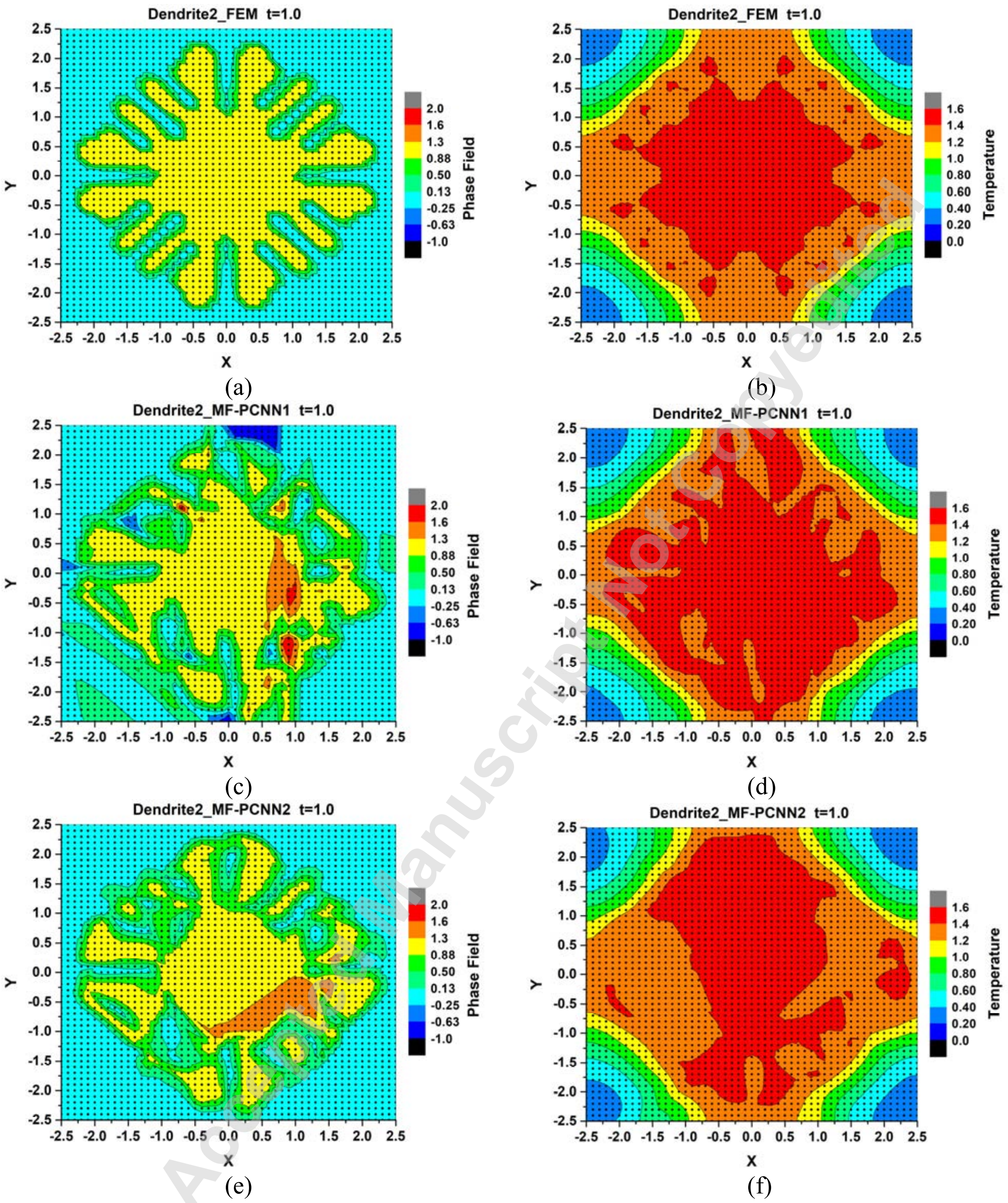


Fig. 8. The predicted phase fields and temperature fields from different models for the second material option ($q_e = 1.4$; $K = 2.8$) at $t = 1.0$. Phase fields are shown in (a), (c), and (e). Temperature fields are shown in (b), (d), and (f).

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

Table 6. MSEs of prediction from the MF-PCNN for dendritic growth at $t = 1.0$

		Training time (second)	MSE of prediction for phase field	MSE of prediction for temperature field
Dendrite 1	MF-PCNN1	4320.18	0.0356	0.0047
	MF-PCNN2	8738.51	0.0346	0.0049
Dendrite 2	MF-PCNN1	37836.12	0.1127	0.0010
	MF-PCNN2	154791.97	0.0713	0.00384

4. DISCUSSION AND CONCLUSIONS

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.

In engineering and physical sciences, the knowledge about the system behaviors is typically described by PDEs as well as the associated initial and boundary condition information. 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. When new knowledge about the system is obtained, more physical constraints can be added into PCNNs to improve the prediction accuracy. Thus the LF-PCNN and HF-PCNN to construct a MF-PCNN can be trained with different sets of constraints. It is reasonable to assume that the information of constraints can

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

be obtained efficiently by the evaluation of analytical functions, which is less costly than obtaining simulation data for 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 three examples of materials modeling. The first example is the heat equation with zero Neumann boundary conditions, which is a linear PDE. The second example is the Allen-Cahn equation with periodic boundary condition, which is a nonlinear PDE. The PCNN is effective for these two different types of PDEs with different boundary conditions. The third example is the dendritic growth during solidification where heat transfer and phase transition are coupled. 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 robust than GPs for extrapolation to predict the discrepancy between the LF-PCNN and HF-PCNN.

The results in this paper have demonstrated the effectiveness of the proposed MF-PCNN framework for simulation prediction. Ideally, to construct the process-structure-property relationship using ML tools, the design variables should be directly incorporated as the input vector. This pure black-box approach however increases the complexity of ML tools and needs a significant amount of training data. The purpose of the proposed MF-PCNN framework is to reduce the required amount of training data by taking a gray-box approach. Although the offline training of MF-PCNN is slow, which takes up to several hours, its online evaluation or forward prediction can be done in a few seconds, once the training is finished. It was also shown that the training efficiency can be improved if the training data are from numerical simulations with different fidelities. The training data however 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.

Future work will include several extensions. First, the complete framework of PCNN based design optimization will be implemented and evaluated. Second, although classical ANN is a good baseline model to demonstrate the proposed MF-PCNN framework, the ANN could be replaced by the recurrent neural networks, such as long short-term memory neural networks, which may be more appropriate to solve time-dependent problems. Third, the adaptive weighting scheme is training data-dependent. Therefore, it is possible to be further optimize the adaptive weighting scheme based on the training data. The architecture of MF-PCNNs can also be optimized for efficiency improvement by training the LF-PCNN and DANN together. Fourth, to further improve the training efficiency, a sequential sampling strategy can be adopted to obtain an optimal combination of the HF and LF sample points for a given computational budget [42]. Finally, a more rigorous and comprehensive comparison of the scalability between the proposed MF-PCNN and multi-fidelity GP models is needed. Theoretically, the standard GP has a cubic time complexity, whereas the standard ANN has a linear time complexity. However, incorporating the computational cost of data sampling in the multi-fidelity simulation scenario will provide an overall scalability picture in terms of training and prediction time.

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

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

enhances the efficiency of engineering design when high-fidelity simulations need to be run repetitively to obtain samples for design optimization. The required number of samples for optimization for high-dimensional problems usually is very large. The cost of training machine learning tools therefore can only be justified for complex problems with a high-dimensional searching space. For high-dimensional problems, the physical constraints can still be applied in the same scheme. However, as the number of constraints increases, they may not be treated as equally important, and those with trivial weights will be removed. In principle, as the dimension of the problem increases, the advantage of PCNNs will become more prominent because the required amount of training data can be reduced more significantly. 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] Cang, R., Xu, Y., Chen, S., Liu, Y., Jiao, Y., and Ren, M. Y., 2016, "Microstructure Representation and Reconstruction of Heterogeneous Materials via Deep Belief Network for Computational Material Design," J. Mech. Des., **139**(7), p. 071404.

[2] Yang, Z., Li, X., Brinson, L. C., Choudhary, A. N., Chen, W., and Agrawal, A., 2018, "Microstructural Materials Design via Deep Adversarial Learning Methodology," J. Mech. Des., **140**(11), p. 111416.

[3] Pan, S. J., and Yang, Q., 2010, "A Survey on Transfer Learning," IEEE Trans. Knowl. Data Eng., **22**(10), pp. 1345–1359.

[4] Lee, S.-I., Chatalbashev, V., Vickrey, D., and Koller, D., 2008, "Learning a Meta-Level Prior for Feature Relevance from Multiple Related Tasks," *Acl*, pp. 489–496.

[5] Yao, Y., and Doretto, G., 2010, "Boosting for Transfer Learning with Multiple Sources," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 1855–1862.

[6] Raina, R., Battle, A., Lee, H., Packer, B., and Ng, A. Y., 2007, "Self-Taught Learning: Transfer

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

Learning from Unlabeled Data," *Proceedings of the 24th International Conference on Machine Learning*, ACM Press, New York, New York, USA, pp. 759–766.

[7] Dai, W., Xue, G.-R., Yang, Q., and Yu, Y., 2007, "Co-Clustering Based Classification for out-of-Domain Documents," *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM Press, New York, New York, USA, pp. 210–219.

[8] Lawrence, N. D., and Platt, J. C., 2004, "Learning to Learn with the Informative Vector Machine," *Proceedings of the Twenty-First International Conference on Machine Learning*, ACM Press, New York, New York, USA, p. 65.

[9] Bonilla, E., Chai, K. M., and Williams, C., 2007, "Multi-Task Gaussian Process Prediction," *Advances in Neural Information Processing Systems*, pp. 153–160.

[10] Tercan, H., Guajardo, A., Heinisch, J., Thiele, T., Hopmann, C., and Meisen, T., 2018, "Transfer-Learning: Bridging the Gap between Real and Simulation Data for Machine Learning in Injection Molding," Procedia CIRP, **72**, pp. 185–190.

[11] 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.

[12] Tran, A., Furlan, J. M., Pagalthivarthi, K. V., Visintainer, R. J., Wildey, T., and Wang, Y., 2019, "WearGP: A Computationally Efficient Machine Learning Framework for Local Erosive Wear Predictions via Nodal Gaussian Processes," Wear, **422–423**, pp. 9–26.

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

[14] 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.

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

[16] 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.

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

[18] Nagarajan, H. P. N., Dimassi, S., Haapala, K. R., Gary Wang, G., Bakrani-Balani, S., Coatanéa, E., Hamedi, A., Jafarian, H., and Mokhtarian, H., 2018, "Knowledge-Based Design of Artificial Neural Network Topology for Additive Manufacturing Process Modeling: A New Approach and Case Study for Fused Deposition Modeling," J. Mech. Des., **141**(2), p. 021705.

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] Kennedy, M., and O'Hagan, A., 2000, "Predicting the Output from a Complex Computer Code When Fast Approximations Are Available," Biometrika, **87**(1), pp. 1–13.

[39] Fernández-Godino, M. G., Park, C., Kim, N.-H., and Haftka, R. T., 2016, "Review of Multi-Fidelity Models," arXiv:1609.07196.

[40] Peherstorfer, B., Willcox, K., and Gunzburger, M., 2018, "Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization," SIAM Rev., **60**(3), pp. 550–591.

[41] Wang, X., Liu, Y., Sun, W., Song, X., and Zhang, J., 2018, "Multidisciplinary and Multifidelity Design Optimization of Electric Vehicle Battery Thermal Management System," J. Mech. Des., **140**(9), p. 094501.

Journal of Mechanical Design. Received March 04, 2019;
Accepted manuscript posted July 10, 2019. doi:10.1115/1.4044400
Copyright © 2019 by ASME

[42] Zhou, Q., Wang, Y., Choi, S. K., Jiang, P., Shao, X., and Hu, J., 2017, "A Sequential Multi-Fidelity Metamodeling Approach for Data Regression," Knowledge-Based Syst., **134**, pp. 199–212.

[43] Zhou, Q., Wang, Y., Choi, S. K., Jiang, P., Shao, X., Hu, J., and Shu, L., 2018, "A Robust Optimization Approach Based on Multi-Fidelity Metamodel," Struct. Multidiscip. Optim., **57**(2), pp. 775–797.

[44] 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.

[45] 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.

[46] 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.

[47] Tran, A., Liu, D., Tran, H., and Wang, Y., 2019, "Quantifying Uncertainty in the Process-Structure Relationship for Al–Cu Solidification," Model. Simul. Mater. Sci. Eng., **27**(6), p. 064005.

[48] Tran, A., Sun, J., Furlan, J. M., Pagalthivarthi, K. V., Visintainer, R. J., and Wang, Y., 2019, "PBO-2GP-3B: A Batch Parallel Known/Unknown Constrained Bayesian Optimization with Feasibility Classification and Its Applications in Computational Fluid Dynamics," Comput. Methods Appl. Mech. Eng., **347**, pp. 827–852.