Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 373 (2021) 113452
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Hierarchical Deep Learning Neural Network (HiDeNN): An artificial intelligence (AI) framework for computational science and engineering

Sourav Saha[a,1], Zhengtao Gan[b,1], Lin Cheng[b,1], Jiaying Gao[b], Orion L. Kafka[b,2], Xiaoyu Xie[b], Hengyang Li[b], Mahsa Tajdari[b], H. Alicia Kim[c], Wing Kam Liu[b,*]

[a] *Theoretical and Applied Mechanics, Northwestern University, Evanston, IL 60208, USA*
[b] *Department of Mechanical Engineering, Northwestern University, Evanston, IL 60208, USA*
[c] *Structural Engineering Department, University of California, San Diego, 9500 Gilman Drive, San Diego, CA 92093, USA*

Available online 21 October 2020

Abstract

In this work, a unified AI-framework named Hierarchical Deep Learning Neural Network (HiDeNN) is proposed to solve challenging computational science and engineering problems with little or no available physics as well as with extreme computational demand. The detailed construction and mathematical elements of HiDeNN are introduced and discussed to show the flexibility of the framework for diverse problems from disparate fields. Three example problems are solved to demonstrate the accuracy, efficiency, and versatility of the framework. The first example is designed to show that HiDeNN is capable of achieving better accuracy than conventional finite element method by learning the optimal nodal positions and capturing the stress concentration with a coarse mesh. The second example applies HiDeNN for multiscale analysis with sub-neural networks at each material point of macroscale. The final example demonstrates how HiDeNN can discover governing dimensionless parameters from experimental data so that a reduced set of input can be used to increase the learning efficiency. We further present a discussion and demonstration of the solution for advanced engineering problems that require state-of-the-art AI approaches and how a general and flexible system, such as HiDeNN-AI framework, can be applied to solve these problems.

Keywords: Deep learning; Machine learning; Reduced order model; Data-driven discovery; Multiscale simulation; Artificial intelligence

1. Introduction

With the great development of modern computer and computational algorithms, computational science and engineering have achieved enormous success in almost all fields, such as physics, chemistry, biology, mechanical, civil, and materials science and engineering. However, many problems in computational science across the disciplines are still challenging. We propose that there are three major classes, or *types*, of problems puzzling the community of computational science and engineering. These three types are:

* Corresponding author.
E-mail address: w-liu@northwestern.edu (W.K. Liu).
[1] Sourav Saha, Zhengtao Gan, and Lin Cheng contributed equally to this work.
[2] Current address: Applied Chemicals and Materials Division, National Institute of Standards and Technology, Boulder, CO 80305, USA.

https://doi.org/10.1016/j.cma.2020.113452
0045-7825/

1. **Type 1 or purely data-driven problems**: The class of analyses with unknown or still developing governing physics but abundant data. For these problems, the lack of knowledge of physics can be compensated by the presence of considerable data from carefully designed experiments regarding the system response.
2. **Type 2 or mechanistically insufficient problems with limited data:** The term *mechanistic* refers to the theories which explain a phenomenon in purely physical or deterministic terms [1]. Type 2 problems are characterized by physical equations that require complementary data to provide a complete solution.
3. **Type 3 or computationally expensive problems:** The problems for which the governing equations are known but too computationally burdensome to solve.

We will attempt to show that artificial intelligence (AI), particularly a subset of AI, deep learning, is a promising way to solve these challenging problems. An AI system is identified by its capability to perform tasks which currently humans perform in a better way [2]. This is famously judged by the Turing test, proposed to measure the intelligence of a machine by its capability to imitate human behavior [3]. An AI system can be classified into three classes, (a) "weak" or narrow AI, (b) general AI, and (c) super AI [4]. A narrow or "weak" AI is designed to perform a specific task and outperform any human in doing that. General AI refers to an AI system that may exhibit intelligent behavior in different areas and might outperform humans [5]. Super AI is a conceptual version of the technology that is the supreme point where machine achieves superhuman intelligence and can perform abstract thinking [6]. Almost all of the AI systems we see around us fall in the category of narrow AI. Super and general AI are still futuristic ideas. Machine learning (ML) is a form of narrow AI [7] and defined as the process by which computers, when given data, create their own knowledge (hence the term learning) by identifying patterns in data [8,9]. Deep neural network is a subset of machine learning tools by which computers "understand" challenging and complex concepts by building the deep hierarchy of simpler concepts [9]. A generic deep neural network consists of input layer, hidden layers, and output layer where the input (layer) is connected (nonlinear information processing) through an activation function (hidden layer) to the output (layer) [8].

There is a growing tendency across the scientific communities to engage narrow AI (machine learning or deep learning) to solve problems in disciplines such as mechanics [10–12], biology and bio-medicine [13–15], materials science and engineering [16–18], manufacturing process monitoring [19–21], topology optimization [22–24], design under uncertainty [25], and miscellaneous engineering disciplines [26–28]. The scope of machine learning tools to aid or solve computational science problems goes beyond merely regressing non-linear data. Deep neural network and transfer learning are now being applied to discover hidden governing physical laws from data [29–31], speed up the computation in multiscale and multiphysics problems [32–36], characterize and reconstruct complex microstructures [37], design of heterogeneous materials and metamaterials [38,39], discover new materials [40,41], and to model path- and history-dependent problems [42–45]. Fig. 1 shows AI tools currently in use to solve state-of-art computational science problems. The AI tools include data generation and collection techniques, feature extraction techniques (wavelet and Fourier transform [46], principal component analysis [46]), dimension reduction techniques (clustering, self-organizing map [21,46]), regression (neural network, random forest) [46], reduced order models (can be something similar to regression techniques or more advanced technique like self-consistent clustering analysis (SCA) [47,48] or Proper Orthogonal Decomposition (POD) [46]) and classification (convolutional neural networks or CNN [46]).

There are several practical challenges in directly applying current AI frameworks to solve aforementioned types of problems: (1) it is often difficult to decide on the criteria to identify the type of the problem and on the set of tools to use; (2) for a computational materials scientist or practicing engineer, it might become a challenging task to go back and forth among the different machine learning tools; (3) a design engineer needs to have a closed form relationship among different parameters controlling the desired property of the system. Moreover, the bridge connecting seemingly disparate fields of data-science and computational methods has to be a general one so that a common framework can be used to solve problems of different nature and originating from different physics. One other problem for applying AI frameworks in science and engineering is the paucity of data. Often experiments are too expensive to be useful to generate a large amount of data. Computational and theoretical predictions are limited by inherent assumptions. Considering these current difficulties and constraints, we propose a unified deep learning framework named Hierarchical Deep Learning Neural Network (HiDeNN). An advantage of using the HiDeNN structure is that such a neural network has a universal approximation capability enabling it to correctly interpolate among the data points generated by extremely non-linear relationships. HiDeNN can identify the governing physics from an experimental dataset without any prior knowledge and therefore can fill the missing link between data and

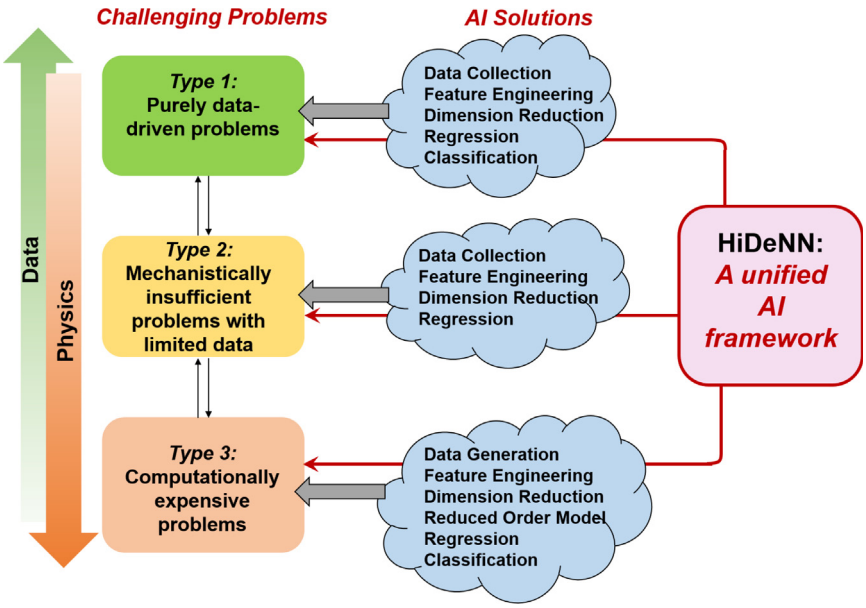

Fig. 1. A comparative picture of state-of-the-art AI tools in computational science and engineering and the proposed Hierarchical Deep Neural Network (HiDeNN) framework. HiDeNN offers the advantage of being a unified framework to solve the problem in computational science and engineering without resorting to different set of tools for different types of problem.

mechanistic knowledge. As will be explained, HiDeNN also has the capability to incorporate mechanistic knowledge in training through proper definition of the loss function. This will reduce the necessity of a large amount of data to get an accurate prediction. A practical example is provided in a companion paper by Tajdari et al. [49], submitted to the same issue, to demonstrate in detail how a small amount of medical data available for adolescent idiopathic scoliosis can be used with mechanistic knowledge and deep learning to predict spine curvature. All the machine learning tools and computational methods mentioned earlier can be built into HiDeNN, eliminating the need for the user to decide on specific tools.

This article is organized as follows: Section 2 introduces and describes the components of HiDeNN, Section 3 presents the application of HiDeNN framework by solving three illustrative problems, Section 4 discusses three examples from each *type* of challenging problems, how those are solved using state-of-the-art methods, and recast the solution of the problems using HiDeNN, Section 5 proposes possible extensions of HiDeNN for general problems.

2. Hierarchical deep learning neural network (HiDeNN)

An example structure of HiDeNN for a general computational science and engineering problem is shown in Fig. 2. Construction of HiDeNN framework is discussed in following points:

- The **input layer** of HiDeNN consists of inputs from spatial (Ω), temporal (t), and parameter (D) spaces. The neurons of this layer serve as independent variables of any physical system.
- The input layer of HiDeNN is connected to a set of neurons that represents a set of **pre-processing functions** $f(\mathbf{x}, t, \mathbf{p})$ where $\mathbf{x}$, t, and $\mathbf{p}$ are position, time, and parameter vector, respectively. These functions can be thought of as tools for feature engineering. For example, the pre-processing functions can convert dimensional parameters into dimensionless inputs. Such conversion can be necessary for fluid mechanics problems where, for example, the Reynolds (Re) number is important.
- The layer of the pre-processing functions is connected to **structured hierarchical deep learning neural networks (DNN)**. Hierarchical DNN layers consist of parameter layers, customized physics-based neural networks (**PHY-NN**) or experimental-data based neural network (**EXP-NN**). In Fig. 2, the indices i and j indicate similar neural networks layers can be appended for both PHY-NN and EXP-NN, respectively. The **PHY-NN** refers to a neural network formulated from physics-based data and **EXP-NN** neural network is designed from experimental data.

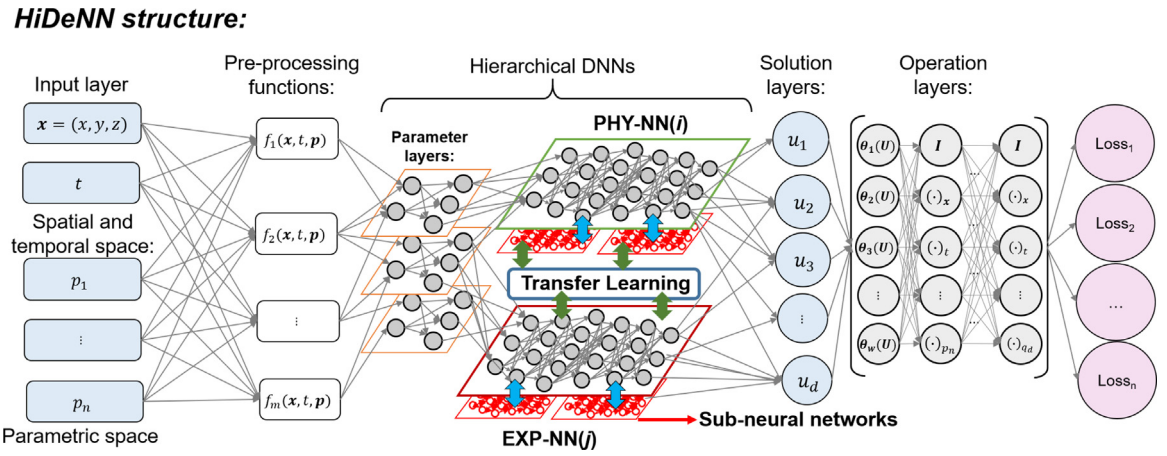

Fig. 2. Detail construction of the proposed HiDeNN framework. The input layer takes in space, time, and parameter variables of a system. The input layer is connected to the pre-processing function, Hierarchical DNNs, and finally the solution layer. Governing equations can be obtained from solution layers through the operation layer and the loss function.

- In the hierarchical DNNs portion of the HiDeNN (see Fig. 2), we see multiple **sub-neural networks** connected (the red blocks). We define the sub-neural networks as stand-alone neural networks that can provide input to the PHY-NN or EXP-NN. This multi-level structure is the source of the name "Hierarchical" in HiDeNN.
- The Hierarchical DNNs can be any type of neural network, including convolutional neural network (CNN), recurrent neural network (RNN), and graph neural network (GNN). In order to enhance the capability of PHY-NN or EXP-NN transfer learning technique can be adopted in the proposed structure.
- Lack of data is of big concern in AI community. Available experimental data often come from dissimilar experimental or computational conditions making them hard to use directly in an AI framework. As one means of dealing with the problem, HiDeNN has provision for transfer learning in the Hierarchical DNN layer. The PHY-NNs and EXP-NNs can be trained separately with the available computational and experimental data. Later, these individual neural networks can be combined through transfer learning.
- The Hierarchical DNN layer is connected to the **solution layer**. The solution layer represents the set of dependent variables of any particular problem.
- To discover unknown governing equations from data, HiDeNN has **operation layers**. In this layer, the neurons are connected through weights and biases in a way that mimics the behavior of different spatiotemporal operators. Through proper training (i.e. minimization of the **loss** function in the HiDeNN), the operation layer can be trained to discover hidden physics from data.
- The **loss function** layer of HiDeNN contains multiple loss function terms as shown in the figure. Each loss function can either come from the hierarchical DNNs or the operational layers. These functions can be optimized simultaneously or separately depending on the problem. This unique feature of the HiDeNN provides the flexibility to solve problems with scarce and abundant data by combining it with physics.

3. Application of HiDeNN framework

In this section, three examples of HiDeNN are discussed in detail to demonstrate the framework's capability.

3.1. HiDeNN for learning the discretization

In this example, we will use the HiDeNN to solve a solid mechanics problem and capture stress concentration by training the position of the nodes used for the discretization to minimize the potential energy of the system. In HiDeNN, the interpolation function for approximating the solution is obtained by constructing neural network and training the weights and biases simultaneously [50]. Fig. 3 presents a 2D bi-linear HiDeNN element, $\mathcal{N}(x, y; \boldsymbol{w}, \boldsymbol{b}, \mathscr{A})$ at node (x_I, y_J), constructed by using the well-defined building blocks proposed in [50]. The

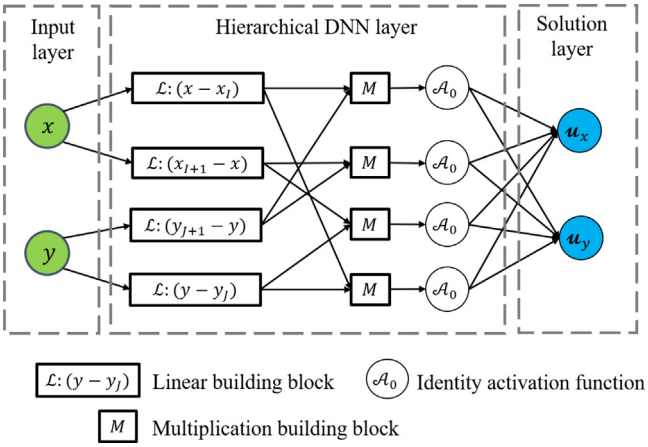

Fig. 3. Construction of the bi-linear HiDeNN element at nodal (x_I, y_J) using building blocks proposed in [50]. The input of the unit neural network is nodal coordinates $(\boldsymbol{x}, \boldsymbol{y})$, while the output is the nodal displacements $(\boldsymbol{u}_x, \boldsymbol{u}_y)$. Note that the weights and biases in the above neural network are functions of nodal positions (x_I^*, y_J^*). The training of the neural network is equivalent to find the optimally nodal positions to achieve optimal performance for the loss function in Eq. (1a).

arguments $\boldsymbol{w}$, $\boldsymbol{b}$, and $\boldsymbol{\mathcal{A}}$ are the weights, biases, and activation function of the neural networks. Here, both $\boldsymbol{w}$ and $\boldsymbol{b}$ are functions of nodal positions. Therefore, updating the weights and biases during training implies updated nodal coordinates. The interpolation function at (x_I, y_J) can be expressed as $\mathcal{N}(x, y; x_I^*, y_J^*, \boldsymbol{\mathcal{A}})$ where x_I^*, y_J^* are the updated nodal positions. As illustrated in Fig. 3, the inputs of the HiDeNN element are the nodal coordinates, (x, y), while the output are the nodal displacements, $\boldsymbol{u}_x$ and $\boldsymbol{u}_y$. When the nodal positions are fixed, HiDeNN is equivalent to standard FEM, while when the nodal coordinates, x_I^*, y_J^*, in the weights and biases are updated during training, HiDeNN is able to accomplish better results like the r-adaptivity in FEM. However, differentiating from stress-based error estimator in r-adaptivity, the proposed HiDeNN method updates the nodal position by learning the performance through structured deep neural networks. The updated nodal coordinates will replace the original nodal coordinates after the training process until the optimal solution accuracy is achieved.

By assembling the HiDeNN elements, a unified neural network is formed to solve any problem of interests. Fig. 4 shows the assembled neural network for a 2D problem. In the operations layer, the neuron $f_1(\cdot)$ is used to formulate the Neumann boundary conditions while the Dirichlet boundary condition is automatically satisfied through the optimization of the loss function. The weights of green arrows in Fig. 4 represent the constitutive model, in this case, the stiffness matrix for an elastic problem. The neural network in Fig. 4 is a variation of the HiDeNN framework in Fig. 2 for solving the problems with known governing equations. The input is the spatial coordinates, the PHY-NN is the construction of the interpolation function and the solution is the displacements. The operation layer is used to define the loss function (total potential energy) given in Eq. (1a). For more details, please refer to [50]. Here, the HiDeNN method is implemented in PyTorch v1.6.0, and the training for the variables, such as nodal displacements and nodal positions, is performed by using the autograd package by Paszke et al. [51] in PyTorch on a laptop with an Intel(R) Core(TM) i7-9750H @2.60 GHz and an NVIDIA GeForce RTX 2060 Graphics Processing Unit (GPU).

$$\mathscr{L}(\boldsymbol{u}^h; \boldsymbol{f}, \boldsymbol{t}) = \frac{1}{2}\int_\Omega \sigma(\boldsymbol{u}^h) : \varepsilon(\boldsymbol{u}^h) d\Omega - \left(\int_\Omega \boldsymbol{u}^h \cdot \boldsymbol{f} d\Omega + \int_{\Gamma_t} \boldsymbol{u}^h \cdot \bar{\boldsymbol{t}} d\Gamma\right) \tag{1a}$$

$$\boldsymbol{u}^h(x, y; \boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{\mathcal{A}}) = \sum_n^{n=N} \mathcal{N}_n(x, y; x_n^*, y_n^*, \boldsymbol{\mathcal{A}}) u_n \tag{1b}$$

where $\boldsymbol{u}^h$ is the displacement field, $\boldsymbol{x}^*$ and $\boldsymbol{y}^*$ are the vector used to save the nodal positions, N is the total number of nodes, u_n and $\mathcal{N}_n(x, y; x_n^*, y_n^*, \boldsymbol{\mathcal{A}})$ denote the nodal displacement and interpolation function at node n. $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$ are the stress and strain tensors, respectively, $\boldsymbol{f}$ is the body force, and $\boldsymbol{t}$ is the external traction applied to boundary Γ_t. For linear elastic problem, we have $\boldsymbol{\varepsilon} = \frac{1}{2}(\nabla \boldsymbol{u}^h + (\nabla \boldsymbol{u}^h)^T)$. Note that to avoid the inverted elements when the

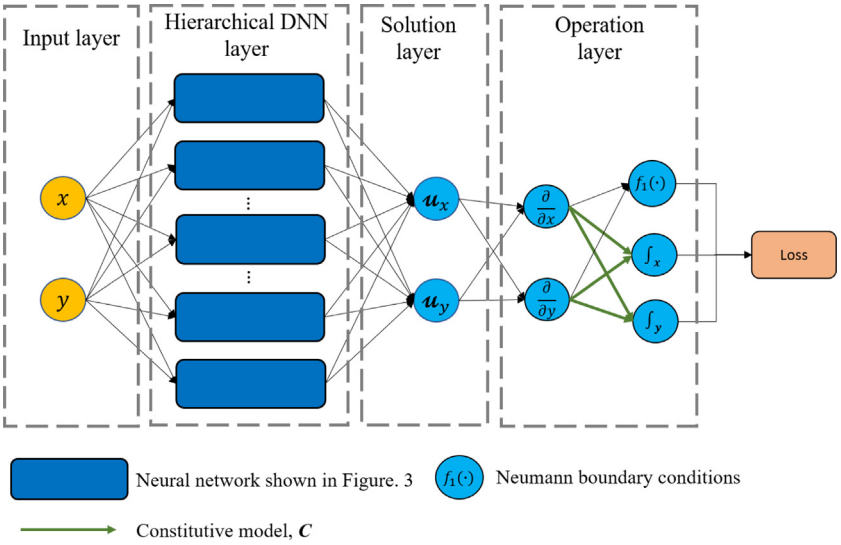

Fig. 4. The assembled neural network by the unit neural network in Fig. 3 for solving the 2D elastic problem. The input of the neural network is the nodal coordinates. The output is the solution of the nodal displacements. The operation layer is used to formulate the governing equations. The loss function is defined in Eq. (1a). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

nodes are moved during training, a stop criterion for detecting a jump of the loss function is added. Inversion of an element will cause the loss function will increase suddenly, at which point the training will be stopped and the previous iteration will be taken as the final result.

$$e_{\mathscr{L}} = \left| \frac{\mathscr{L}_{n+1} - \mathscr{L}_n}{\mathscr{L}_n} \right| \tag{2}$$

where $e_{\mathscr{L}}$ denotes the change of the loss function between the neighbor iterations. When $e_{\mathscr{L}} > 0.2$, the training is stopped and the previous iteration is taken as the results.

To assess the method, we compare the computational cost of HiDeNN with the standard FEM. To do this, we fix the nodal positions during optimization similar to standard FEM. Under such conditions, HiDeNN solves a problem by minimizing a loss function, which is the potential energy of the structure for a mechanics problem, using a state-of-the-art optimizer (i.e. the Adam method [52]) available in most deep learning software packages. Fig. 5 illustrates the problem used for the study. The test problem is an elastic material under simple tensile loading with four initial elliptical voids, solved under the plane stress condition. The domain of the test problem is a square with dimensions 2 by 2 The displacement of left side of the domain is fixed while a uniform loading of $F = 20$ is applied to the right side along the $+x$-direction. Young's modulus, E of the elastic material is 10^5, and the Poisson's ratio, ν, is 0.3. The domain is discretized by conformal mesh with differing numbers of quadrilateral elements using Abaqus [53]. We consider several conformal meshes with an increasing number of degrees of freedom: 1154, 2022, 4646, 8650, 16 612, 33 340, 65 430, 130 300, 259 430, 1 236 948 and 2 334 596 First, we solve the problem using Abaqus and the displacements at each node are later used as the reference for estimating the HiDeNN solution. Here, the $\|e\|_{L_1}$ error of the displacement defined in Eq. (3) is used for estimation. If $\|e\|_{L_1} < 10^{-6}$, the HiDeNN computations are considered to be finished.

$$\|e\|_{L_1} = \frac{\sum_{I=1}^{n} |u_I^{Abaqus} - u_I^{HiDeNN}|}{\sum_{I=1}^{n} |u_I^{Abaqus}|}, \tag{3}$$

u_I^{Abaqus} is the displacement at node I obtained by Abaqus, while u_I^{HiDeNN} is the corresponding value obtained by HiDeNN method. I is the index for the nodes in the domain, and n denotes the total number of nodes within the domain.

The computational time of HiDeNN with respect to the degrees of freedom (DOFs) is plotted on logarithmic axes in Fig. 6. As can be seen, the computational cost of HiDeNN increases with the degrees of freedom (DOFs).

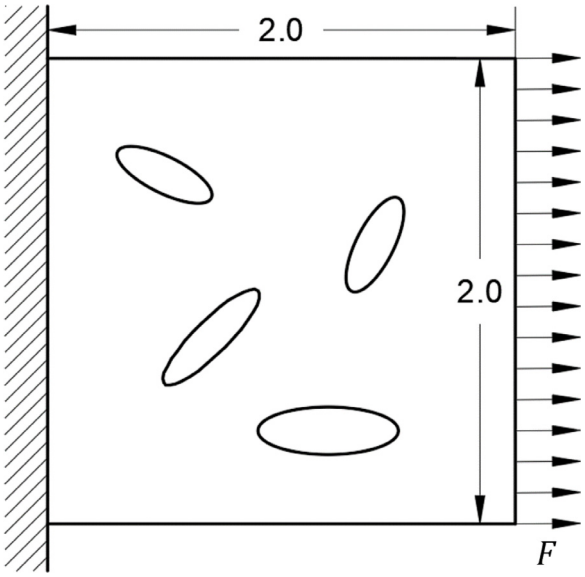

Fig. 5. Schematic diagram of the test problem, a square domain with four initial elliptical voids. The dimensions of the square domain are 2×2. Young's modulus of the material is 10^5 and Poisson's ratio is 0.3. The left side of the domain is fixed while a uniform load of $F = 20$ is applied to the right-side of the domain.

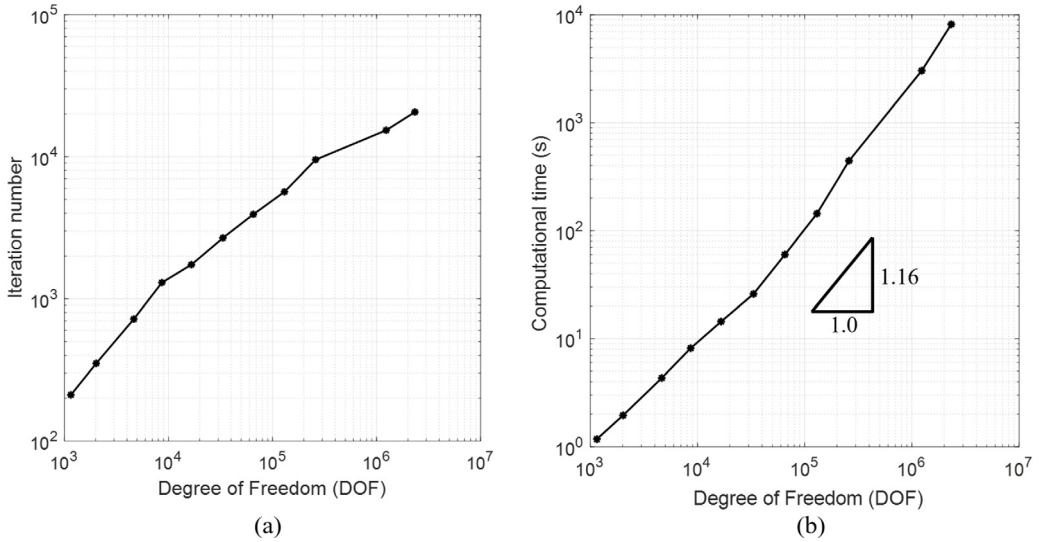

Fig. 6. Computational iterations and time of HiDeNN with respect to the number of degrees of freedom. Computational time increases slightly faster than the degrees of freedom. (a) Iteration number versus degrees of freedom, (b) Computational time versus degrees of freedom.

It has an approximately linear relationship with the DOFs on the log–log plot with slope slightly larger than 1. This implies that computational cost increases more quickly than the number of degrees of freedom.

In order to show how the HiDeNN can "intelligently" capture the stress concentrations, we relax the nodal position constraints in the neural network and train the nodal positions and nodal displacements simultaneously. For comparison, a convergence study for the maximum local stress is conducted in Abaqus [53] with a convergence criterion of less than 1% of change between two cosecutively more refined meshes. The converged mesh is taken as the reference solution to examine the performance of the HiDeNN. The converged mesh and the stress distributions are given in Fig. 7(a), (b), and (c), respectively. The maximum local stress within the test geometry converges to

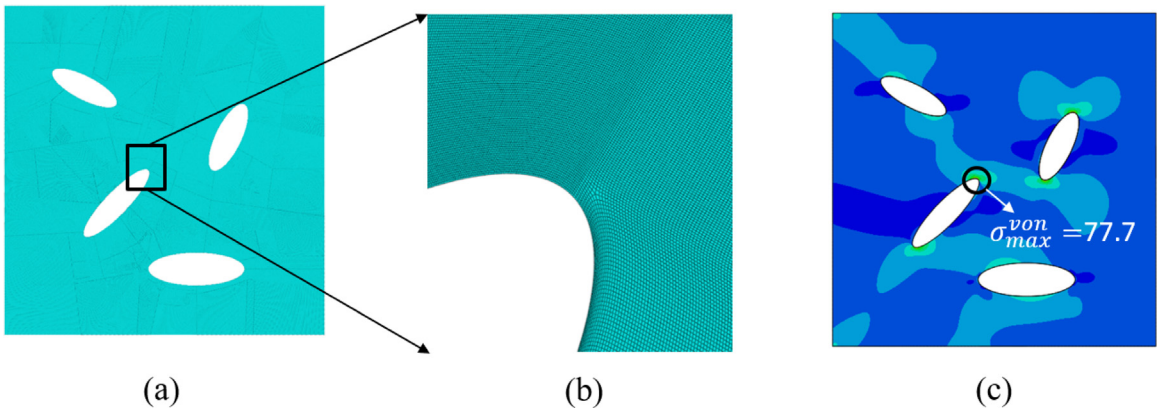

Fig. 7. Converged conformal mesh and the corresponding FEM results (von Mises stress) for the test problem with four elliptical holes. (a) Full domain with four elliptical holes, (b) detail of the converged mesh near the stress concentration, (c) stress distribution inside the full domain for the converged solution in Abaqus.

77.7 for a mesh with 3 867 168 quadrilateral elements and 7 748 156 DOFs. As illustrated in the figure, the maximum local stress occurs near the top corner of the bottom left ellipse. In order to capture the stress peak, an extremely fine mesh is required at this region when using standard FEM (refer to Fig. 7(b)).

For a one-to-one comparison between the FEM and HiDeNN solutions, the test problem is discretized with same conformal meshes as Abaqus. Four meshes are used, with 524 quadrilateral elements with 1154 DOFs, 938 quadrilateral elements with 2022 DOFs, 2194 quadrilateral elements with 4646 DOFs, and 4143 quadrilateral elements with 8650 DOFs, as shown in Fig. 8(e)–(h). Both HiDeNN and FEM are applied to these four meshes and compared against converged solution for maximum stress.

The maximum computed stresses from FEM and HiDeNN, and their differences from the converged, conforming mesh solution are tabulated in Table 1. For FEM, due to the inherent complexity for generating a conformal mesh to capture stress concentrations for ellipses, the predicted values from coarse mesh are still lower than the converged stresses (57.92%, 53.41%, 54.05%, and 54.44% lower for four cases). On the other hand, the results obtained by HiDeNN show much better accuracy through learning the **optimal** nodal positions. As shown in Fig. 8, HiDeNN moves the nodes during training to the regions with stress concentrations. Even with the coarsest discretization, 524 quadrilateral elements and 938 quadrilateral elements, HiDeNN is able to capture the maximum stress well, with about 2.70% and 2.06% difference from the converged value. We also compare the computational time with the HiDeNN method with the converged results. For the same accuracy for maximum stress, the HiDeNN method with moving nodes takes 17.53 s, 22.87 s, 31.09 s and 40.25 s while Abaqus takes about 1334.37 s. Fig. 8(i) presents the maximum von Mises stress obtained by both Abaqus and HiDeNN method with plotted against the number of elements in the mesh. When using the same coarse mesh, HiDeNN is able to precisely capture the stress concentration by moving the nodes to the maximum stress region. Note that the computational time for both HiDeNN and Abaqus is based on calculations conducted on a CPU. The performance of HiDeNN even with a coarse discretization demonstrates the potential for HiDeNN to bypass computationally expensive conformal mesh generation for complex geometry and reduce the expensive computational cost of capturing the stress concentration with conforming meshes. In concept, this is similar to isogeometric analysis (IGA), in that the difficulty of mesh generation is mitigated [54].

3.2. HiDeNN for multiscale analysis

This example shows that multiscale analysis can be conducted with HiDeNN by augmenting it with a sub-neural network. In general, multiscale analysis predicts the material response at relatively large length scales (the “part scale”) considering contribution from significantly smaller scales (the “microscale”). For example, the behaviors of composite materials are affected by inhomogeneous distribution of the reinforcement phase, as discussed in [55].

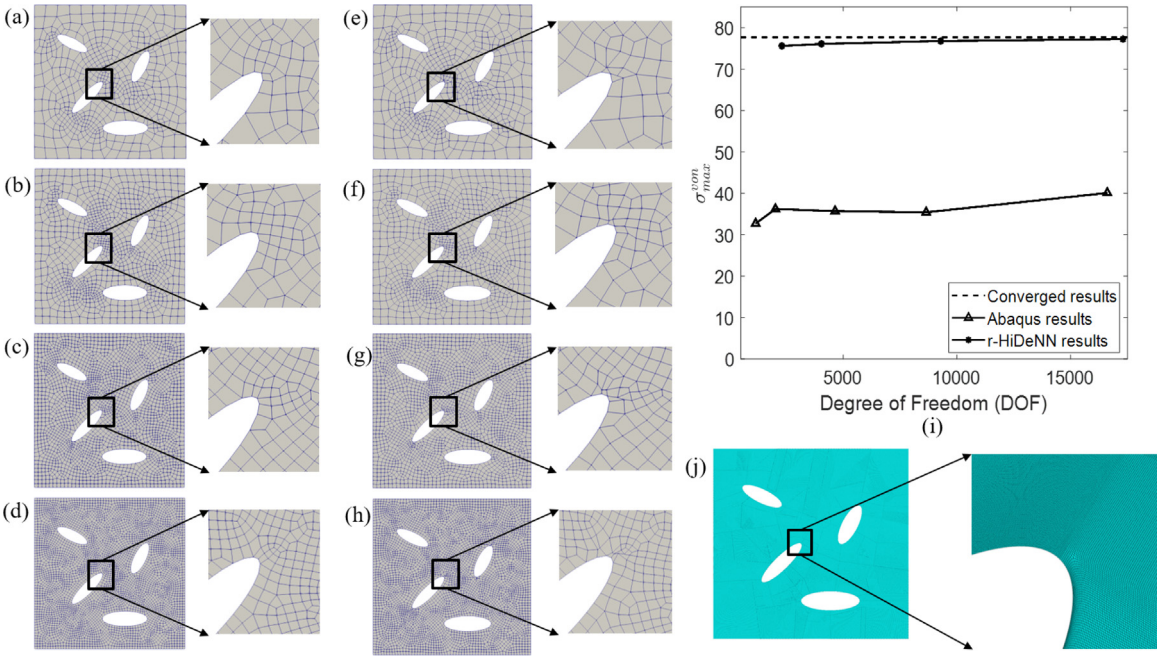

Fig. 8. Comparison of the discretization between Abaqus and the HiDeNN for stress concentration regions after learning the nodal positions. Conformal mesh from Abaqus with (a) 1154 DOFs, (b) 2022 DOFs, (c) 4646 DOFs, (d) 8650 DOFs. Conformal mesh from HiDeNN with (e) 1154 DOFs, (f) 4044 DOFs, (g) 9292 DOFs, (h) 17 300 DOFs. (i) Maximum von Mises stress versus number of elements, (j) conformal mesh from converged solution with 7,748,156 DOFs.

Table 1
Summary of difference in maximum von Mises stress between the HiDeNN 2D solutions (with an initially uniform mesh) and the conforming mesh solution from FEM.

Type of analysis	Number of elements	Degrees of freedom (DOFs)	σ_{max}^{von}	Difference	CPU-based computational time (s)
Abaqus (reference solution)	3 867 168	7 748 156	**77.7**	–	**1334.37**
Abaqus	524	1154	32.7	57.92%	–
r-HiDeNN	524	2308	**75.6**	**2.70%**	**17.53**
Abaqus	938	2022	36.2	53.41%	–
r-HiDeNN	938	4044	**76.1**	**2.06%**	**22.87**
Abaqus	2194	4646	35.7	54.05%	–
r-HiDeNN	2194	9292	**76.8**	**1.16%**	**31.09**
Abaqus	4143	8650	35.4	54.44%	–
r-HiDeNN	4143	17 300	**77.3**	**0.51%**	**40.25**

In the case of a fiber-reinforced composite, the variation of fiber volume fraction leads to variable material properties throughout the composite part. To account for this effect, multiscale analysis can be used to capture local microstructure. We examine the capability of HiDeNN to conduct multiscale analysis in the following sample multiscale problem. Note that the implementation and training for the multiscale problem is conducted on the same GPU as the problem in Section 3.1.

The most common scale-coupling technique for heterogeneous materials is the FE^2 approach proposed by Feyel and Chaboche [56]. In this method, the macroscopic material model is replaced by solving a representative volume element (RVE) model at the microscale, which takes strains/strains as boundary conditions, estimates the average material response, and passes that back to the macroscale material point. The micro-analysis is performed on the

RVE by solving

$$\begin{cases} div(\boldsymbol{\sigma}(\boldsymbol{x})) = 0, \ in \ \Omega \\ \boldsymbol{\sigma}(\boldsymbol{x}) = \boldsymbol{C^0}(\boldsymbol{x}) : \boldsymbol{\varepsilon}(\boldsymbol{x}) + \boldsymbol{p}(\boldsymbol{x}) \end{cases} \tag{4}$$

where $\boldsymbol{\sigma}(\boldsymbol{x})$ and $\boldsymbol{\varepsilon}(\boldsymbol{x})$ represent the micro-stress and micro-strain field, respectively. The stiffness of reference material is given by $\boldsymbol{C^0}(\boldsymbol{x})$, and $\boldsymbol{p}(\boldsymbol{x})$ denotes the so-called polarization stress. Once the microscale analysis is completed, the constitutive behavior at macroscale can thus be obtained by

$$\begin{cases} \bar{\boldsymbol{\varepsilon}} = \frac{1}{|\Omega|} \int \boldsymbol{\varepsilon}(\boldsymbol{x}) d\boldsymbol{x} \\ \bar{\boldsymbol{\sigma}} = \frac{1}{|\Omega|} \int \boldsymbol{\sigma}(\boldsymbol{x}) d\boldsymbol{x} \end{cases} \tag{5}$$

FE^2 solves the microanalysis of RVE model at each integration point with the FEM. This makes the computational cost extremely high for concurrent multiscale analysis.

To accelerate the solution process of Eq. (4), the self-consistent clustering analysis (SCA) is proposed by Liu et al. [47] and applied to multiscale modeling in [57]. SCA is a two-step reduced order modeling (ROM) process with a training step and a prediction step. The training step has three stages: data collection (e.g., finding microstructure elastic responses for each material point in the RVE), unsupervised learning (the clustering process which all material points in the RVE are grouped into N_c clusters), and computation of the interaction tensor. During the prediction step, the clusters and interaction tensors are used with the reduced Lippmann–Schwinger equation to predict the response of the microstructure to loading applied via macroscopic deformations subject to periodic boundary conditions. In practice, this involves solving an incremental form of the discretized Lippmann–Schwinger equation, given by (6),

$$\Delta\boldsymbol{\varepsilon}^I = \Delta\bar{\boldsymbol{\varepsilon}} - \sum_{J=1}^{N_C} \boldsymbol{D}^{IJ} : \left(\Delta\boldsymbol{\sigma}^J - \boldsymbol{C}^0 : \Delta\boldsymbol{\varepsilon}^J\right), \forall I \in \{1, \ldots, N_C\} \tag{6}$$

where $\Delta\varepsilon^I(x)$ represents the incremental strain for cluster I, $\Delta\bar{\boldsymbol{\varepsilon}}$ represents the applied macroscopic incremental strain on the RVE, N_c is the number of clusters, D^{IJ} denotes the interaction tensor, $\Delta\sigma^J$ and $\Delta\varepsilon^J$ are the incremental stress and strain for cluster J. The computational cost of this method compared to a direct solve for the RVE (e.g. with a Fast Fourier Transform-based method) is significantly reduced, because the clustering discretization greatly reduces the number of equations that must be solved. For the comprehensive details of SCA, and a derivation of Eq. (6), readers are referred to [32,47,48].

The HiDeNN applied to this example has sub-neural networks (neural networks that determine the parameters to use for neurons of a larger network) to conduct RVE analyses from data generated by the SCA. This follows the pattern of FE^2, where sub-FEM solutions are used at each material point. As shown in Fig. 9, the neural network developed in Section 3.1 is used to solve the response at macroscale with the loss function of Eq. (1a). At the microscale, a number of sub-neural networks are developed to solve Eq. (6). The loss function at microscale is defined as the residual of the governing equation in Eq. (6). By combining the loss function in Eq. (1a) and the microscale loss function based on Eq. (6), the multiscale analysis is carried out by concurrently solving the hierarchical neural networks. This is done by solving the following loss function in Eq. (7). Note that superscript (1) denotes the macroscale and superscript (2) denotes the microscale.

$$\begin{aligned} &\mathscr{L}(\boldsymbol{u}^h, \boldsymbol{\varepsilon}^{I,(2)}, \lambda^{(2)}) = \mathscr{L}^{(1)} + \lambda^{(2)}\mathscr{L}^{(2)} \\ &= \frac{1}{2}\int_\Omega \sigma(\boldsymbol{u}^h) : \varepsilon(\boldsymbol{u}^h) d\Omega - \left(\int_\Omega \boldsymbol{u}^h \cdot \boldsymbol{f} d\Omega + \int_{\Gamma_t} \boldsymbol{u}^h \cdot \bar{\boldsymbol{t}} d\Gamma\right) \Rightarrow \text{scale 1} \\ &+ \lambda^{(2)}\left(\frac{1}{N_c^{(2)}} \sum_{I=1}^{N_c^{(2)}} \left| \Delta\varepsilon^{I,(2)}(\boldsymbol{x}) + \sum_{J=1}^{N_c^{(2)}} \boldsymbol{D}^{IJ,(2)} : \left[\Delta\sigma^{J,(2)} - \boldsymbol{C}^0 : \Delta\varepsilon^{J,(2)}\right] - \Delta\bar{\boldsymbol{\varepsilon}} \right|^2 + \left| \sum_{I=1}^{N_c^{(2)}} c^{I,(2)} \Delta\varepsilon^{I,(2)} - \Delta\bar{\varepsilon} \right|^2 \right) \\ &\Rightarrow \text{scale 2} \end{aligned} \tag{7}$$

where $\boldsymbol{u}^h$ is the displacement in the macroscale (scale 1), and $\boldsymbol{\varepsilon}^{I,(2)}$ is the cluster-wise strain tensor in the microscale (scale 2), $\lambda^{(2)}$ is the Lagrangian multiplier applied on the loss function contributed by the microscale, $c^{I,(2)}$ is the volume fraction of cluster I in the microscale, $\mathscr{L}^{(1)}$ is the macroscale loss function (scale 1), $\mathscr{L}^{(2)}$ is the microscale

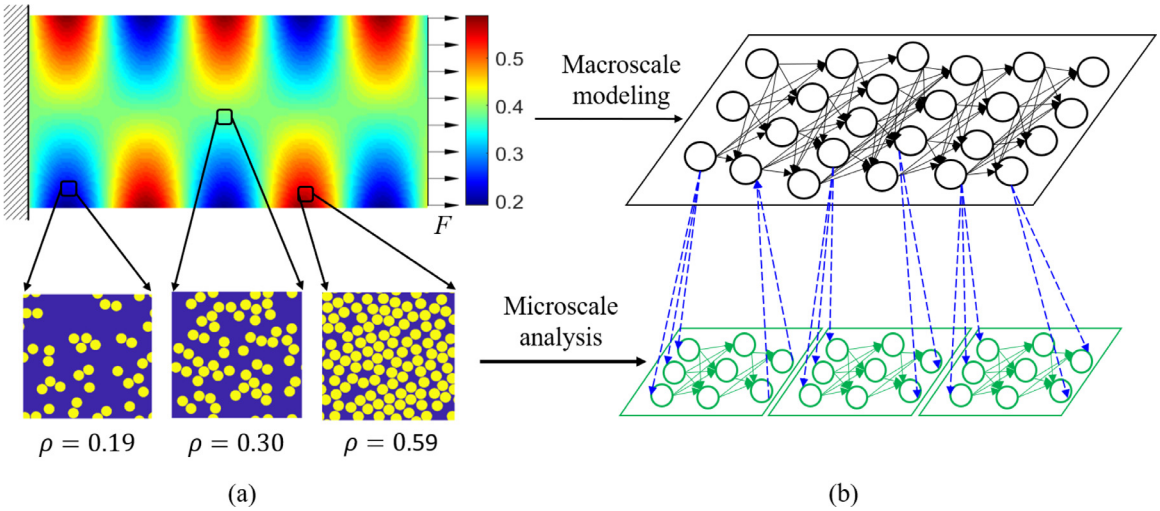

Fig. 9. HiDeNN framework of two-scale analysis of fiber-enforced composite materials. (a) A tensile bar with spatially varying fiber density at the microscale. The left side of the bar is fixed while a uniform loading is subjected to the right side of the bar. (b) A schematic of the hierarchical neural network for two-scale analysis. The top neural network is for macroscale analysis as shown in 4, while the bottom sub-neural networks are used for the micro-analysis of different RVE models shown in Fig. 10.

loss function (scale 2). The loss function, a summation of different loss functions, constructs the concurrent multiscale modeling using HiDeNN framework.

In HiDeNN, any parameter is treated as an input dimension in the parameter space of D, (refer to Fig. 2). For functionally distributed fiber-reinforced composite analysis, the primary parameter is fiber volume fraction.

Using SCA, RVEs with different volume fractions are analyzed to form a training data set. Afterwards, a feed forward neural network (FFNN) with the parameter of fiber volume fraction as the input is trained to rapidly compute the microscale response. Fig. 10 presents the FFNN for the microscale analysis. The inputs of the FFNN include macroscale strains and the fiber volume fraction and the outputs are the homogenized stress of the RVE model at macroscale integration points.

As a demonstration, 2D tensile bar under plane stress consisting of graded microstructure is modeled using the structured HiDeNN for two-scale analysis. Fig. 9 shows the 2D model, the functional distribution of fiber-reinforced microstructures in a bar, as well as the microstructure of the RVE for different volume fractions. The dimension of the bar is 2.0 by 1.0 Fixed boundary conditions are applied at the left side of the bar and a uniform tension F of 10^5 is applied in the $+x$ direction at the right side of the bar. The volume fraction of the fiber is distributed following the function of $\rho(x, y) = (y - 0.5)sin(8x)$; high volume fraction and low volume fraction RVEs are periodically distributed along the x direction. The aim is to induce multiple stress concentrations and investigate the performance of the proposed methodology in detecting the stress concentration by learning the problem with HiDeNN. To examine the performance, the problem is solved with two different discretization strategies: one with 320×160 quadrilateral elements and another with 80×40 quadrilateral elements using the finite element method, with the material response coming from SCA. For comparison with FE-SCA, the proposed two-scale HiDeNN network is solved for the model with mesh of 80×40 four-nodes elements by training both the nodal displacements and nodal positions simultaneously.

To solve this problem, 30 different RVEs with volume fraction ranging from 0.19 to 0.59 are solved using SCA. It takes 2513 s to solve for each RVE and around 5 s to train the neural network. MATLAB is used for training.

The results of the HiDeNN model for the functional composite are given in Fig. 11 including both the nodes and von Mises stress, σ^{von}, distributions. Due to the microstructure variation there are five stress concentrations at macroscale. Even with a coarse mesh, the HiDeNN network can obtain a result within 1% of the fine mesh solution obtained with FEM by moving nodes to stress concentration regions (0.480 for HiDeNN versus 0.481 for the fine mesh FEM solution).

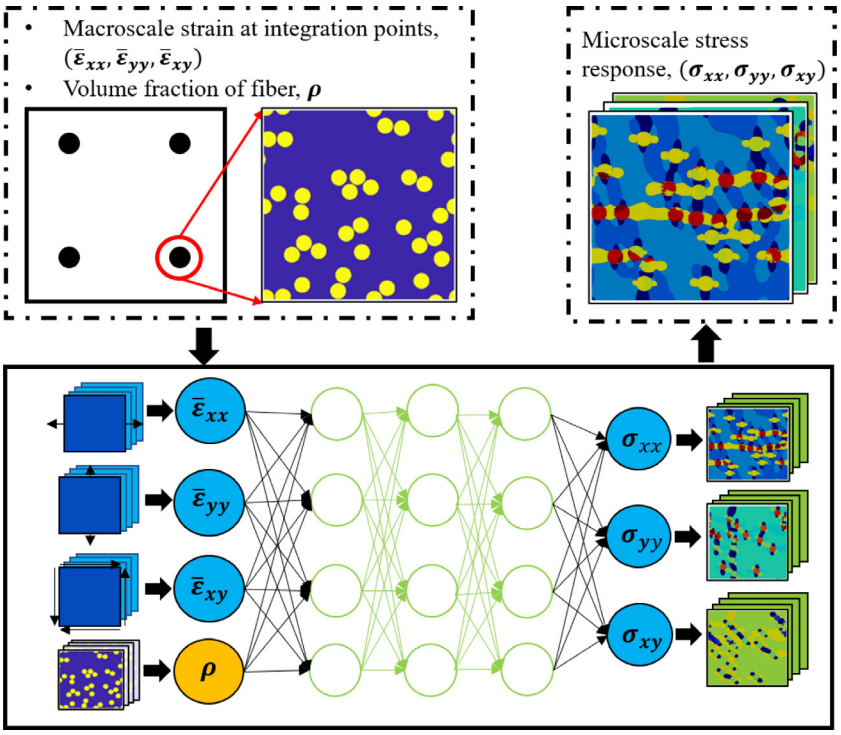

Fig. 10. The sub-neural networks of performing the analysis for RVE model attributed to the interpolation points at microscale. The input of the sub-neural networks includes the macroscopic strain at integration points and the volume fraction of the fiber. The output of the neural networks are the microscale stresses of the RVE model.

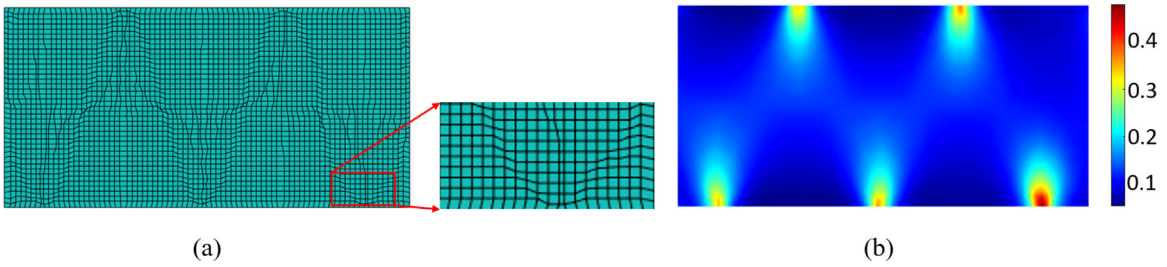

Fig. 11. Discretization and stress distribution obtained by multiscale HiDeNN framework. (a) Final state of the discretization after learning the optimal nodal position by HiDeNN; (b) Stress distribution by using the HiDeNN to solve the functionally distributed fiber-enforced composite. The maximum stress is 0.480 by using HiDeNN analysis. The difference is 0.3% comparing with the converged value obtained by four times fine mesh while the difference from same uniform mesh is 9.71%.

3.3. HiDeNN for multivariate system: discovery of governing dimensionless numbers from data

The HiDeNN can handle data in a high-dimensional parametric space, $p_1 \sim p_n$ as shown in Fig. 2. However, a large number of input parameters often causes two severe problems: first, the number of data required for training the network exponentially increases with the dimensionality of the inputs, i.e., the curse of dimensionality [58]; second, a large number of parametric inputs with complex dependencies and interactions could significantly degrade the capability of extrapolation and prediction of the network [59]. In order to reduce the dimensionality of the input parameters such that HiDeNN can be applied to a wide range of science and engineering problems, we propose a dimensionally invariant deep network (DimensionNet) embedded in the HiDeNN. The DimensionNet reduces the dimensionality of the original input parameters by automatically discovering a smaller set of governing

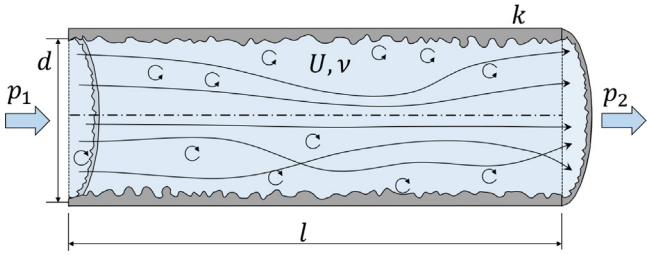

Fig. 12. Schematic of the fluid flow in a rough pipe with dimensional quantities, including inlet pressure p_1, outlet pressure p_2, pipe diameter d, pipe length l, average steady-state velocity U, kinematic viscosity ν, and surface roughness Ra.

dimensionless numbers and transforming the high-dimensional inputs to the dimensionless set. The DimensionNet can identify appropriate pre-processing functions and parameter layers for HiDeNN.

To illustrate the performance and features of the proposed DimensionNet we use it to "rediscover" well-known dimensionless numbers, e.g., Reynolds number (Re) and relative roughness (Ra^*), in a classical fluid mechanics problem: laminar to turbulent transition in rough pipes [60–63]. We use the experimental data collected by Nikuradse [61] to demonstrate that the proposed DimensionNet can recreate the classical governing dimensionless numbers and scaling law.

A schematic of turbulent pipe flow is presented in Fig. 12. The dependent parameter of interest is the dimensionless resistance factor λ that can be expressed as [61]

$$\lambda = \frac{p_1 - p_2}{l}\frac{2d}{\rho U^2}, \tag{8}$$

where $p_1 - p_2$ represents the pressure drop from inlet to outlet, l is the length of the pipe, d is the diameter of the circular pipe, ρ is the density of fluid and U measures the average velocity over a steady-state, i.e., fully-developed, section of the pipe.

We postulate that the resistance factor λ depends on four parameters: the steady-state velocity of fluid U, kinematic viscosity ν, pipe diameter d, and surface roughness of the pipe Ra: $\lambda = f(U, \nu, d, Ra)$.

We assume that there are only two governing dimensionless parameters in this system (the maximum number of the governing dimensionless parameters can be determined by dimensional analysis). To discover these two dimensionless combinations from the dataset, we take the experimental data [61] with various U, ν, d, Ra as the four inputs of the DimensionNet, and $\log(100\lambda)$ as the output to be consistent with the original results,

$$\{\boldsymbol{p}\}_{n=1}^{N} = \{U_n, \nu_n, d_n, Ra_n\}_{n=1}^{N} \tag{9}$$

$$\{u\}_{n=1}^{N} = \{[\log(100\lambda)]_n\}_{n=1}^{N} \tag{10}$$

where $\boldsymbol{p}$ is the dimensional parametric input and u is a solution as shown in Fig. 2, the subscript n denotes the nth data point and runs from 1 to N, which is the total number of the data points. The 448 data points used are divided into 359 the training set for training the parameters of regression models and 89 test set for evaluating the performance of the models.

A schematic of DimensionNet is shown in Fig. 13. The proposed DimensionNet includes two parts:

- A scaling network used to discover explicit form of hidden dimensionless number(s). The scaling network corresponds to the parameter layers in Fig. 2.
- A deep feedforward network represents nonlinear correlations, i.e., similarity function, between the dimensionless numbers. The deep network corresponds to the PHY-NN or EXP-NN in Fig. 2.

As shown in Fig. 13, the first layer of the scaling network constructs several dimensionless parameters, Π_{bj}, as a set of basis via weights $\boldsymbol{w}^{(1j)}$ as (there is no bias used in the scaling network)

$$\Pi_{bj} = \sum_i w_i^{(1j)} \log(p_i) = \log\left(\prod_i p_i^{w_i^{(1j)}}\right) \tag{11}$$

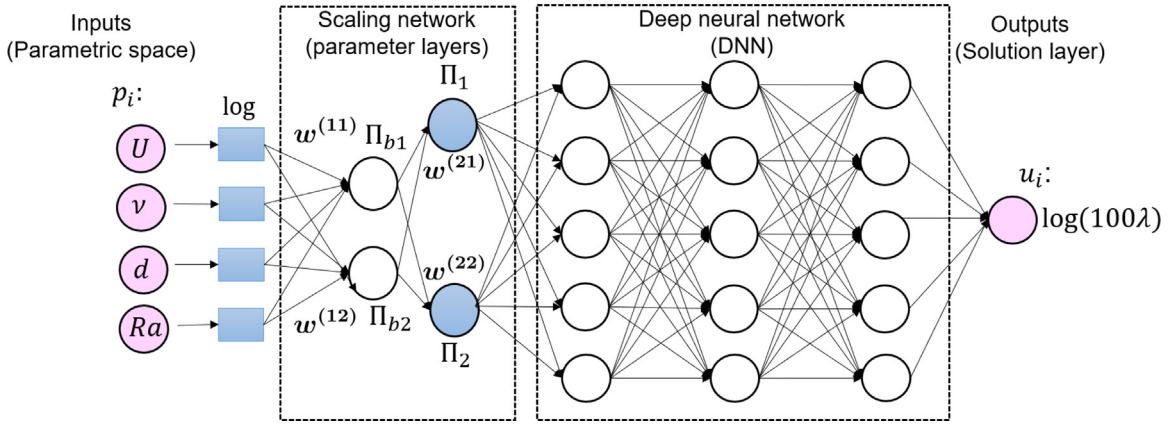

Fig. 13. Schematic of the dimensionally invariant deep network (DimensionNet). The four inputs are $p_1 = U$, $p_2 = \nu$, $p_3 = d$, and $p_4 = Ra$. The output is $u = \log(100\lambda)$. The number of neurons at each layers depends on the applied problem. The network structure presented in this figure is just for illustration.

The weights of the first layer $\boldsymbol{w}^{(1j)}$ can be predetermined from the dimensional matrix B, in which the rows are the dimensions and the columns are the input parameters. For example, the dimensional matrix B for the pipe flow problem is expressed as

$$\mathrm{B} = \begin{matrix} \\ [\mathrm{L}] \\ [\mathrm{T}] \end{matrix} \begin{matrix} \begin{matrix} U & \nu & d & Ra \end{matrix} \\ \begin{bmatrix} 1 & 2 & 1 & 1 \\ -1 & -1 & 0 & 0 \end{bmatrix} \end{matrix}$$

where [L] and [T] are fundamental dimension: length and time, respectively.

To make sure the Π_{bj} are dimensionless, the weights of the first layer should satisfy

$$\mathrm{B}\boldsymbol{w}^{(1j)} = 0 \tag{12}$$

There are infinitely many vectors that yield this condition, and actually they span a two-dimensional space (in this example). We arbitrarily choose two of them as basis vectors of this two-dimensional space and they are the weights of the first layer of the DimensionNet,

$$\boldsymbol{w}^{(11)} = [1 \ -1 \ 0 \ 1]^T \tag{13}$$

$$\boldsymbol{w}^{(12)} = [2 \ -2 \ 1 \ 1]^T \tag{14}$$

The set of dimensionless basis, Π_{bj}, then create new dimensionless parameters at the second layer via weights $\boldsymbol{w}^{(2j)}$ by

$$\Pi_j = \sum_i w_i^{(2j)} \log\left(\Pi_{bi}\right) = \log\left(\prod_i \Pi_{bi}^{w_i^{(2j)}}\right) \tag{15}$$

Since we use linear activation function in scaling network, the scaling weights, $\boldsymbol{w}^{(1)}$ and $\boldsymbol{w}^{(2)}$, can be defined by linearly combining weights from the first and second layers:

$$\boldsymbol{w}^{(1)} = [\boldsymbol{w}^{(11)} \ \boldsymbol{w}^{(12)}]\boldsymbol{w}^{(21)} \tag{16}$$

$$\boldsymbol{w}^{(2)} = [\boldsymbol{w}^{(11)} \ \boldsymbol{w}^{(12)}]\boldsymbol{w}^{(22)} \tag{17}$$

Thus, the dimensionless parameters Π_j can be represented by inputs, $\boldsymbol{p}$, and scaling weights, $\boldsymbol{w}^{(j)}$, as

$$\Pi_j = \log\left(\prod_i p_i^{w_i^{(j)}}\right) \tag{18}$$

The deep feedforward network maps the Π_j to the dependent output u. Any inherent nonlinear relationship $f(\cdot)$ can be captured since the universal approximation capability of deep neural networks [64]. The output of the

DimensionNet can be expressed as

$$\log(100\lambda) = f(\log(\Pi_1), \log(\Pi_2)) = f\left(\log\left(\prod_i p_i^{w_i^{(1)}}\right), \log\left(\prod_i p_i^{w_i^{(2)}}\right)\right) \tag{19}$$

Two objectives can be achieved by training the DimensionNet: first, identify the weights of the second layer $\boldsymbol{w}^{(2j)}$ such that the expression of the hidden dimensionless parameters Π_j can be quantified by Eq. (15); and second, train the weights and biases in the deep neural network (DNN) to represent the nonlinear function $f(\cdot)$ such that the difference between the network output and the dependent parameters of interests is minimized. The proposed loss function of the DimensionNet is

$$\mathscr{L} = \frac{1}{N} \parallel \boldsymbol{u} - \hat{\boldsymbol{u}} \parallel_2^2 + \beta_1 \parallel \boldsymbol{w}^{(1)} \parallel_1 + \beta_2 \parallel \boldsymbol{w}^{(2)} \parallel_1 \tag{20}$$

where the first term indicates the mean square error (MSE), N is the number of training data points, $\boldsymbol{u}$ is the output vector of the DimensionNet including $\{u\}_{n=1}^{N}$, $\hat{\boldsymbol{u}}$ is the corresponding measurements of the dependent parameters. The second and third terms indicate the L1 norm of the scaling weights of the scaling network, and β_1 and β_2 are hyper-parameters that determine the relative weighting of the three terms in the loss function. The loss function encourages the DimensionNet to minimize the MSE error and to use the minimal number of input parameters for the representation of data.

We train the DimensionNet using the Adam optimizer [52]. Weights of the scaling network are randomly initialized between -1 and 1 before training. In addition to the loss function weightings ($\beta_1 = \beta_2 = 5.0 \times 10^{-4}$ in this case), there are several other hyper-parameters including learning rate (3.0×10^{-3}), decay rate (0.3), decay step (100), the number of epochs (400), and the number of dimensionless parameters (2). For the deep feedforward network, we use 4 fully-connected layers (10 neurons) with biases and Rectified Linear Unit (ReLU) activation functions. The choice of hyper-parameters affects the accuracy and efficiency of the method. In this study, we determined those hyper-parameters by trial and error, based on our experience. Optimization of the hyper-parameters for data-driven models is a very important topic. Bayesian optimization is a promising method to determine those hyper-parameters. We save 16 709 snapshot results and get 3968 points which have high R^2 (greater than or equal to 0.98). Then we use the Bayesian information criterion (BIC) [65] to select the parsimonious model that has the best predictive capability but with the minimal non-zero parameters. The expression of the BIC used in this study is

$$\mathrm{BIC} = \mathrm{N} \cdot \ln \delta_\epsilon^2 + \mathrm{e}^{\mathrm{k}} \cdot \ln \mathrm{N} \tag{21}$$

where N is the number of data points used in training or testing procedure, ϵ is the residual represented by $\epsilon = u - \hat{u}$, δ_ϵ^2 is the variance of the residuals, and k is the number of non-zero components of the scaling weights, i.e., $\boldsymbol{w}^{(1)}$ and $\boldsymbol{w}^{(2)}$. It is noted that in order to enhance the effect of the number of parameters we use e^{k} rather than k in the original BIC expression [65].

To figure out the optimal combinations of dimensional inputs, the results of identified weights at the second layer, i.e., $\boldsymbol{w}^{(21)}$ and $\boldsymbol{w}^{(22)}$ are shown in Fig. 14. The x and y axes indicate the first and second components of a weight, respectively. Fig. 14(a)–(d) presents the selected results with different training BIC thresholds. It is noted that smaller BIC means that the model has better balance between accuracy (R^2) and complexity (the number of non-zero parameters). The results in Fig. 14(d) show the parsimonious models with the smallest BIC. As shown in Fig. 14(d), most of the scaling weights converge to two lines: $w_y = -w_x$ and $w_y = -2w_x$. Thus, based on Eq. (15) the governing dimensionless numbers have the form

$$\Pi_1 = \log\left(\frac{Ud}{\nu}\right) \tag{22}$$

$$\Pi_2 = \log\left(\frac{Ra}{d}\right), \tag{23}$$

Interestingly, the dimensionless numbers identified with the DimensionNet from data perfectly match those discovered "manually" in the 1950s [61] (The log function in Eqs. (22) and (23) can be vanished by using exponential activation functions for the neurons at the second layer of the scaling network). They are the well-known Reynolds number and relative surface roughness [61]

$$\mathrm{Re} = \frac{Ud}{\nu} \tag{24}$$

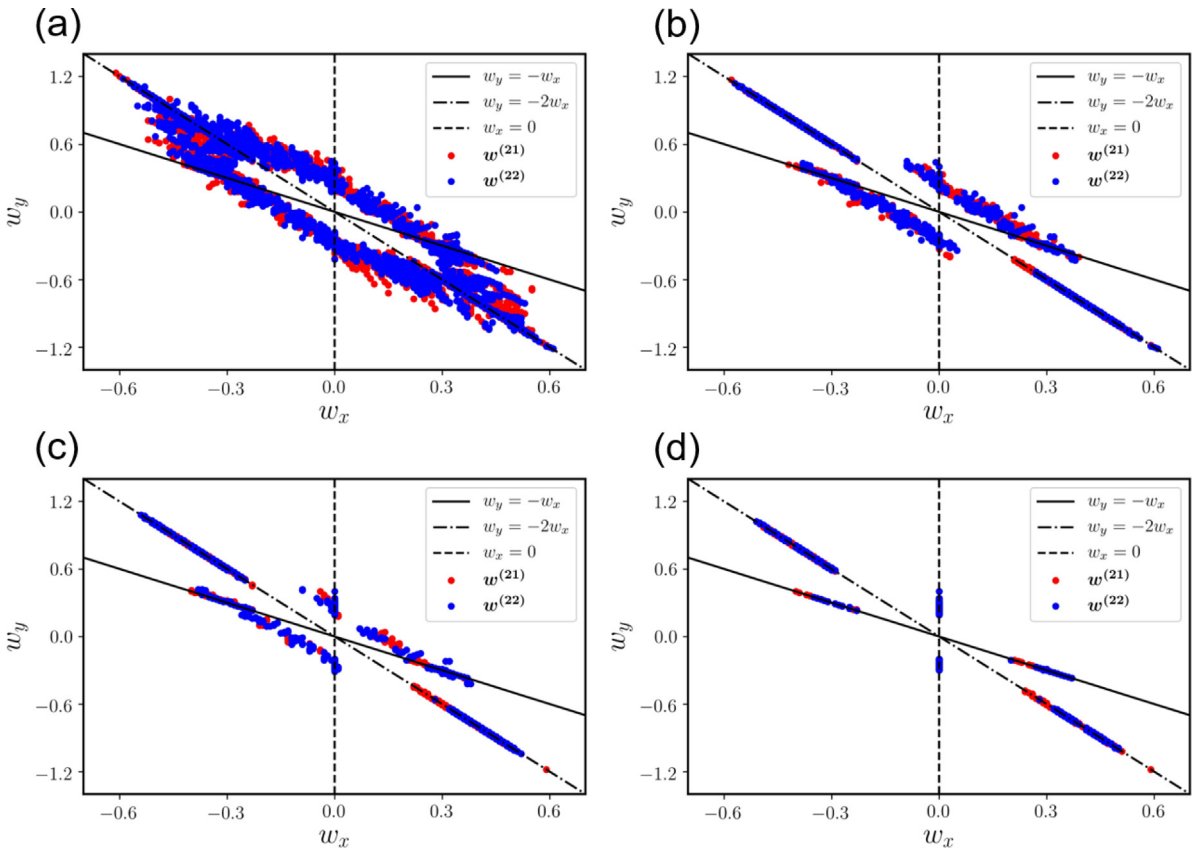

Fig. 14. Distribution of the identified weights of the basis, i.e., $\boldsymbol{w}^{(21)}$ and $\boldsymbol{w}^{(22)}$ from snapshot results with high R^2 (greater than or equal to 0.98) and different training BIC thresholds: (a) no BIC threshold; (b) results with BIC$\leqslant 0$; (c) results with BIC$\leqslant -750$; and (d) results with BIC$\leqslant -1500$.

$$Ra^* = \frac{Ra}{d} \tag{25}$$

The scaling law or similarity function captured by the DimensionNet can be expressed as

$$\log(100\lambda) = f\left(\log\left(\frac{Ud}{\nu}\right), \log\left(\frac{Ra}{d}\right)\right) = f\left(\log(\mathrm{Re}), \log\left(Ra^*\right)\right) \tag{26}$$

The coefficients of determination R^2 are shown in Fig. 15(a) for the training set and 15(b) for test set. The R^2 values are nearly 1, indicating the good predictive capability of the DimensionNet. Fig. 15(c) shows the two-dimensional pattern hidden in the original four-dimensional parametric space, and this low-dimensional pattern is governed by two identified dimensionless numbers, i.e., Reynolds number Re and relative roughness Ra^*. In this study, we assume the number of governing dimensionless parameters is known, but it does not have to be known for a general problem. If we do not know the number of dimensionless parameters, we would start at one and train the DimensionNet and see if the network can converge to a highly accurate result. If so, we conclude that there is only one governing dimensionless number in the problem. If not, we will set up one more dimensionless parameter and re-train the DimensionNet. We will repeat this procedure until we find a converged result. In this way, we can identify the number of governing dimensionless number for a problem or a system without governing equations.

Traditionally, the dimensionless numbers are identified by dimensional analysis [66] or from normalized governing equations [67]. However, for many complex systems the optimal dimensionless numbers cannot be determined by using dimensional analysis alone [68], and for many applications we do not have well-tested governing equations of the problems or only know part of them. For those problems, we can alternatively use the proposed DimensionNet to discover the governing dimensionless numbers purely from data. The identified

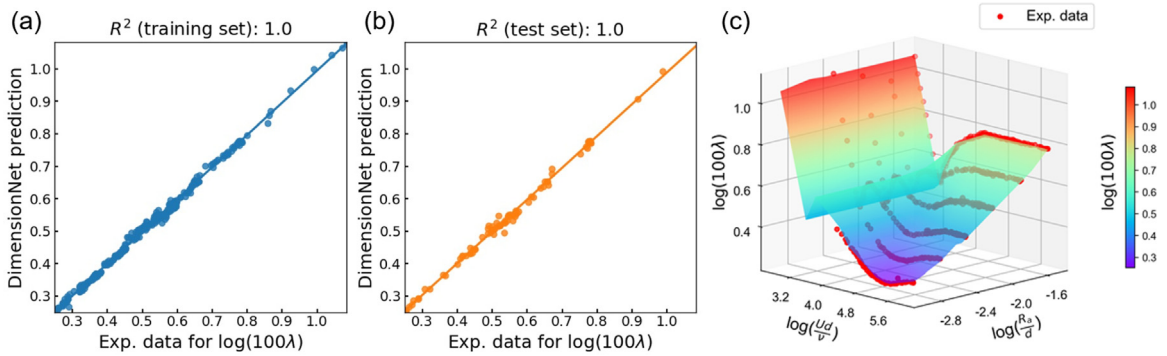

Fig. 15. Comparison between experimental data and DimensionNet prediction: (a) R^2 for the training dataset; (b) R^2 for the test dataset; and (c) captured relationship between identified dimensionless numbers and resistance factor. The points represent experimental data [61] and the surface represents the DimensionNet result.

smaller set of dimensionless numbers informs HiDeNN such that it can predict more complex behaviors of the problems in a more accurate and efficient manner. The DimensionNet involves the principle of the similitude and dimensional invariance [67]. It can eliminate the physically inherent dependency between the dimensional input parameters without any loss of accuracy, and thus has better extrapolation capability than traditional dimensionality reduction method such as principal component analysis (PCA).

The proposed DimensionNet is a very general tool and thus can be applied to many other physical, chemical and biological problems where abundant data are available but complete governing laws and equations are vague [69]. The identified reduced parameter list can be used as the input to the HiDeNN. It can significantly improve the efficiency and interpretability of the network and avoid overfitting by reducing the input space and dependency.

4. Extension of HiDeNN to solve challenging problems

This section demonstrates a typical AI solution method for one example of each *type* of the challenging problems introduced in Section 1, and make note of challenges with these existing methods that might be mitigated by using HiDeNN.

4.1. Type 1: Purely data-driven problems

The case study involves finding the salient relationship between the local thermal history and ultimate tensile strength in a thin wall built by direct energy deposition with Inconel 718 alloy. In this case, we assume there is no known physical law connecting these two factors; thus, an AI/ML method is used to infer the relationship.

Fig. 16 shows the framework used in this example. Infrared (IR) imaging records the thermal history for each point in an AM built thin wall. A total of 12 such walls are considered for the study each having 120 layers with 0.5 mm layer height. Details of the experiments are reported in [70]. A total of 135 temperature–time histories and corresponding ultimate tensile strength are accumulated as samples.

Because of the high dimensional nature of collected thermal histories, a *binning* technique for dimension reduction is applied as shown in Fig. 17(a). The total temperature range from the first peak to the end from all the collected samples are divided into bins of 50 °C and time spent (in seconds) in all those bins are considered as features. In this way, the continuous thermal histories are converted into an $N \times M$ matrix where N is the number of samples and M is the number of total bins (see Fig. 17(b)). The corresponding ultimate tensile strength of the AM parts are collated into a $N \times 1$ vector. Using the binned data, a Random Forest (RF) regression [71] supervised machine learning is used to link the reduced thermal history with mechanical performance. The coefficient of determination R^2 with 95% confidence interval for both training and testing data (split as 80% training–20% testing) are shown in Fig. 18. Increasing the number of considered features (number of bins in the input matrix), tends to increase the R^2 both in training and testing. Thirty five features completely describe all the thermal histories in the sample. The training time for the random forest algorithm is 0.18 s on a 2.3 GHz Intel Core i5 processor.

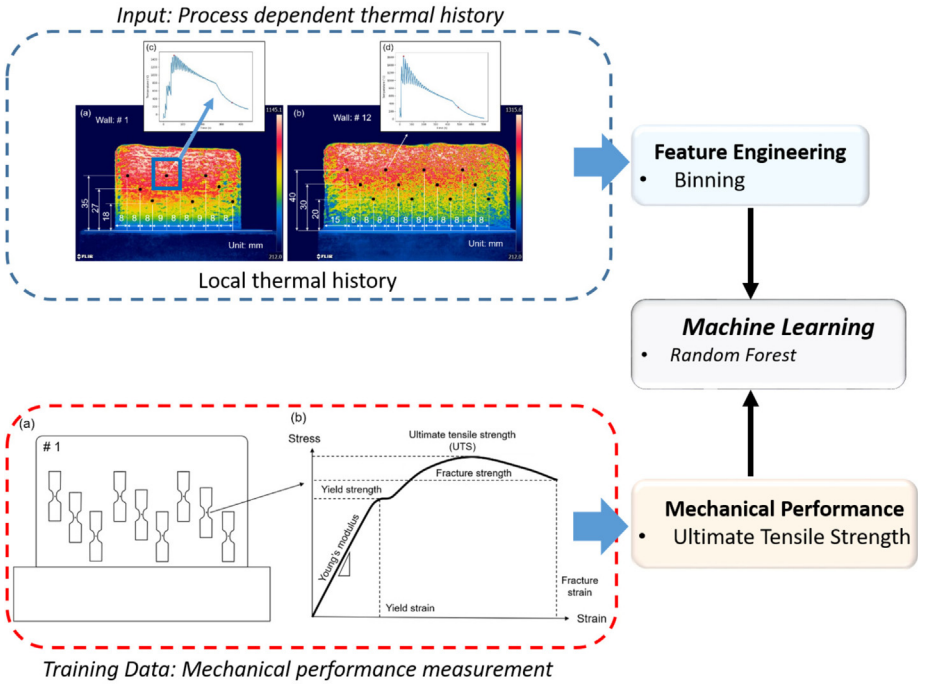

Fig. 16. Schematic diagram outlining the AI framework to link thermal history with part performance. Extracted local thermal history at designated points are reduced by binning. The target database is built using ultimate tensile strength obtained by experiments. Machine learning training uses Random Forest algorithm.

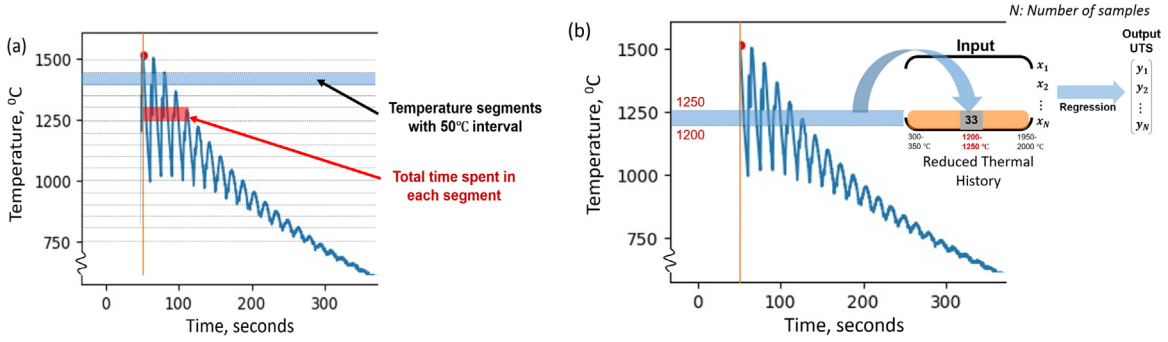

Fig. 17. (a) Binning technique used in this work. The thermal history is divided into bins of 50 °C. (b) Time spent in each bin is taken as input and output vector consists of ultimate tensile strength.

Although this AI approach can capture very complex relationships between temperature history and ultimate tensile strength in AM, our model overfits the data as indicated by the difference of R^2 between training and test datasets as shown in Fig. 18. Hence, an alternative dimension reduction or regression method is required. With no prior knowledge it is hard to decide which AI tool or technique should be chosen.

We can use the HiDeNN framework to solve this problem and obtain insight on the governing physics as shown in Fig. 19. To solve this example, the HiDeNN will consist of input layer (location, time, temperature, and manufacturing process parameter (such as scan speed) as inputs), pre-processing functions, EXP-NN layer, solution layer, and operation layers. If we look back to Section 2, spatial location is Ω, processing history is t, and manufacturing conditions are p. The pre-processing function can be designed so that high-dimensional thermal history data at different locations on the wall becomes tractable for learning algorithms. A candidate function for this operation can be continuous wavelet transformation function. The combination of solution and operation layers

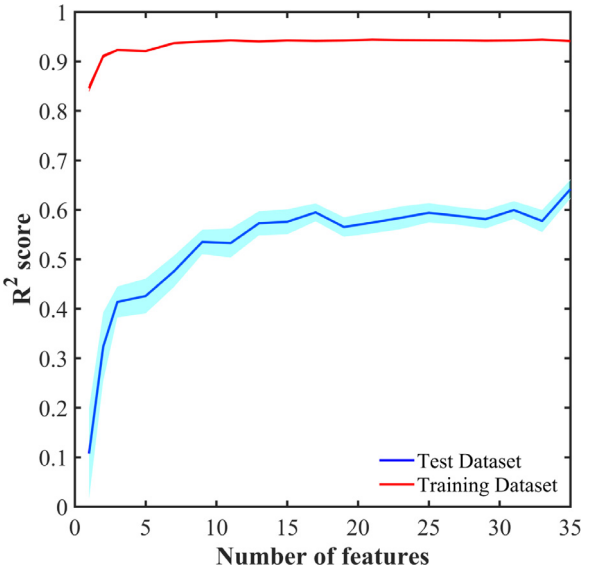

Fig. 18. Variation in the R^2-score with considered number of features in Random Forest algorithm. Shaded regions represent 95% confidence interval.

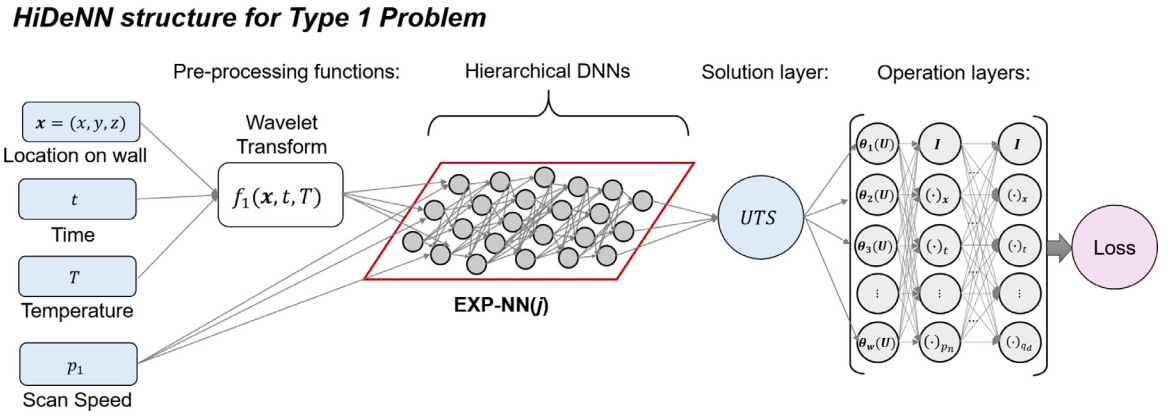

Fig. 19. Proposed HiDeNN framework to solve Type 1 problem. The input spatiotemporal variables are first pre-processed by wavelet transform before entering DNN. The DNN uses those processed inputs and parameters to predict the UTS, and the UTS will enter the loss function for training variables in DNN to improve prediction.

will discover unknown physics relating thermal history and mechanical property in AM built wall. The loss function can be defined as,

$$\mathcal{L} = \frac{1}{N_T}\sum_{i=1}^{N_T}\left(UTS_{exp} - UTS_i\right)^2 + \lambda\|\mathbf{P}(\mathbf{x}, t, T, UTS_{exp}) - \Theta(\mathbf{x}, t, T, UTS_{exp})\|_2 \tag{27}$$

where, $\mathcal{L}$ is the loss function, N_T is the number training samples, UTS_{exp} is the ultimate tensile strength from experimental observations, UTS_i is the predicted ultimate tensile strength from the HiDeNN, λ is the Lagrange multiplier, $\mathbf{P}(\mathbf{x}, t, T, UTS_{exp})$ is a function of operators and expressions such as addition, multiplication, differentiation, or integration, $\Theta(\mathbf{x}, t, T, UTS_{exp})$ is a function of position (location on the wall), time, and temperature, and $\|.\|_2$ is the L_2 norm. The first term of Eq. (27) comes from the hierarchical DNN layer while the second term comes from the operations layer. Combined minimization of these two terms with the Lagrange multiplier for the latter one will give us a mathematical expression for the relationship between spatiotemporal co-ordinates, temperature, and ultimate tensile strength, revealing unknown physics. One concern is that the

experimental data contain noise and uncertainty. In order to tackle this problem the hierarchical DNN layer can be a Bayesian neural network resulting in probabilistic terms in the mathematical expression. This will be a part of our future research on HiDeNN.

4.2. Type 2: Mechanistically insufficient problems with limited data

Type 2 problems are problems for which the available physical information is incomplete. For example, the governing equations may be known, but all the parameters in the governing equations are not explicitly identified. To illustrate, we present here how fatigue life of an AM part can be predicted from statistical information about microstructures with porosity. In this case, we know the governing physics of the problem on the continuum scale but there is limited data relating microstructural porosity and process parameters, and the spread in fatigue life is quite large making empirical fatigue predictions inaccurate. By incorporating experimental images directly higher simulation fidelity is achieved, with the trade-off of higher computational expense.

To predict fatigue response a computational crystal plasticity material law is used [72,73], which predicts the local cyclic change in plastic shear strain (denoted $\Delta\gamma^p$). This cyclic change saturates relatively quickly (up to 10 cycles may be needed, but in this case after about 3 or 4 cycles), and the saturated value is used as input to a microstructurally relevant Fatemi–Socie-type fatigue indicator parameter (FIP) for high cycle fatigue [74]. The FIP can be calibrated to fatigue life using, e.g., reference experimental data for the material of interest.

The crystal plasticity and FIP methods have been implemented in already explained Self-consistent Clustering Analysis (SCA) with crystal plasticity material law (termed CPSCA, as described in previous works [48,73,75,76]). Example images, a schematic of the solution method, and the resulting prediction of number of incubation cycles for an example microstructure from various possible images is shown in Fig. 20. For this model there are 16 clusters in the matrix phase and 4 in the void phase, selected to balance accuracy and computation cost based on prior experience with similar systems [48]. Constructing the "offline" data for each image in the SCA database cost about 200 s but need only be run once to provide a complete training set for all possible boundary conditions for that image using an implementation of the FFT-based elastic analysis in Fortran. The "online" part of SCA took about 15 or 20 s per loading condition per microstructure image to compute fatigue crack incubation life, N_{inc}, using crystal plasticity. While a comparison to an DNS solution with crystal plasticity has not been conducted for this case (see [48] for a more thorough analysis), this represents about a factor of about two speed up even when comparing an elastic analysis with DNS versus a full crystal plasticity analysis with the online SCA method for one loading condition. The more loading conditions required, the more favorable this comparison becomes for SCA as no re-training is required after the initial "offline" data is generated. The loading conditions shown are approximately uniaxial tension/compression in the vertical axis (extracted from a multiscale simulation, so uniaxiality is not fully guaranteed), specified via applied deformation gradient in each voxel. The resulting information can be used as training data for the HiDeNN, as shown mathematically in the loss function given in Eq. (28).

For this example, HiDeNN could be applied to construct a relationship between the process, experimental microstructural images, and material performance. The relationship can be regarded as a new material performance prediction formulation, where microstructural features can be directly considered by using a deep convolutional neural network (CNN) as the NN within HiDeNN for image feature identification. A proposed framework for solving this problem is shown in Fig. 21. For an AM build, the x and t can be location and process history. In the parametric input, we can consider the process parameters, basic material properties, and images of porosity (and potentially other microstructural features). The pre-processing functions are employed to prepare the 3D images for the CNN. The solution layer contains the fatigue life and the mechanical response of the RVE. The operation layer is designed to construct the loss function as,

$$\begin{aligned}\mathscr{L}(\boldsymbol{u}^h, \boldsymbol{\varepsilon}, \boldsymbol{N}_{inc}) =& \mathscr{L}^{(1)} + \mathscr{L}^{(2)} \\ =& \frac{1}{2}\int_{\Omega} \sigma(\boldsymbol{u}^h) : \varepsilon(\boldsymbol{u}^h) d\Omega - \left(\int_{\Omega} \boldsymbol{u}^h \cdot \boldsymbol{f} d\Omega + \int_{\Gamma_t} \boldsymbol{u}^h \cdot \bar{\boldsymbol{t}} d\Gamma\right) \Rightarrow \text{scale 1} \\ &+ \frac{1}{N_T^{(2)}}\left(\sum_{I=1}^{N_T^{(2)}} \left(N_{inc}^H - N_{inc}\right)\right)^2 \Rightarrow \text{scale 2}\end{aligned} \tag{28}$$

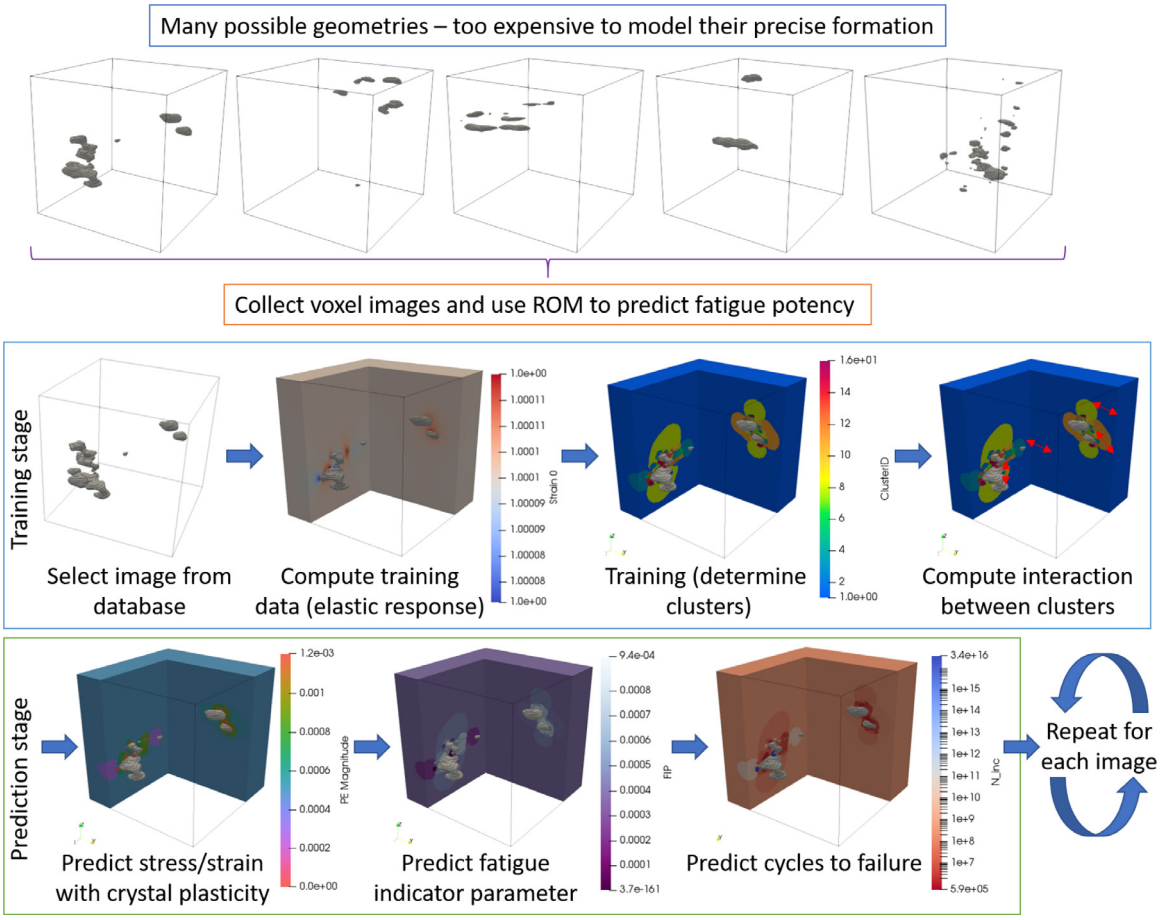

Fig. 20. Outline of the solution scheme used for this example of a type 2 problem, including example images and the process of CPSCA-based fatigue analysis.

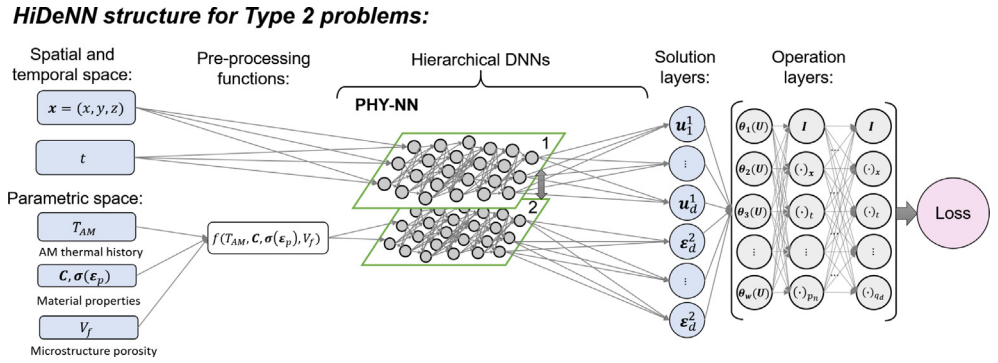

Fig. 21. Proposed HiDeNN framework for type 2 problem. The inputs will be different AM processing parameters and associated material and microstructure parameters. The HiDeNN structure could solve the macroscale problem, a AM process simulation, to obtain thermal history. Microscale could be solved for microstructure fatigue life based on the thermal history and material parameters.

where $\boldsymbol{u}^h$ is displacement history, $\boldsymbol{\varepsilon}$ is the applied strain history, N_{inc}^H is the fatigue crack incubation life computed from the HiDeNN and N_{inc} is the fatigue incubation life computed from CPSCA, $\mathscr{L}^{(1)}$ is the macroscale loss function (scale 1), $\mathscr{L}^{(2)}$ is the microscale loss function (scale 2).

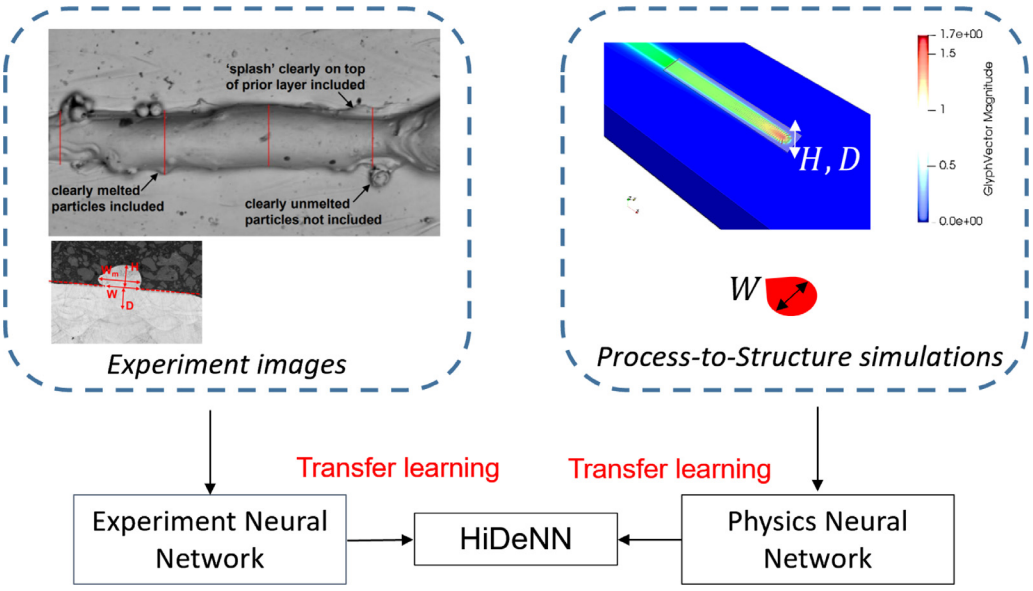

Fig. 22. The structure of data processing on experimental measurement and process-structure predictions.

Another approach to solve type 2 problems is using transfer learning to combine the experimental and simulation data. Transfer learning [77] refers to taking a pre-trained machine learning model and extending it to new circumstances combining experimental and computational data. These pre-trained models can be trained by experimental data and improved by combining simulation data or vice versa. It is an effective and efficient solution because experimental data come from a more realistic source but harder to get and simulation data can be easily generated but suffer from simplified assumptions in physics. By fusing the models with transfer learning, the HiDeNN can leverage small amount of experimental data to compensate for the lack of knowledge in physics coming from computational data.

One example of such a problem is the prediction of the melt pool dimensions in metal additive manufacturing [78,79]. The melt pool dimension can be predicted from computational models. However, these models fail to capture the uncertainties coming from process parameters, spatial distribution of powder particles and corresponding instantaneous change in the melt pool dimension. Fig. 22 presents a schematic of the problem. A single track sample (printed using an EOS M280 Laser Powder Bed Fusion (L-PBF) system) of commercially available Inconel 625 gas atomized powder is shown in the figure. For the melted track geometry, W is defined as the width of cross-sectioned track at the substrate plane, W_m is maximum width (the widest section of melted track), H is the height (highest point from the substrate plane), and D is the depth (deepest point from the substrate plane). For experimental measurement, the W, H, and D have variances dW, dH, and dD, respectively, caused by uncertainties. For the simulation, An effective medium CFD model is used to predict W, H, and D [78]. As discussed before, in simulation it is hard to consider all the uncertainties in manufacturing procedure. The experiment model and physics model will be combined together through transfer learning to make accurate predictions.

Fig. 23 shows the HiDeNN structure for this problem. The inputs are the coordinates and time in spatial and temporal space. For the parametric space, P refers to the laser power, v refers to the scan speed, and $\boldsymbol{C}$, and $\boldsymbol{\sigma}(\boldsymbol{\varepsilon})$ refer to the material properties. The pre-processing functions are used to extract features from experimental data such as images of the melt pool. The hierarchical DNNs have PHY-NN and EXP-NN combined by transfer learning. The operation layers are used to define the loss function. Several algorithms can be used to train the model, such as Kernel-Mean Matching (KMM) [80]:

$$\begin{aligned} &\min_{\beta} \quad \frac{1}{2}\beta^T K\beta - \kappa^T\beta \\ &s.t. \quad \beta_i \in [0, B] \ and \ \left|\sum_{i=1}^{n_S}\beta_i - n_S\right| \leq n_S\epsilon \end{aligned} \tag{29}$$

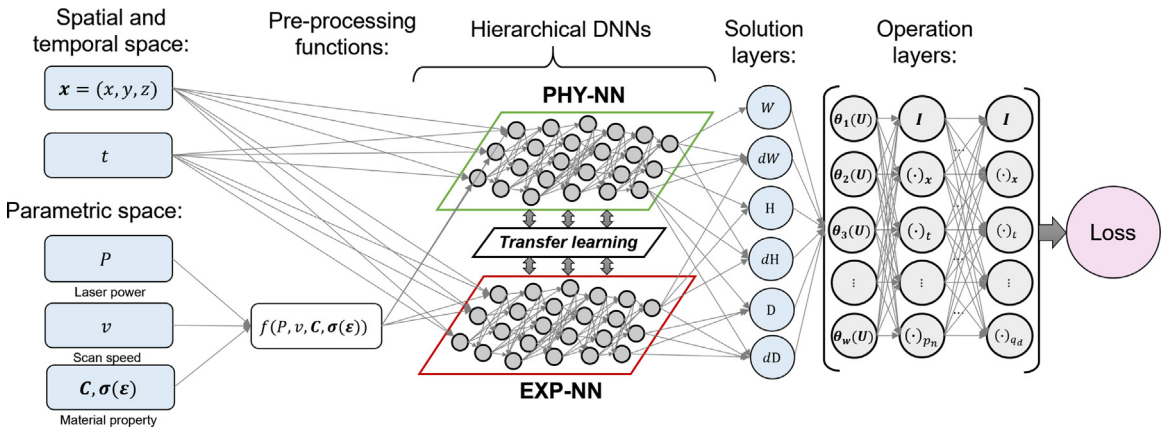

Fig. 23. Detail construction of the proposed HiDeNN transfer learning framework. The input layer takes in space, time, and parameters. The input layer is connected to the pre-processing function, Hierarchical DNNs, and finally the solution layer. By training the system, two neural networks are fused together.

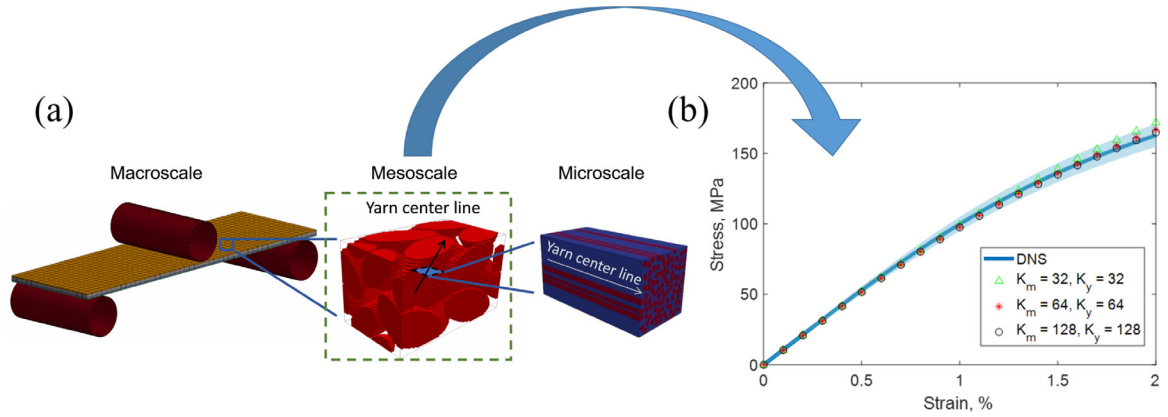

Fig. 24. (a) The illustration of the 3-scale multiscale model setup for 3-D braided composite laminate three point bending test. The macroscale is the 3-point bending geometry, and the mesoscale the 3D braided composite structure located in each macroscale material point. The mesoscale structure can be modeled with yarns and matrix (not shown for clarity) materials. Each yarn can be further modeled using UD structure at microscale. This multiscale setup uses no macroscale phenomenological model. Instead, the model can utilize mesoscale and microscale structures to predict the final laminate responses in the three point bending test. (b) Stress and strain responses for the DNS solution obtained using high-fidelity RVE and the ROM solutions with three different numbers of clusters. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The probability ratio between source domain and target domain is defined as β. The task of transfer learning is to find the ratio β, with the aim to eliminate the discrepancy in different models. In the above equations, K is the kernel matrix for the source domain data and the target domain data. Number of source samples is n_s and number of test samples is n_T. By solving the above equations, the EXP-NN and PHY-NN are fused together to a predictive HiDeNN model. The authors are further exploring the topic and more details will be included in future works.

4.3. Type 3: computationally expensive problem

For *Type 3* problems, systems with known physics are solved efficiently using SCA to predict material response with nonlinear behaviors. SCA can be used to provide fast and accurate data for implementing Reduced Order Model (ROM).

Consider the 3D braided composite laminate for 3-pt bending model (macroscale) shown in Fig. 24(a). At each integration point in the macroscale, a high fidelity Representative Volume Element (RVE) for the 3D braided mesoscale structure is applied to compute material responses, as shown in the middle of Fig. 24(a). The 3D braided

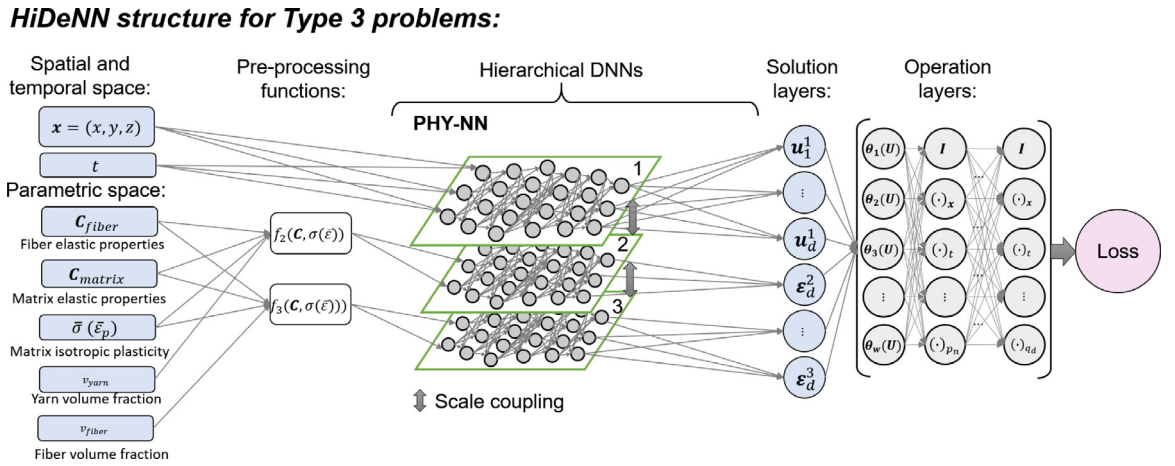

Fig. 25. Proposed HiDeNN structure for Type 3 problem. The spatial and temporal space describes the macroscopic loading history, and the parametric space defines the material parameters of the multiscale model. The pre-process functions will generate the reduced order model databases at meso- and micro-scale. The Hierarchical DNNs are established for three scales (1 for macroscale, 2 for mesoscale, and 3 for microscale) for concurrent solution of local displacements at macroscale and strain tensors at meso- and micro-scales.

RVE yarn responses are computed directly using a microscale unidirectional (UD) RVE, where the UD RVE fibers align with the yarn center line, as depicted in Fig. 24(a). In a 3-scale modeling, mesoscale, and microscale structures should be solved using SCA to ensure computational efficiency. For illustration purpose, only the mesoscale structure is studied in this work to reveal the significant saving in computational cost by SCA. The mesoscale RVE, shown in Fig. 24(a) with dashed green box, contains two phases: matrix and yarn. The matrix property is based on the data in [81], assuming isotropic elasto-plastic material response. The yarn is assumed to be transversely isotropic elastic, where the stiffness matrix is computed through computational homogenization of a UD RVE with fiber volume fraction of 51%. The 3D braided RVE, originally containing 212,500 voxel elements, is compressed into three sets of ROM. Each ROM contains different numbers of cluster for the matrix (K_m) and yarn (K_y) phases. Case 1 contains, $K_m = K_y = 32$; case 2 has $K_m = K_y = 64$, and case 3 contains $K_m = K_y = 128$. The time for offline calculation is 109.3 s for case 1, 379.2 s for case 2, and 1398.2 s for case 4. The ROM predictions of the uniaxial transverse tensile test responses are compared against direct RVE prediction, achieving a 5% difference as shown in Fig. 24(b) with a maximum speed-up of 446,896 for the $K_m = K_y = 32$ case.

For this problem, HiDeNN could be applied to convert the 3-scale multiscale model depicted in Fig. 24(a) into the HiDeNN formulation. The possible HiDeNN formulation of the multiscale problem is illustrated in Fig. 25. The associated loss function to Fig. 25 that couples macroscale (scale 1), mesoscale (scale 2), and microscale (scale 3), is shown in Eq. (30). Note that for practical implementation, three loss functions representing three distinct scales can be solved separately, and coupled together though scale coupling for information exchange, as shown in Fig. 25.

$$
\begin{aligned}
&\mathscr{L}(\boldsymbol{u}^h, \boldsymbol{\varepsilon}^{I,(2)}, \boldsymbol{\varepsilon}^{I,(3)}, \lambda^{(2)}, \lambda^{(3)}) = \mathscr{L}^1 + \lambda^{(2)}\mathscr{L}^{(2)} + +\lambda^{(3)}\mathscr{L}^{(3)} \\
&= \frac{1}{2}\int_\Omega \sigma(\boldsymbol{u}^h) : \varepsilon(\boldsymbol{u}^h) d\Omega - \Big(\int_\Omega \boldsymbol{u}^h \cdot \boldsymbol{f} d\Omega + \int_{\Gamma_t} \boldsymbol{u}^h \cdot \bar{\boldsymbol{t}} d\Gamma\Big) \Rightarrow \text{scale 1} \\
&+ \lambda^{(2)}\Bigg(\frac{1}{N_c^{(2)}}\sum_{I=1}^{N_c^{(2)}}\Big|\Delta\varepsilon^{I,(2)}(\boldsymbol{x}) + \sum_{J=1}^{N_c^{(2)}} \boldsymbol{D}^{IJ,(2)} : \big[\Delta\sigma^{J,(2)} - \boldsymbol{C}^0 : \Delta\varepsilon^{J,(2)}\big] - \Delta\bar{\boldsymbol{\varepsilon}}\Big|^2 + \Big|\sum_{I=1}^{N_c^{(2)}} c^{I,(2)}\Delta\varepsilon^{I,(2)} - \Delta\bar{\varepsilon}\Big|^2\Bigg) \\
&\Rightarrow \text{scale 2} \\
&+ \lambda^{(3)}\Bigg(\frac{1}{N_c^{(3)}}\sum_{I=1}^{N_c^{(3)}}\Big|\Delta\varepsilon^{I,(3)}(\boldsymbol{x}) + \sum_{J=1}^{N_c^{(3)}} \boldsymbol{D}^{IJ,(3)} : \big[\Delta\sigma^{J,(3)} - \boldsymbol{C}^0 : \Delta\varepsilon^{J,(3)}\big] - \Delta\bar{\boldsymbol{\varepsilon}}\Big|^2 + \Big|\sum_{I=1}^{N_c^{(3)}} c^{I,(3)}\Delta\varepsilon^{I,(3)} - \Delta\bar{\varepsilon}\Big|^2\Bigg) \\
&\Rightarrow \text{scale 3}
\end{aligned}
\tag{30}
$$

where $\boldsymbol{u}^h$ is the macroscale displacement, $\boldsymbol{\varepsilon}^{I,(2)}$ is the cluster-wise strain tensor in the mesoscale (scale 2), $\boldsymbol{\varepsilon}^{I,(3)}$ is the cluster-wise strain tensor in the microscale (scale 3), $\lambda^{(2)}$ and $\lambda^{(3)}$ are Lagrangian multipliers applied on the loss functions contributed by the meso- and micro-scales, $\mathscr{L}^{(1)}$ is the macroscale loss function (scale 1), $\mathscr{L}^{(2)}$ is the mesoscale loss function (scale 2), $\mathscr{L}^{(3)}$ is the microscale loss function (scale 3).

5. Future outlooks of HiDeNN

The article so far discusses the construction and application of HiDeNN framework and how we can apply the framework to three challenging problems (see Section 4) in computational science and engineering. This section discusses on necessary future extensions of HiDeNN.

To solve *type 2* or *mechanistically insufficient problems with limited data*, we might need to leverage the available experimental data from literature. However, the data coming from multiple sources are bound to suffer from lack of similarity in experimental conditions and thus cannot be applied directly as input. For such physical problems, we know some information about the system and data can come from known physics. In order to develop the DNN layers of HiDeNN (see Fig. 2), we can use transfer learning technique to merge experimental observations from different sources and existing data on the system. We have demonstrated a representative example on how transfer learning can be incorporated in HiDeNN. However, more research is needed to develop a method that can smoothly combine the EXP-NN and PHY-NN through transfer learning. Combining HiDeNN with symbolic regression methods, such as genetic programming [69], might provide explicit mathematical expressions, which sheds the important insights on the problems of interest.

In order to solve multi-scale problems, the HiDeNN can be explore further so that it can directly take experimental results (such as micrographs) into the framework as input.

Another possible application of HiDeNN would be in computational bio-mechanics where the governing physical equations of many problems (like bone growth) are not known. These problems fall under *type 1* or *purely data-driven problems*. By implementing HiDeNN, leveraging diagnostic results and/or images, the governing equations of bio-mechanics problems can be extracted as a function of the complex inputs such as age, sex, genetic information, or bone mineral density.

6. Summary

We present a novel framework, HiDeNN, as a narrow AI methodology to solve a variety of computational science and engineering problems. HiDeNN can assimilate many data-driven tools in an appropriate way, which provides a general approach to solve challenging computational problems from different fields. A detailed discussion on the construction of HiDeNN highlights the flexibility and generality of this framework. We illustrate an application of HiDeNN to perform multiscale analysis of composite materials with heterogeneous microstructure. Unique features of HiDeNN can offer automatic enrichment at the locations of strain concentration thus capturing effect of variable microstructure at part-scale. The results imply HiDeNN's ability to be applied to a class of computational mechanics problem where each material point at macroscale corresponds to non-uniform structure at microscale such as functional gradient alloy materials. We need further research to make HiDeNN automatic for arbitrary 3D problems. Furthermore, we apply HiDeNN to discover governing dimensionless parameters from experimental mechanistic data. The successful application of HiDeNN to such problems implies that similar framework can be applied to the field where the explicit physics is scarce, such as additive manufacturing. Finally, we propose future outlooks for solving three challenging problems using the same proposed AI framework. We demonstrate that HiDeNN has extra-ordinary features and can be a general solution method that takes advantage of ever increasing data from different experiments and theoretical model for fast prediction. A word of caution is that HiDeNN is still a proposed framework and further extensions and validations are needed before it can become a generally applicable AI framework to solve problems in diverse fields from mechanical engineering to biological science in the near future.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to acknowledge the support of National Science Foundation (NSF, USA) grants CMMI-1762035 and CMMI-1934367 and AFOSR, USA grant FA9550-18-1-0381. We thank Jennifer Bennett and her academic adviser Jian Cao for providing experimental data for Section 4.1.

References

[1] M.-W. Staff, Merriam-Webster's Collegiate Dictionary, Vol. 2, Merriam-Webster, 2004.
[2] E. Rich, K. Knight, Artificial Intelligence, second ed., Mc Graw-Hill, 1990.
[3] A.M. Turing, Computing machinery and intelligence, in: Parsing the Turing Test, Springer, 2009, pp. 23–65.
[4] A. Kaplan, M. Haenlein, Siri, siri, in my hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence, Bus. Horiz. 62 (1) (2019) 15–25.
[5] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., Mastering the game of go without human knowledge, Nature 550 (7676) (2017) 354–359.
[6] B. Goertzel, P. Wang, A foundational architecture for artificial general intelligence, Adv. Artif. Gen. Intell.: Concepts Archit. Algorithms 6 (2007) 36.
[7] N. Peek, C. Combi, R. Marin, R. Bellazzi, Thirty years of artificial intelligence in medicine (aime) conferences: a review of research themes, Artif. Intell. Med. 65 (1) (2015) 61–73.
[8] L. Deng, D. Yu, et al., Deep learning: methods and applications, Found. Trends Signal Process. 7 (3–4) (2014) 197–387.
[9] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT press, 2016.
[10] T. Kirchdoerfer, M. Ortiz, Data-driven computational mechanics, Comput. Methods Appl. Mech. Engrg. 304 (2016) 81–101.
[11] R. Ibañez, D. Borzacchiello, J.V. Aguado, E. Abisset-Chavanne, E. Cueto, P. Ladevèze, F. Chinesta, Data-driven non-linear elasticity: constitutive manifold construction and problem discretization, Comput. Mech. 60 (5) (2017) 813–826.
[12] W.K. Liu, G. Karniakis, S. Tang, J. Yvonnet, A Computational Mechanics Special Issue On: Data-Driven Modeling and Simulation—Theory, Methods, and Applications, Springer, 2019.
[13] A.L. Tarca, V.J. Carey, X.-w. Chen, R. Romero, S. Drăghici, Machine learning and its applications to biology, PLoS Comput. Biol. 3 (6) (2007).
[14] E. García-Cano, F.A. Cosío, L. Duong, C. Bellefleur, M. Roy-Beaudry, J. Joncas, S. Parent, H. Labelle, Prediction of spinal curve progression in adolescent idiopathic scoliosis using random forest regression, Comput. Biol. Med. 103 (2018) 34–43.
[15] S.S. Halabi, L.M. Prevedello, J. Kalpathy-Cramer, A.B. Mamonov, A. Bilbily, M. Cicero, I. Pan, L.A. Pereira, R.T. Sousa, N. Abdala, et al., The rsna pediatric bone age machine learning challenge, Radiology 290 (2) (2019) 498–503.
[16] N. Wagner, J.M. Rondinelli, Theory-guided machine learning in materials science, Front. Mater. 3 (2016) 28.
[17] J. Schmidt, M.R. Marques, S. Botti, M.A. Marques, Recent advances and applications of machine learning in solid-state materials science, npj Comput. Mater. 5 (1) (2019) 1–36.
[18] D. Ushizima, M. Noack, A. Hexemer, Data science and machine learning for polymer films and beyond, Bull. Amer. Phys. Soc. (2020).
[19] C. Drouillet, J. Karandikar, C. Nath, A.-C. Journeaux, M. El Mansori, T. Kurfess, Tool life predictions in milling using spindle power with the neural network technique, J. Manuf. Process. 22 (2016) 161–168.
[20] B. Zhang, S. Liu, Y.C. Shin, In-process monitoring of porosity during laser additive manufacturing process, Addit. Manuf. 28 (2019) 497–505.
[21] Z. Gan, H. Li, S.J. Wolff, J.L. Bennett, G. Hyatt, G.J. Wagner, J. Cao, W.K. Liu, Data-driven microstructure and microhardness design in additive manufacturing using a self-organizing map, Engineering 5 (4) (2019) 730–735.
[22] Y. Yu, T. Hur, J. Jung, I.G. Jang, Deep learning for determining a near-optimal topological design without any iteration, Struct. Multidiscip. Optim. 59 (3) (2019) 787–799.
[23] X. Lei, C. Liu, Z. Du, W. Zhang, X. Guo, Machine learning-driven real-time topology optimization under moving morphable component-based framework, J. Appl. Mech. 86 (1) (2019).
[24] I. Sosnovik, I. Oseledets, Neural networks for topology optimization, Russ. J. Numer. Anal. Math. Modelling 34 (4) (2019) 215–223.
[25] M. Bessa, R. Bostanabad, Z. Liu, A. Hu, D.W. Apley, C. Brinson, W. Chen, W.K. Liu, A framework for data-driven analysis of materials under uncertainty: countering the curse of dimensionality, Comput. Methods Appl. Mech. Engrg. 320 (2017) 633–667.
[26] Y.R. Tabar, U. Halici, A novel deep learning approach for classification of eeg motor imagery signals, J. Neural Eng, 14 (1) (2016) 016003.
[27] C. Fan, Y. Sun, Y. Zhao, M. Song, J. Wang, Deep learning-based feature engineering methods for improved building energy prediction, Appl. Energy 240 (2019) 35–45.
[28] E.A. del Rio-Chanona, J.L. Wagner, H. Ali, F. Fiorelli, D. Zhang, K. Hellgardt, Deep learning-based surrogate modeling and optimization for microalgal biofuel production and photobioreactor design, AIChE J. 65 (3) (2019) 915–923.
[29] S.H. Rudy, S.L. Brunton, J.L. Proctor, J.N. Kutz, Data-driven discovery of partial differential equations, Sci. Adv. 3 (4) (2017) e1602614.
[30] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. 113 (15) (2016) 3932–3937.
[31] K. Champion, B. Lusch, J.N. Kutz, S.L. Brunton, Data-driven discovery of coordinates and governing equations, Proc. Natl. Acad. Sci. 116 (45) (2019) 22445–22451.

[32] H. Li, O.L. Kafka, J. Gao, C. Yu, Y. Nie, L. Zhang, M. Tajdari, S. Tang, X. Guo, G. Li, S. Tang, G. Cheng, W.K. Liu, Clustering discretization methods for generation of material performance databases in machine learning and design optimization, Comput. Mech. 64 (2) (2019) 281–305.
[33] F. Fritzen, M. Fernández, F. Larsson, On-the-fly adaptivity for nonlinear twoscale simulations using artificial neural networks and reduced order modeling, Front. Mater. 6 (2019) 75.
[34] S. Xiao, R. Hu, Z. Li, S. Attarian, K.-M. Björk, A. Lendasse, A machine-learning-enhanced hierarchical multiscale method for bridging from molecular dynamics to continua, Neural Comput. Appl. (2019) 1–15.
[35] H. Chan, B. Narayanan, M.J. Cherukara, F.G. Sen, K. Sasikumar, S.K. Gray, M.K. Chan, S.K. Sankaranarayanan, Machine learning classical interatomic potentials for molecular dynamics from first-principles training data, J. Phys. Chem. C 123 (12) (2019) 6941–6957.
[36] W. Yan, S. Lin, O.L. Kafka, Y. Lian, C. Yu, Z. Liu, J. Yan, S. Wolff, H. Wu, E. Ndip-Agbor, et al., Data-driven multi-scale multi-physics models to derive process–structure–property relationships for additive manufacturing, Comput. Mech. 61 (5) (2018) 521–541.
[37] X. Li, Y. Zhang, H. Zhao, C. Burkhart, L.C. Brinson, W. Chen, A transfer learning approach for microstructure reconstruction and structure-property predictions, Sci. Rep. 8 (1) (2018) 1–13.
[38] J. Gao, M. Shakoor, H. Jinnai, H. Kadowaki, E. Seta, W.K. Liu, An inverse modeling approach for predicting filled rubber performance, Comput. Methods Appl. Mech. Engrg. 357 (2019) 112567.
[39] M.A. Bessa, P. Glowacki, M. Houlder, Bayesian machine learning in metamaterial design: Fragile becomes supercompressible, Adv. Mater. 31 (48) (2019) 1904845.
[40] Y. Liu, T. Zhao, W. Ju, S. Shi, Materials discovery and design using machine learning, J. Mater. 3 (3) (2017) 159–177.
[41] C.P. Gomes, B. Selman, J.M. Gregoire, Artificial intelligence for materials discovery, MRS Bull. 44 (7) (2019) 538–544.
[42] M. Mozaffar, R. Bostanabad, W. Chen, K. Ehmann, J. Cao, M. Bessa, Deep learning predicts path-dependent plasticity, Proc. Natl. Acad. Sci. 116 (52) (2019) 26414–26420.
[43] H. Yang, X. Guo, S. Tang, W.K. Liu, Derivation of heterogeneous material laws via data-driven principal component expansions, Comput. Mech. 64 (2) (2019) 365–379.
[44] S. Tang, G. Zhang, H. Yang, Y. Li, W.K. Liu, X. Guo, Map123: A data-driven approach to use 1d data for 3d nonlinear elastic materials modeling, Comput. Methods Appl. Mech. Engrg. 357 (2019) 112587.
[45] S. Tang, Y. Li, H. Qiu, H. Yang, S. Saha, S. Mojumder, W.K. Liu, X. Guo, Map123-ep: A mechanistic-based data-driven approach for numerical elastoplastic analysis, Comput. Methods Appl. Mech. Engrg. 364 (2020) 112955.
[46] S.L. Brunton, J.N. Kutz, Data-Driven Science and Engineering: MAchine Learning, Dynamical Systems, and Control, Cambridge University Press, 2019.
[47] Z. Liu, M. Bessa, W.K. Liu, Self-consistent clustering analysis: an efficient multi-scale scheme for inelastic heterogeneous materials, Comput. Methods Appl. Mech. Engrg. 306 (2016) 319–341.
[48] C. Yu, O.L. Kafka, W.K. Liu, Self-consistent clustering analysis for multiscale modeling at finite strains, Comput. Methods Appl. Mech. Engrg. 349 (2019) 339–359.
[49] M. Tajdari, A. Pawar, H. Li, F. Tajdari, A. Maqsood, E. Cleary, S. Saha, Y.J. Zhang, J.F. Sarwark, W.K. Liu, Image-based modeling for adolescent idiopathic scoliosis:mechanistic machine learning analysis and prediction, Comput. Methods Appl. Mech. Engrg. (2020) submitted for publication, in the same special issue.
[50] L. Zhang, L. Cheng, H. Li, J. Gao, C. Yu, R. Domel, Y. Yang, S. Tang, W.K. Liu, Hierarchical deep learning neural networks: Finite elements and beyond, Comput. Mech. (2020) (in press).
[51] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, 2017.
[52] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[53] ABAQUS/Standard User's Manual, Dassault Systèmes Simulia Corp, United States, 2016.
[54] T. Hughes, J. Cottrell, Y. Bazilevs, Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Engrg. 194 (39) (2005) 4135–4195.
[55] S. Yang, A. Tewari, A.M. Gokhale, Modeling of non-uniform spatial arrangement of fibers in a ceramic matrix composite, Acta Mater. 45 (7) (1997) 3059–3069.
[56] F. Feyel, J.-L. Chaboche, Fe2 multiscale approach for modelling the elastoviscoplastic behaviour of long fibre sic/ti composite materials, Comput. Methods Appl. Mech. Engrg. 183 (3–4) (2000) 309–330.
[57] Z. Liu, M. Fleming, W.K. Liu, Microstructural material database for self-consistent clustering analysis of elastoplastic strain softening materials, Comput. Methods Appl. Mech. Engrg. 330 (2018) 547–577.
[58] N. Altman, M. Krzywinski, The curse (s) of dimensionality, Nature Methods 15 (2018) 399–400.
[59] H.W. Lin, M. Tegmark, D. Rolnick, Why does deep and cheap learning work so well? J. Stat. Phys. 168 (6) (2017) 1223–1247.
[60] O. Reynolds, Xxix. An experimental investigation of the circumstances which determine whether the motion of water shall be direct or sinuous, and of the law of resistance in parallel channels, Phil. Trans. R. Soc. Lond. (174) (1883) 935–982.
[61] J. Nikuradse, et al., Laws of Flow in Rough Pipes, National Advisory Committee for Aeronautics Washington, 1950.
[62] E. Logan Jr, J. Jones, Flow in a pipe following an abrupt increase in surface roughness, 1963.
[63] M. Shockling, J. Allen, A. Smits, Roughness effects in turbulent pipe flow, J. Fluid Mech. 564 (2006) 267–285.
[64] K. Hornik, M. Stinchcombe, H. White, Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks, Neural Netw. 3 (5) (1990) 551–560.
[65] M.B. Priestley, Spectral analysis and time series: probability and mathematical statistics, 1981, 04; QA280, P7..
[66] Q.-M. Tan, Dimensional Analysis: with Case Studies in Mechanics, Springer Science & Business Media, 2011.
[67] S.J. Kline, Similitude and Approximation Theory, Springer Science & Business Media, 2012.

[68] G.I. Barenblatt, G.I. Barenblatt, B.G. Isaakovich, Scaling, Self-Similarity, and Intermediate Asymptotics: Dimensional Analysis and Intermediate Asymptotics, Vol. 14, Cambridge University Press, 1996.
[69] Z. Gan, O.L. Kafka, N. Parab, C. Zhao, O. Heinonen, T. Sun, W. Liu, Universal low-dimensional scaling laws in 3d printing of metals, 2020, arXiv preprint arXiv:2005.00117.
[70] J.L. Bennett, O.L. Kafka, H. Liao, S.J. Wolff, C. Yu, P. Cheng, G. Hyatt, K. Ehmann, J. Cao, Cooling rate effect on tensile strength of laser deposited inconel 718, Procedia Manuf. 26 (2018) 912–919.
[71] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[72] R.D. McGinty, Multiscale Representation of Polycrystalline Inelasticity (Ph.D. thesis), Georgia Tech, 2001, pp. 1–449.
[73] Z. Liu, O.L. Kafka, C. Yu, W.K. Liu, Data-driven self-consistent clustering analysis of heterogeneous materials with crystal plasticity, Computational Methods in Applied Sciences, Vol. 46, Springer International Publishing, 2018, pp. 221–242, http://dx.doi.org/10.1007/978-3-319-60885-3_11.
[74] M. Shenoy, J. Zhang, D. McDowell, Estimating fatigue sensitivity to polycrystalline ni-base superalloy microstructures using a computational approach, Fatigue Fract. Eng. Mater. Struct. 30 (10) (2007) 889–904.
[75] O. Kafka, C. Yu, M. Shakoor, Z. Liu, G. Wagner, W. Liu, Data-driven mechanistic modeling of influence of microstructure on high-cycle fatigue life of nickel titanium, JOM 70 (7) (2018) 1154–1158, http://dx.doi.org/10.1007/s11837-018-2868-2.
[76] M. Shakoor, O. Kafka, C. Yu, W.K. Liu, Data science for finite strain mechanical science of ductile materials, Comput. Mech. 64 (1) (2018) 33–45.
[77] L. Torrey, J. Shavlik, Transfer learning, in: Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques, IGI global, 2010, pp. 242–264.
[78] Z. Gan, Y. Lian, S.E. Lin, K.K. Jones, W.K. Liu, G.J. Wagner, Benchmark study of thermal behavior, surface topography, and dendritic microstructure in selective laser melting of inconel 625, Integr. Mater. Manuf. Innov. 8 (2) (2019) 178–193.
[79] Z. Gan, K. Jones, Y. Lu, W.K. Liu, The air force research laboratory, additive manufacturing (AM) modeling challenge series, 2020.
[80] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, A.J. Smola, Correcting sample selection bias by unlabeled data, in: Advances in Neural Information Processing Systems, 2007, pp. 601–608.
[81] J. Gao, M. Shakoor, G. Domel, M. Merzkirch, G. Zhou, D. Zeng, X. Su, W.K. Liu, Predictive multiscale modeling for unidirectional carbon fiber reinforced polymers, Compos. Sci. Technol. 186 (2020) 107922.