Front. Struct. Civ. Eng. 2020, 14(6): 1285–1298
https://doi.org/10.1007/s11709-020-0691-7

TRANSDISCIPLINARY INSIGHT

Multiscale computation on feedforward neural network and recurrent neural network

Bin LI, Xiaoying ZHUANG*

Department of Geotechnical Engineering, Tongji University, Shanghai 200092, China
**Corresponding author. E-mail: xiaoyingzhuang@tongji.edu.cn*

© Higher Education Press 2020

ABSTRACT Homogenization methods can be used to predict the effective macroscopic properties of materials that are heterogenous at micro- or fine-scale. Among existing methods for homogenization, computational homogenization is widely used in multiscale analyses of structures and materials. Conventional computational homogenization suffers from long computing times, which substantially limits its application in analyzing engineering problems. The neural networks can be used to construct fully decoupled approaches in nonlinear multiscale methods by mapping macroscopic loading and microscopic response. Computational homogenization methods for nonlinear material and implementation of offline multiscale computation are studied to generate data set. This article intends to model the multiscale constitution using feedforward neural network (FNN) and recurrent neural network (RNN), and appropriate set of loading paths are selected to effectively predict the materials behavior along unknown paths. Applications to two-dimensional multiscale analysis are tested and discussed in detail.

KEYWORDS multiscale method, constitutive model, feedforward neural network, recurrent neural network

1 Introduction

Homogenization theories for heterogeneous microstructures have been developed since the 1950s. In the homogenization method, in order to separate the macro and micro scales, there are two basic hypotheses: scale separation hypothesis and periodicity hypothesis. Hierarchical multiscale method based on the representative volume element (RVE) is a type of multiscale method that offers the numerical constitutive relationship at the macroscopic point depending on the micro- or the fine-scale model. The RVE is usually defined as a sample volume element of material. It should be sufficiently large in order to represent the statistical fluctuations or spatial variation in the microstructure and hence capture the effective mechanical properties, and yet it should be small enough to fulfill the assumption of separation in the length scale. Computational homogenization is one of the most widely used approaches, now used systematically for the assessment of structure-property relations.

Article history: Received Feb 19, 2020; Accepted Mar 18, 2020

Computational homogenization method offers the possibility of relatively high accuracy for complex material composition, however at high computational cost. Various efforts have been made in order to reduce the computational costs especially in nonlinear coupled simulations at two scales, such as the proper orthogonal decomposition (POD) [1,2], nonuniform transformation analysis (NFTA) [3,4], the self-consistent clustering analysis (SCA) [5,6]. However, the computational cost at present remains high and thus reduce its applicability for engineering applications in analysis of large-scale model. Recently, data-driven based methods were developed to compute the response of heterogeneous microstructures using an interpolation technique from a prior set of databases computed from offline nonlinear calculations on the RVEs [7]. However, these methods are usually problem-dependent, and the ability to extrapolation to other cases is not guaranteed, e.g., different material laws and loading paths [8].

Artificial neural networks (ANNs) have been used to approximate various constitutive model because of the capability to learn complex nonlinear relationships. In the early 90s, Ghaboussi et al. [9] proposed to model the

material behavior with neural network. Afterwards, the use of ANNs for direct representation of constitutive behavior [10–13] and macroscopic mechanical properties [14] has been studied by several researchers. The development echoes with the renaissance of neural network thanks to the raise of back propagation (BP) algorithm by Rumelhart et al. [15]. Recently, due to the recent growth in data availability, algorithm, and computing power that have brought a resurgence to the machine learning (ML), especially for ML based deep-learning neural network [16]. With deep learning framework, such as Tensorflow, PyTorch and so on, researchers can build and train deep-learning neural network expediently.

The neural networks can be used to construct fully decoupled approaches in nonlinear multiscale methods by mapping macroscopic loading and microscopic response [7]. Unger and Könke [17] used support vector machines and a multilayer perceptron for the decision of loading/unloading and the evaluation of stress tensor in a multiscale simulation, respectively. Bessa et al. [18] proposed a framework for data-driven multiscale analysis of materials, and Sobol sequence was used for the design of experiments. Le et al. [19] used neural network to determine the constitutive law of nonlinear elastic heterogeneous material. Lefik et al. [20] adopted neural networks in modeling of composites and hierarchical structures with a relatively small set of suitable numerical experiments. However, choosing the loading case for adequate training of multiscale constitutive model is a challenging task, which is not precisely known at this time. Different loading paths selected to train neural network for constitutive model or multiscale simulation will be discussed hereinafter. Most neural networks for multiscale constitutive model are based on feedforward neural network (FNN) and BP algorithm. Recurrent neural network (RNN) is a type of neural network where the output from previous step is fed as input to the current step, so that it may be used to model the nonlinear multiscale constitutive model considering material history dependency. However, only limited papers have adopted RNN for constitutive modeling. Zhu et al. [21] developed a RNN model for simulating and predicting soil behavior. Recently, Wang and Sun [22] built a multiscale multi-permeability poroplasticity model using long short-term memory (LSTM) neural network.

This article intends to model the multiscale constitution using FNN and RNN, and appropriate loading paths are selected so that the neural network multiscale constitutive model can be effectively generalized to unknown paths. The nonlinear multiscale computation is carried out using a multilevel finite element (FE^2) [23] method in ABAQUS. It is found that the trained FNN and RNN model both have good generalization ability on test set, and the stacked RNN model is more capable of mapping a given path, but also more sensitive to hyperparameters and initial parameters. The paper is composed as follows. In Section 2, multiscale modeling, including computational homogenization method for nonlinear material and implementation of offline multiscale computation. In Section 3, neural network, FNN and RNN for multiscale constitutive model are introduced. The detail of loading paths and training method will be discussed. Finally, applications to 2-dimensional nonlinear multiscale analysis are addressed in Section 4. Concluding remarks are given in Section 5.

2 Multiscale modeling

2.1 Computational homogenization method for nonlinear material

In the nonlinear computational homogenization scheme, at least two-scales models are needed, namely macro- and microscales or sometimes called coarse- and fine-scale models. Macroscale incremental strain is applied on RVE, while the homogenized incremental effective stress and homogenized tangent constitutive tensor are returned back to the macroscale model as shown in Fig. 1. In the two-scale homogenization theory, two different coordinates notations, namely $\boldsymbol{x}$ and $\boldsymbol{y}$, associated to the macroscale domain Ω and microscale models Θ, are adopted respectively. These two coordinates are related by $y = x/\zeta$ with $0 < \zeta \ll 1$. Consider a heterogeneous inelastic solid on a domain Ω^ζ with boundary $\partial\Omega^\zeta$. The strong incremental form of the boundary value problem on the macroscale domain is given as:

$$\Delta\sigma^\zeta_{ij,j} + \Delta b^\zeta_i = 0 \text{ on } \Omega^\zeta, \tag{1}$$

$$\Delta\sigma^\zeta_{ij} = \overline{L}^\zeta_{ijkl}\Delta\varepsilon^\zeta_{kl} \text{ on } \Omega^\zeta, \tag{2}$$

$$\Delta\varepsilon^\zeta_{ij} = \frac{1}{2}\left(\Delta u^\zeta_{i,j} + \Delta u^\zeta_{j,i}\right) \text{ on } \Omega^\zeta, \tag{3}$$

$$\Delta\sigma^\zeta_{ij}n^\zeta_j = \Delta\overline{t}_i{}^\zeta \text{ on } \partial\Omega^{t\zeta}, \tag{4}$$

$$\Delta u^\zeta_i = \Delta\overline{u}_i{}^\zeta \text{ on } \partial\Omega^{u\zeta}, \tag{5}$$

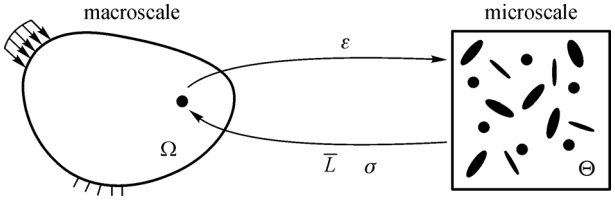

Fig. 1 The computational homogenization scheme.

where the superscript ζ denotes the scale factor between the macroscale and the microscale, $\partial\Omega^{t\zeta}$ and $\partial\Omega^{u\zeta}$ denote the traction and displacement boundaries, respectively,

$\Delta\sigma_{ij}^{\zeta}$, $\Delta\varepsilon_{ij}^{\zeta}$, Δu_i^{ζ}, $\overline{L}_{ijkl}^{\zeta}$ denote the components of incremental stress, strain, displacement and tangent constitutive tensor, respectively. Δb_i^{ζ} and t_i^{ζ} denote the body force vector and traction vectors, respectively.

With the use of the asymptotic expansion of the incremental displacement, the governing equations of the two-scale problem can be obtained. The derivation of the governing equations of a two-scale system will not be repeated here and readers are referred to Refs. [24–26].

1) The macroscale problem

$$\Delta\sigma_{ij,j} + \Delta b_i = 0 \text{ on } \Omega, \tag{6}$$

$$\Delta\sigma_{ij} = \overline{L}_{ijkl}\Delta\varepsilon_{kl} \text{ on } \Omega, \tag{7}$$

$$\Delta\sigma_{ij}n_j = \Delta\overline{t}_i \text{ on } \partial\Omega^t, \tag{8}$$

$$\Delta u_i = \Delta\overline{u}_i \text{ on } \partial\Omega^u. \tag{9}$$

2) The microscale problem

$$[L_{ijkl}(H_{k,y_l+I_{klmn}}^{mn})]_{,y_j} = 0 \text{ on } \Theta, \tag{10}$$

$$H_i^{mn}(y) = H_i^{mn}(y+l) \text{ on } \partial\Theta, \tag{11}$$

$$H_i^{mn}(y) = 0 \text{ on } \partial\Theta^{vert}, \tag{12}$$

where $H_k^{mn}(y)$ is termed as a first-order displacement influence function, Θ is the domain of the microscale model.

The bridging between the two-scale problems is given by

$$\overline{L}_{ijkl} = \frac{1}{|\Theta|}\int_{\Theta} L_{ijkl}\left(H_{k,y_l}^{mn} + I_{klmn}\right)\mathrm{d}\Theta, \tag{13}$$

$$\langle\sigma_{ij}\rangle_y^{t+1} = \langle\sigma_{ij}\rangle_y^t + \frac{1}{|\Theta|}\int_{\Theta}\Delta\sigma_{ij}\mathrm{d}\Theta, \tag{14}$$

where $\langle\sigma_{ij}\rangle_y^{t+1}$ is the average stress of the microscale material stress in increment $t+1$.

2.2 Implementation of offline multiscale computation

In the FE^2 method, the microscopic displacement can be written as the sum of a periodic field and a macroscopic field [23].

$$u(x,y) = v(x,y) + \varepsilon(x)\cdot b, \tag{15}$$

where $v(x,y)$ is a y-periodic displacement field, $\varepsilon(x)$ is the macroscopic strain. The general expression of periodic boundary condition is deduced as

$$\begin{aligned} u_i(y_j^+) - u_i(y_j^-) &= v_i(y_j^+) - v_i(y_j^-) + \varepsilon_{ij}(y_j^+ - y_j^-) \\ &= \varepsilon_{ij}(y_j^+ - y_j^-). \end{aligned} \tag{16}$$

The periodic boundary conditions can be applied efficiently by creating constrain equations, readers are referred as Ref. [24] for more details.

In offline multiscale calculation, the macroscopic strain, obtained a prior from a given loading path without computed results from macroscale model, is applied on a single RVE with constraint equations. The microscale model is simulated incrementally by restart analysis until the final step in the loading path is reached. To calculate the macroscopic stress and tangent tensor, one general and three perturbation steps for plane-stress problems are carried out on the RVE model. For restart analysis, the results from the previous step are used for the analysis in the current step and then overwritten in the next analysis step, which greatly reduces data storage in memory. The implementation comprises of the following steps as illustrated in Fig. 2. The flowchart demonstrates the following four steps:

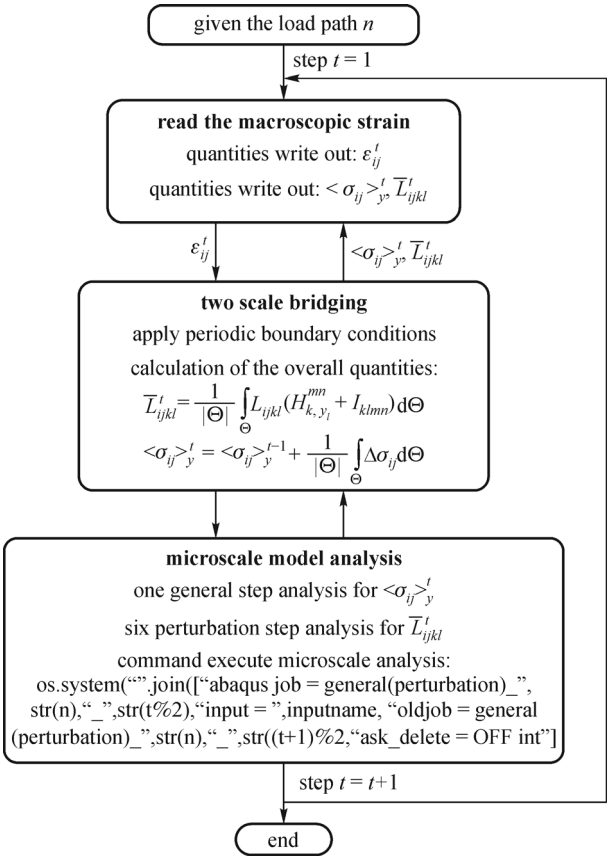

Fig. 2 Flow chart of offline multiscale computation using ABAQUS.

step 1: obtain the macroscopic total strain ε_{ij}^t (step t) from the loading path;

step 2: call the fine-scale model script to apply the periodic boundary conditions and macroscopic strain to the RVE model at global load step t, which will be the base

configuration of the step $t+1$. Then calculate the overall quantities $\langle\sigma_{ij}\rangle_y^t$;

step 3: carry out the perturbation steps for the global load step t, then compute the macroscopic consistent tangent tensor $\overline{L}_{ijkl}^t$;

step 4: write out the macroscopic stress $\langle\sigma_{ij}\rangle_y^t$ and tangent tensor $\overline{L}_{ijkl}^t$ into the corresponding output files and continue for the next load step.

3 Neural networks

Two types of neural networks are here used in this paper, namely FNN and RNN. In the following sections, a short overview highlighting the main features of the two approaches and why they are best suited for our problem is outlined. Most importantly, the generation of training data and neural network training details will be described and discussed.

3.1 Feed forward network (FNN)

A feed forward network (FNN) usually consists of an input layer, multiple hidden layers and an output layer, as shown in Fig. 3. FNN can learn and store a large number of input-output mode-mapping relationships without first revealing mathematical equations describing such mapping relationships. The learning mechanism of FNN is to continuously adjust the weights and the thresholds of network by using the steepest descent method. Through BP process that the minimization of the errors from the network between the output values and target values can be finally realized and the training is completed.

In Fig. 3, each circle represents a neuron, which is connected to other neurons in the neighboring layers with a weight and bias. Only forward connections are allowed in the FNN. The output a_i^l of neuron i in layer l is calculated as Ref. [17].

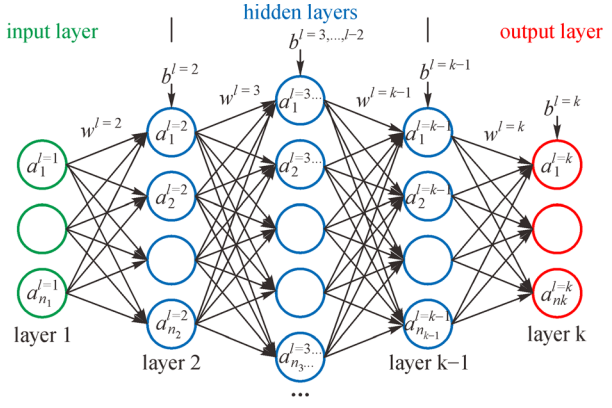

Fig. 3 General layout of FNN.

$$a_i^l = f(n_i^l), \tag{17}$$

$$n_i^l = \sum_{j=1}^{N_i^l} w_{ji}^l a_j^{l-1} + b_i^l, \tag{18}$$

where f is the activation function, w_{ji}^l corresponds to the weights of each connection and b^l is the bias.

As the nonlinear response of macroscale model is depend on the loading history and current loading increment, the macroscopic stress and strain of the previous step and the current incremental strain are used as the input parameters within this paper. The overall structure of the neural network can be described as

$$\Delta\sigma_{ij}^{t+1} = \mathcal{F}_{FNN}(\sigma_{ij}^t, \varepsilon_{ij}^t, \Delta\varepsilon_{ij}^{t+1}), \tag{19}$$

where $\mathcal{F}_{FNN}$ is FNN that maps inputs $(\sigma_{ij}^t, \varepsilon_{ij}^t, \Delta\varepsilon_{ij}^{t+1})$ and outputs $(\Delta\sigma_{ij}^{t+1})$.

The stiff matrix D_{ijkl}^{t+1} can be obtained by computing the partial derivatives of the output $\Delta\sigma_{ij}^{t+1}$ with respect to its input $\Delta\varepsilon_{ij}^{t+1}$, described by Eq. (20), which can be calculated using Jacobian function in Tensorflow.

$$D_{ijkl}^{t+1} = \frac{\partial\Delta\sigma_{ij}^{t+1}}{\partial\Delta\varepsilon_{ij}^{t+1}}. \tag{20}$$

3.2 Recurrent neural network (RNN)

FNN can be regarded as a complex function where each input is independent, and the output of the network depends only on the current input. This may work for many types of problems in engineering. However, for the loading path problem that we are investigating in this work, it is apparent that the input to the network is not only related to the input at the current moment, but also related to the output of the past period of time as a timing dependent problem. That means the "history" of previous events will influence the current event. To solve this issue, we will employ RNN. The RNN is a kind of neural network with short-term memory ability. In a RNN, neurons cannot only receive information from the neurons in the upper layer, but also accept their own information to form a network structure with loops.

RNN have been widely used in tasks such as speech recognition, language modeling and natural language processing. When the input sequence is relatively long, there will be the vanishing or exploding problem, known as the long-term dependence problem. A very good solution to the problem is to introduce gating mechanism to control the rate of information accumulation, including selectively adding new information and forgetting previous accumulated information, which is called Gated RNN. There are two kinds of Gated RNN: Long Short-Term

memory (LSTM) and Gated Recurrent Unit (GRU). Here, the latter is adopted in this paper because of its simple structure and promising performance. As proposed by Cho et al. [27], the architecture of a basic GRU cell is shown in Fig. 4.

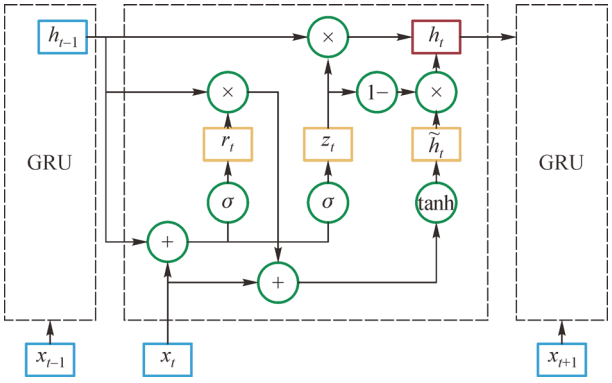

Fig. 4 GRU cell.

An update gate z_t is introduced to control the current state h_t that requires information to be accepted from the historical state h_{t-1} and candidate states $\tilde{h}_t$ given as

$$h_t = z_t h_{t-1} + (1-z_t)\tilde{h}_t, \tag{21}$$

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \tag{22}$$

$$\tilde{h}_t = \tanh(W_h x_t + U_h(r_t h_{t-1}) + b_h), \tag{23}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r), \tag{24}$$

where r_t is the reset gate used to control whether the computation of candidate state $\tilde{h}_t$ depends on the previous state h_{t-1}, σ is the sigmoid function $\sigma(x) = \dfrac{1}{1+\exp(-x)}$, 'tanh' is the hyperbolic tangent function $\tanh(x) = \dfrac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$, W_z, W_h, W_r, U_z, U_h, U_r are weight matrices, b_z, b_h, b_r are bias vectors. To map the current state h_t to the final output y_t, fully connection layers equivalent to FNN are introduced to complete this process.

The motivation to use RNN for multiscale nonlinear computation is that the loading history information can be retained. Different from training discrete data points for each path in FNN, a loading path is used as a time-dependent data chain in RNN, and the previous data will affect the later outputs. The inputs and outputs for RNN are only strain increments and stress increments of different time, respectively. Deep RNN that stacks multiple RNN can be used to enhance the capability of RNN. Within this paper, a stacked GRU neural network (SRNN) connected with fully connected layers at each time step is adopted, which is illustrated in Fig. 5.

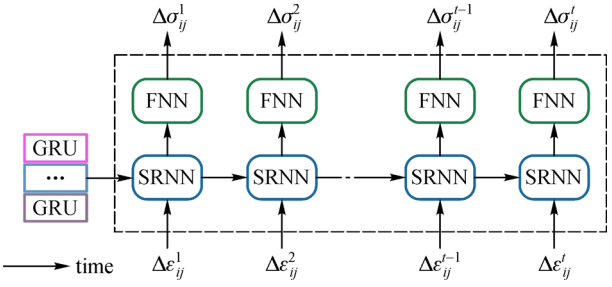

Fig. 5 The stacked GRU neural networks unrolled through time.

3.3 Generation of training data

For ML, the most important part is the acquisition of data. The question of what constitutes sufficient data sets for adequate training of neural network material model is not precisely known at this time [28], and the well-trained neural network model does not guarantee a good generalization to other paths. The choice of loading cases to be included in the training data set is often based on empirical knowledge [22].

A most popular loading method is the so-called spherical coordinate system (R,θ,α) method with a constant magnitude in each components of strain vector [10,14,17]. In previous works, loops of hysteresis [29] and proportional loading paths [22] have been used to model the nonlinear constitutive behavior. However, the capability of a method for adapting to different types of loading paths are not fully stated or investigated in the above works. To breakthrough this limit, the macroscopic strain tensor increment is set to be inhomogeneous by using a cosine function, and thus the total macroscopic strain tensor component is given as

$$\varepsilon_i^t = \varepsilon_i^{t-1} + \frac{k}{n} R\cos\left(\theta_i^t\right),\ k = 1,...,n, \tag{25}$$

where $\varepsilon_i^0 = 0$, R and θ are the loading radius and angle, respectively. The applied macroscopic strain tensor components are increased with independent constant increments until $k = n$, and the loading angles will change randomly within the set $\{0°,1°,...,180°\}$. Therefore, the training data can reflect the characteristics of different loading direction. The procedure for applying the strain component is illustrated in Fig. 6. The reason for using the cosine function is to generate the data points with relatively large strain increments denser while changing the loading direction illustrated in Fig. 7.

3.4 Training the neural network

In neural network, hyperparameters refer to prior parameters that need to be tuned to optimize it, including the network structure (number of layers, number of neurons in each layer, activation function, etc.), optimized parameters

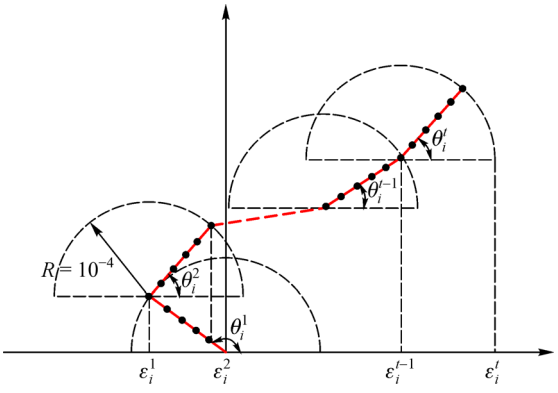

Fig. 6 The procedure for applying the strain component.

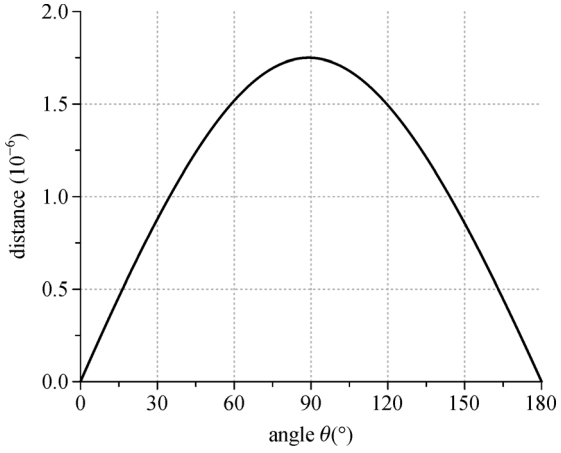

Fig. 7 The distance between two adjacent points the strain component in the cosine function cosθ.

(learning rate, mini-batch size, etc.) and regularization coefficient. Grid search and manual search are the most widely used strategies for hyperparameter optimization. Within this paper, grid search with orthogonal experimental design (OED) is used to obtain optimal result in a short time. The OED is a multifactor experiment design method based on the orthogonal array, it selects representative points from full factorial experiment in a way that the points are distributed uniformly within the test range and thus can represent the overall situation [30]. Considering a problem having 3 number of 3 level factors, 9 experimental points can be selected from 27 test points of full experiment with orthogonal array $L_9(3^4)$, as Fig. 8 shows. The Adam optimization algorithm, mini-batch gradient descent and dropout are adopted based on the Tensorflow within this paper. In addition, the input data X and output data Y for both FNN and RNN are scaled to be within -1 to $+1$ range, and the activation functions for fully connected layers are all chosen to be tanh. The error function J and accuracy P for predicted values $\overline{Y}$ and true values Y of M samples are computed as

$$J = \left(\frac{1}{M}\sum_{i=1}^{M}(|\overline{Y}_i - Y_i|)\right), \tag{26}$$

$$P = 1 - \frac{\sum_{i=1}^{M}\left|\left(\frac{\overline{Y}_i - Y_i}{Y_i}\right)Y_i\right|}{\sum_{i=1}^{M}|Y_i|} = 1 - \frac{\sum_{i=1}^{M}|\overline{Y}_i - Y_i|}{\sum_{i=1}^{M}|Y_i|}. \tag{27}$$

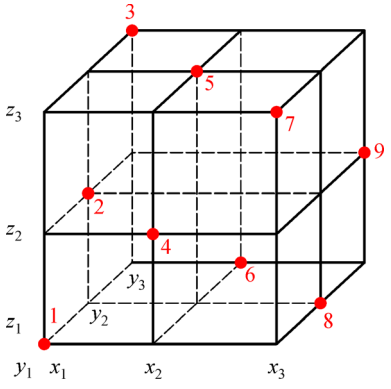

Fig. 8 9 experimental points selected from 27 test points with orthogonal array $L_9(3^4)$.

4 Example: two-dimensional inelastic multiscale analysis for plane stress problem

In this section, the FNN and deep RNN will be applied to model two-dimensional inelastic plane stress problem. The RVE model shown in Fig. 9 is studied in this section. Materials B and M, respectively marked with color red and blue, follow the isotropic hardening law (Fig. 10), and their material parameters are shown in Table 1.

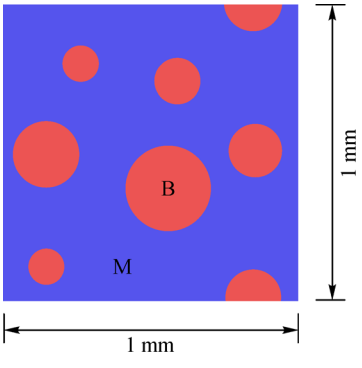

Fig. 9 Two-dimensional RVE model.

The parameters n of Eq. (25) is set to be 2, 5, 10, 20, and each parameter is aligned with 50 loading paths, so that a

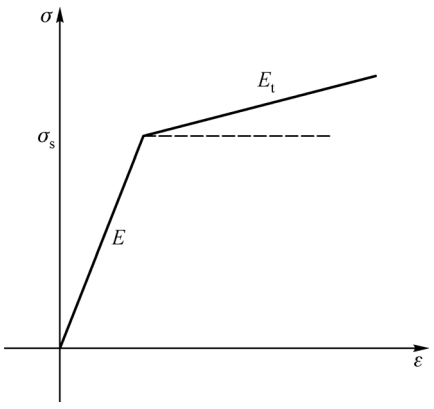

Fig. 10 The isotropic hardening law model.

Table 1 Material parameters

material	E (MPa)	E_t (MPa)	ν	σ_s (MPa)
M	1×10^5	2×10^4	0.3	200
B	1×10^5	1×10^4	0.3	100

total of 200 loading paths are selected. The parameters R is set to be 10^{-4}, and each path comprises 100 sub steps, so the macroscopic strain component falls in the range of $[-0.01,0.01]$. However, the macroscopic strain component $|\varepsilon_i|_{\max}$ in the training set is about to be 0.005 given that the loading angles are chosen randomly, and the extreme values can only be obtained when the loading angles are selected to be 0° or 180° every time. Take one loading path as an example ($n=5$), the macroscopic strain components of different time steps in cartesian are shown in Fig. 11. The data set is divided into training set, validation set and test set, and the partition is 6:2:2.

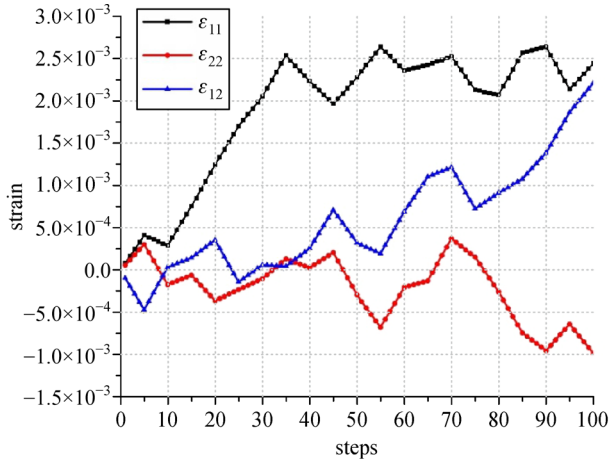

Fig. 11 The macroscopic strain components of different time steps.

4.1 FNN model

As mentioned in previous section, the input and output are $(\sigma_{ij}^N,\varepsilon_{ij}^N,\Delta\varepsilon_{ij}^{N+1})$ and $(\Delta\sigma_{ij}^{N+1})$, respectively. The validation set is used for hyperparameter optimization with orthogonal experimental design (OED). The hyperparameters to be optimized for FNN are the number of neural network layers, learning rate, mini-batch size and dropout rate. The orthonormal testing calculation and result based on orthogonal array $L_9(3^4)$ are shown in Table 2. The optical hyperparameters are not selected simply from the sample points, but obtained by comparing the average of the results for each hyperparameter.

Table 2 The orthonormal testing calculation

experiment number	layers number	learning rate	mini-batch size	dropout rate	accuracy
1	4	1e−2	16	0.75	0.767
2	4	1e−3	32	0.85	0.838
3	4	1e−4	64	0.95	0.856
4	5	1e−2	32	0.95	0.866
5	5	1e−3	64	0.75	0.866
6	5	1e−4	16	0.85	0.870
7	6	1e−2	64	0.85	0.867
8	6	1e−3	16	0.95	0.933
9	6	1e−4	32	0.75	0.895
parameter	6	1e−3	32	0.95	

Note: The number of neurons of layer i is set to be n_input$\cdot 2^{n-i}$, where n_input is the neuron number of input layer, n is total number of layers.

The accuracy is calculated using Eq. (27), and the error and accuracy curves of training set are shown in Fig. 12. The accuracies of training set and test set are 93.0% and 91.7%, respectively. One test loading path and corresponding stress-strain curves in the test set are shown in Fig. 13. It can be seen from the figure that the curves calculated by FNN in the test set fit well with the numerical result using FEM, which shows the generalization ability of FNN model is very good.

4.2 Deep RNN model

Different from FNN, the input and output for RNN are only strain increment and stress increment, respectively. Deep RNN that stacks multiple RNN is used to enhance the capability of RNN. The hyperparameters to be optimized for RNN are the number of GRU cell and fully connected layer, learning rate, and dropout rate. Similar to last section, orthogonal array $L_9(3^4)$ is used for hyperparameter optimization shown in Table 3.

The error and accuracy curves of training set are shown in Fig. 14. The accuracies of training set and test set are

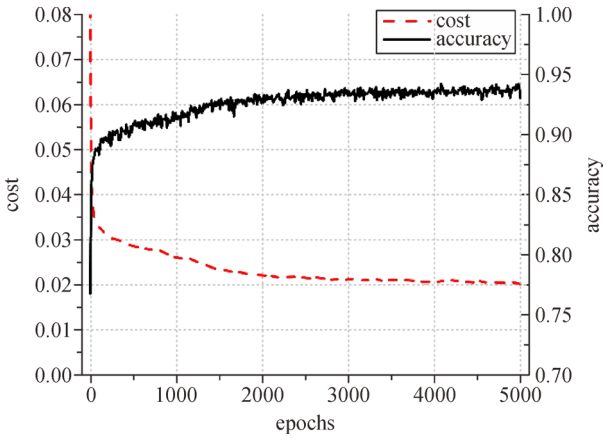

Fig. 12 The error and accuracy curves of training set.

96.3% and 93.6%, respectively. As the network is more complex, the RNN model is more capable of mapping a given path, but also more sensitive to hyperparameters and initial parameters. One loading path and corresponding stress-strain curves in the test set are shown in Fig. 15.

4.3 The ability to generalize to other paths

To illustrate the ability of the neural network model to generalize to other paths, three typical loading paths are selected, where the increment and strain are within the corresponding value range of the neural network model. In the second loading path, the strain increases monotonically, and the strain increments at different time are Sobol sequence. The three loading paths and stress-strain curves of numerical computation, FEM, FNN, and RNN are shown in Figs. 16–18. The accuracies of three paths predicted by FNN are 90.6%, 96.8%, and 92.4%, respectively, while the accuracy for RNN are 88.3%, 96.7%, and 88.2%, respectively.

It can be seen from the Figs. 16–18 that the ability to

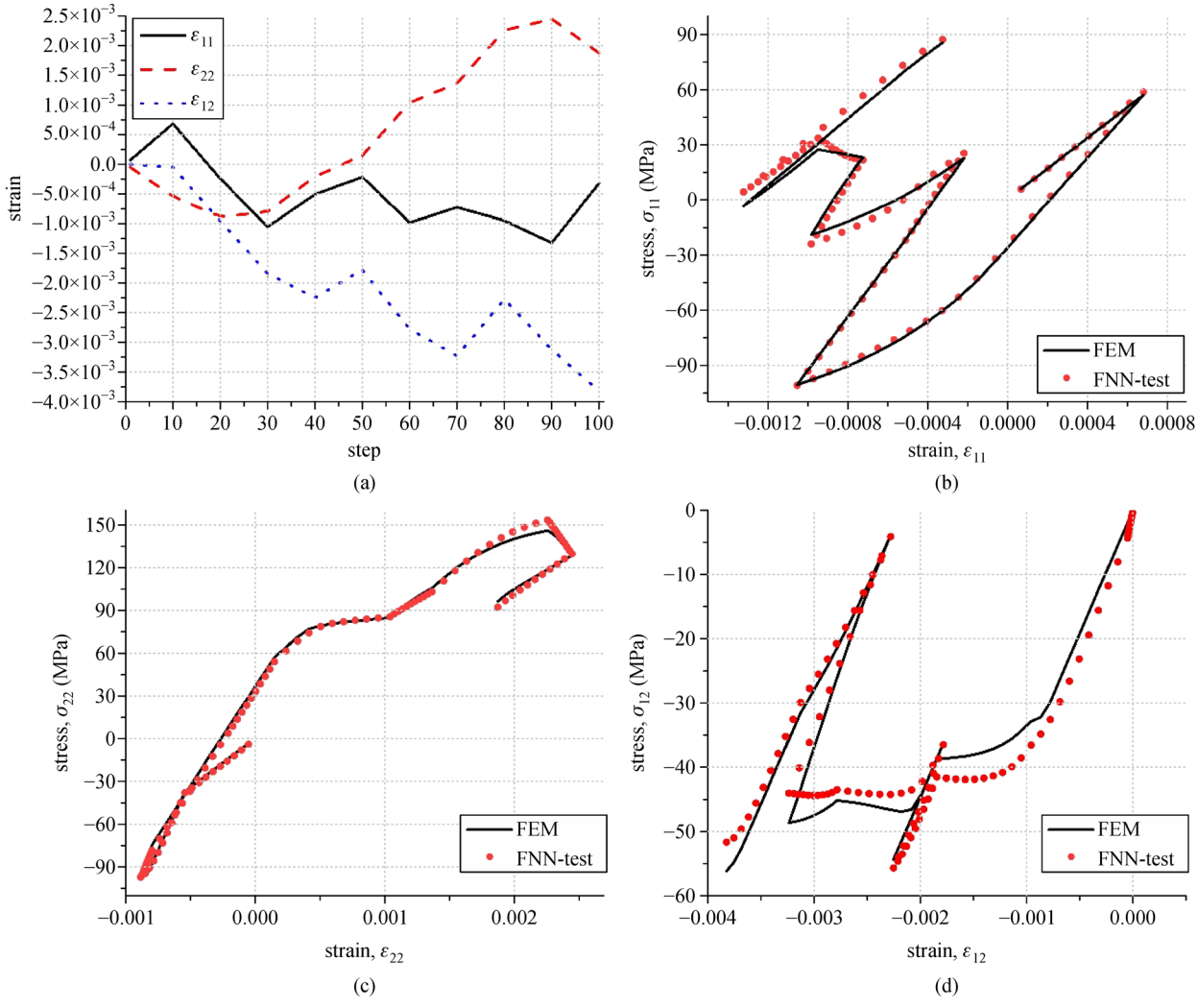

Fig. 13 One loading path and corresponding stress-strain curves in the test set. (a) The loading paths; (b) the stress-strain curves (σ_{11}-ε_{11}); (c) the stress-strain curves (σ_{22}-ε_{22}); (d) the stress-strain curves (σ_{12}-ε_{12}).

Table 3 The orthonormal testing calculation

experiment number	GRU layers	FNN layers	learning rate	dropout rate	accuracy
1	1	1	1e−3	0.75	0.817
2	1	2	1e−4	0.85	0.800
3	1	3	1e−5	0.95	0.681
4	2	1	1e−4	0.95	0.840
5	2	2	1e−5	0.75	0.660
6	2	3	1e−3	0.85	0.938
7	3	1	1e−5	0.85	0.521
8	3	2	1e−3	0.95	0.949
9	3	3	1e−4	0.75	0.828
parameter	2	3	1e−3	0.95	

Note: The number of neurons of GRU layer and fully connected layer i are set to be n_input·16 and n_input·2^{n+1-i}, respectively, where n_input is the number of neurons of input layer, n is the number of fully connected layers.

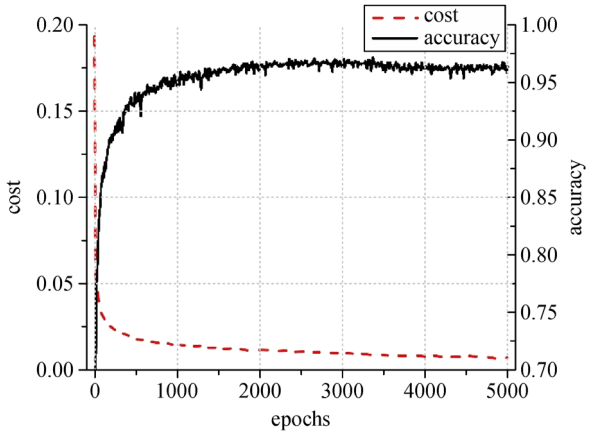

Fig. 14 The error and accuracy curves of training set.

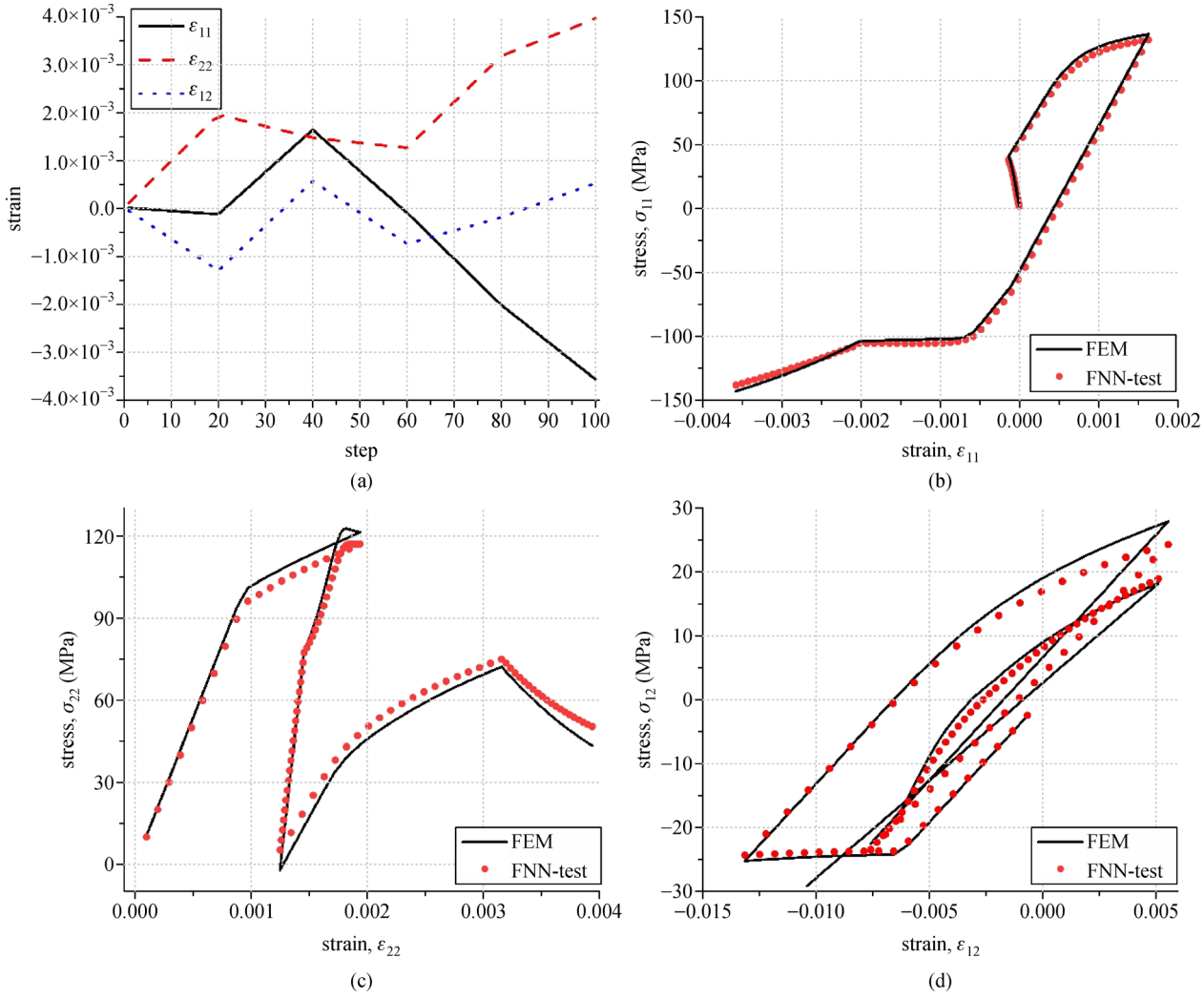

Fig. 15 One loading path and corresponding stress-strain curves in the test set. (a) The loading paths; (b) the stress-strain curves (σ_{11}-ε_{11}); (c) the stress-strain curves (σ_{22}-ε_{22}); (d) the stress-strain curves (σ_{12}-ε_{12}).

generalize to other simple paths is guaranteed for both FNN and RNN, especially in the case of monotonic loading. Due to the small shear modulus, σ_{12} is relatively small, and its prediction is not as accurate as other stress components. The data set for RNN is the loading paths, which is much smaller than the data set of FNN, but the prediction capability of the two model are comparable. Although some other paths have been given to prove the generalization ability of FNN and RNN models, the proof of extrapolation to other paths is insufficient, which requires further research.

In the examples shown in Figs. 16–18, the calculation time of FEM, FNN model and RNN model are recorded in Table 4, and the time of FEM, FNN prediction and RNN prediction is the average of the three samples. It takes more time to train RNN model than FNN model, but the prediction time difference between the two model is very small. With the neural network model, the computational expense can be greatly reduced. Once the neural network model is trained, it can predict the behavior of the RVE model in just a few seconds, while FEM can take about half an hour. However, data acquisition consumes a lot of time as the finite element simulation of different paths is not parallel, which needs to be improved in the future.

4.4 The ability to generalize to other RVE model

In the calculation of different RVE models, it is found that the macroscopic stress and tangent tensor of the microstructure are mainly related to the proportion of different components when the material constitution remain unchanged. Another RVE model (RVE2) shown in Fig. 19 has the same volume fraction and material constitution with the RVE model (RVE1) in Fig. 9. The macroscopic stress-strain curves of the two RVE model at a given loading path shown in Fig. 20 are almost coincide, so the trained FNN and RNN model based on RVE1 can be used to predicted the behavior of RVE model with the same

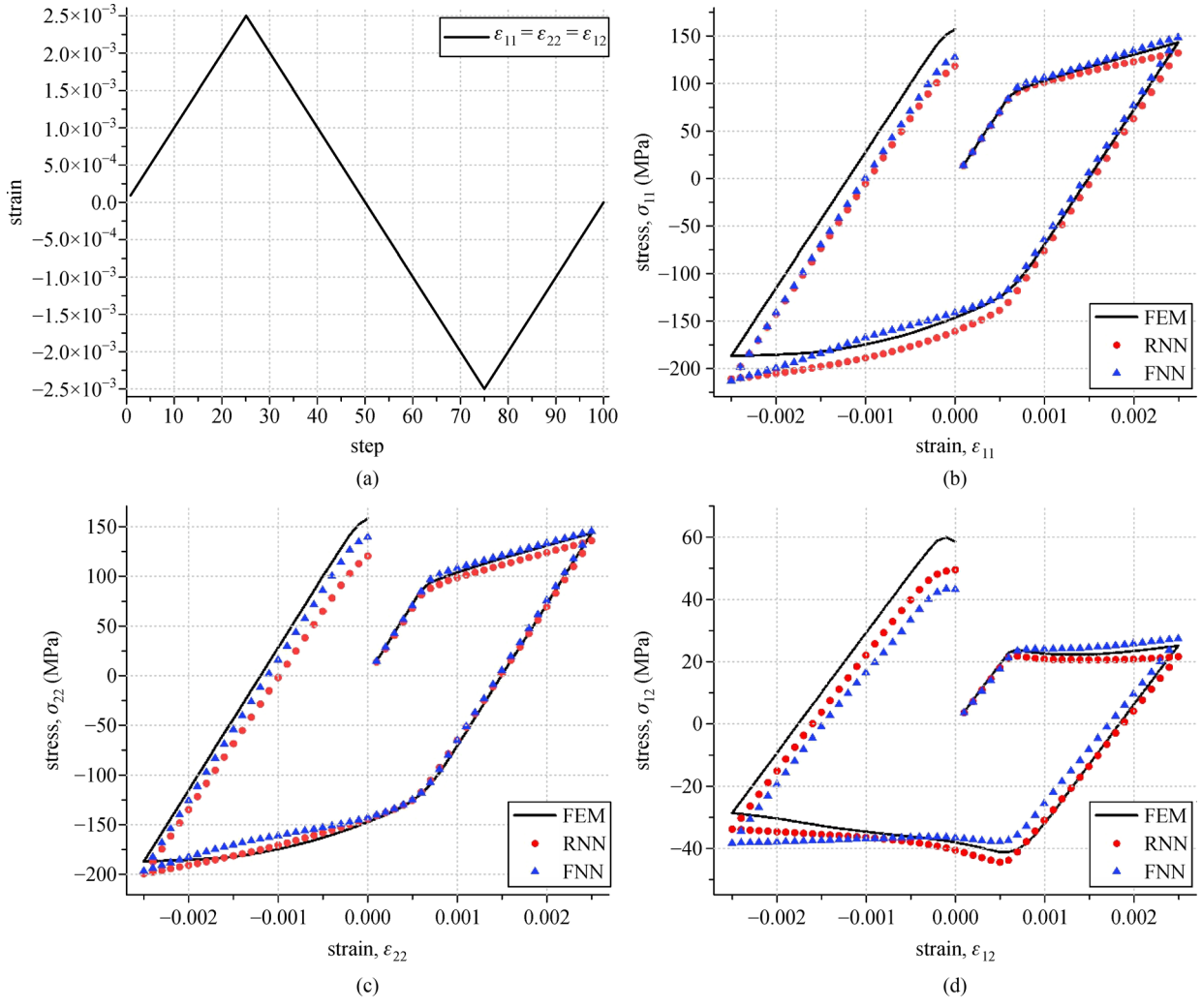

Fig. 16 First loading path and corresponding stress-strain curves. (a) The loading paths; (b) the stress-strain curves (σ_{11}-ε_{11}); (c) the stress-strain curves (σ_{22}-ε_{22}); (d) the stress-strain curves (σ_{12}-ε_{12}).

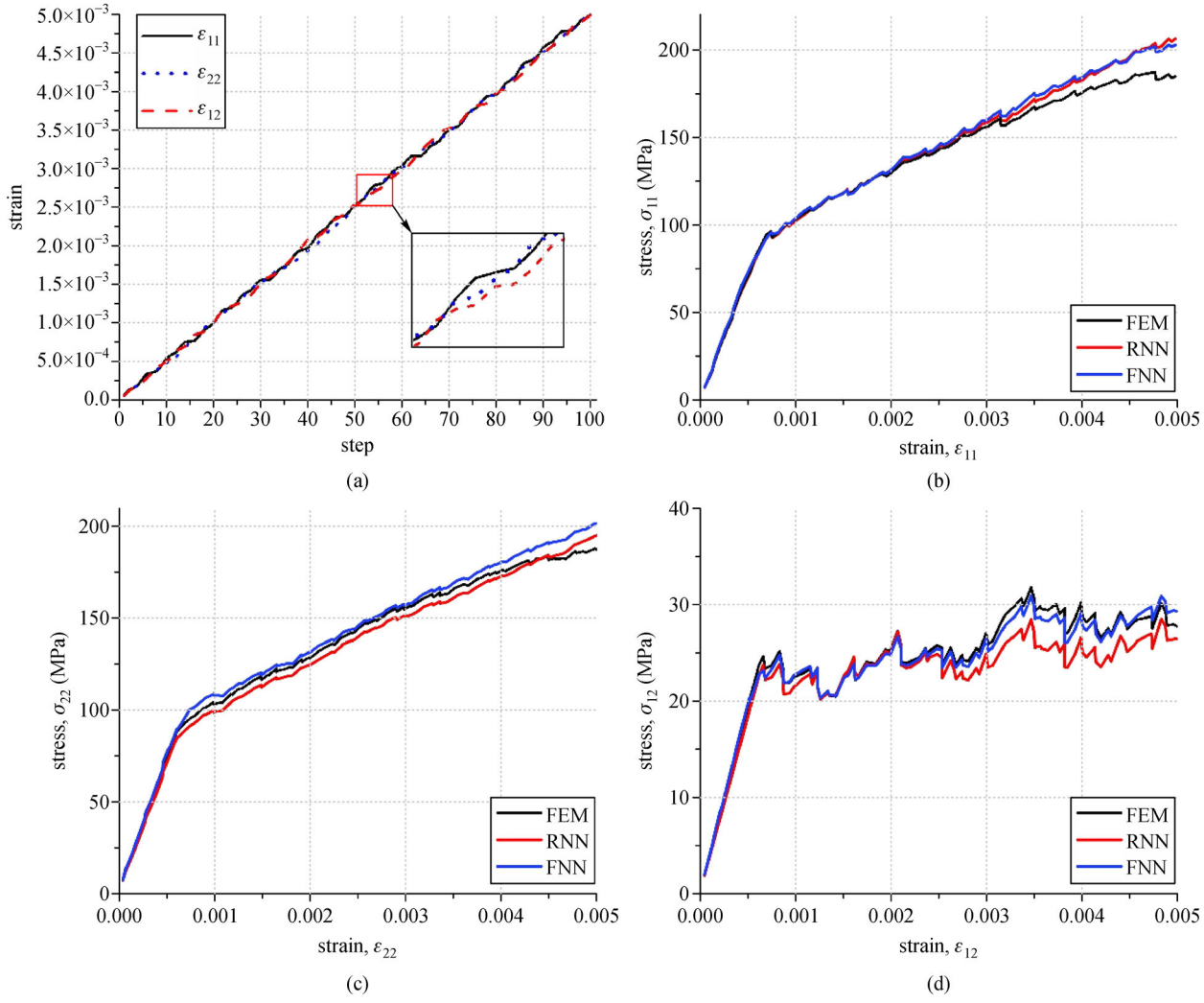

Fig. 17 Second loading path and corresponding stress-strain curves. (a) The loading paths; (b) the stress-strain curves (σ_{11}-ε_{11}); (c) the stress-strain curves (σ_{22}-ε_{22}); (d) the stress-strain curves (σ_{12}-ε_{12}).

components. The loading path is obtained from Eq. (25), and the parameter n is 10. It can be seen from the Fig. 20 that both FNN and RNN models can fit well with the result from FEM. However, the input parameters of FNN model and RNN model do not include the proportion of different components and their spatial distribution, so the ability to generalize to other RVE model is limited, which can be studied in the future.

In the example shown in Fig. 20, the calculation time of FEM, FNN model and RNN model is recorded in Table 5. Similar to the result in Table 4, the computational expense can be greatly reduced with the neural network model.

5 Conclusions

Within this paper, neural network based multiscale material model using FNN and RNN are proposed, and proper loading paths are selected to effectively generalized to unknown paths to some extent. The data set is obtained via implementation of offline multiscale computation based on FE^2 method. The trained FNN and RNN model both have good generalization ability on test set. The stacked RNN model is more capable of mapping a given path considering its more complex network structure, but also more sensitive to hyperparameters and initial parameters. The main ideas of building a material model based on FNN and RNN considering material history dependency are as follows: 1) including as many paths as possible; 2) taking historical information as input variables. Physical variables such as internal variables [31–33] can also be introduced to build one to one or many to one mapping for neural network material model, thus enhance the robustness and generalization ability of the neural network based multiscale model. Recently, an energy approach [34,35] and a collocation method [36,37] have

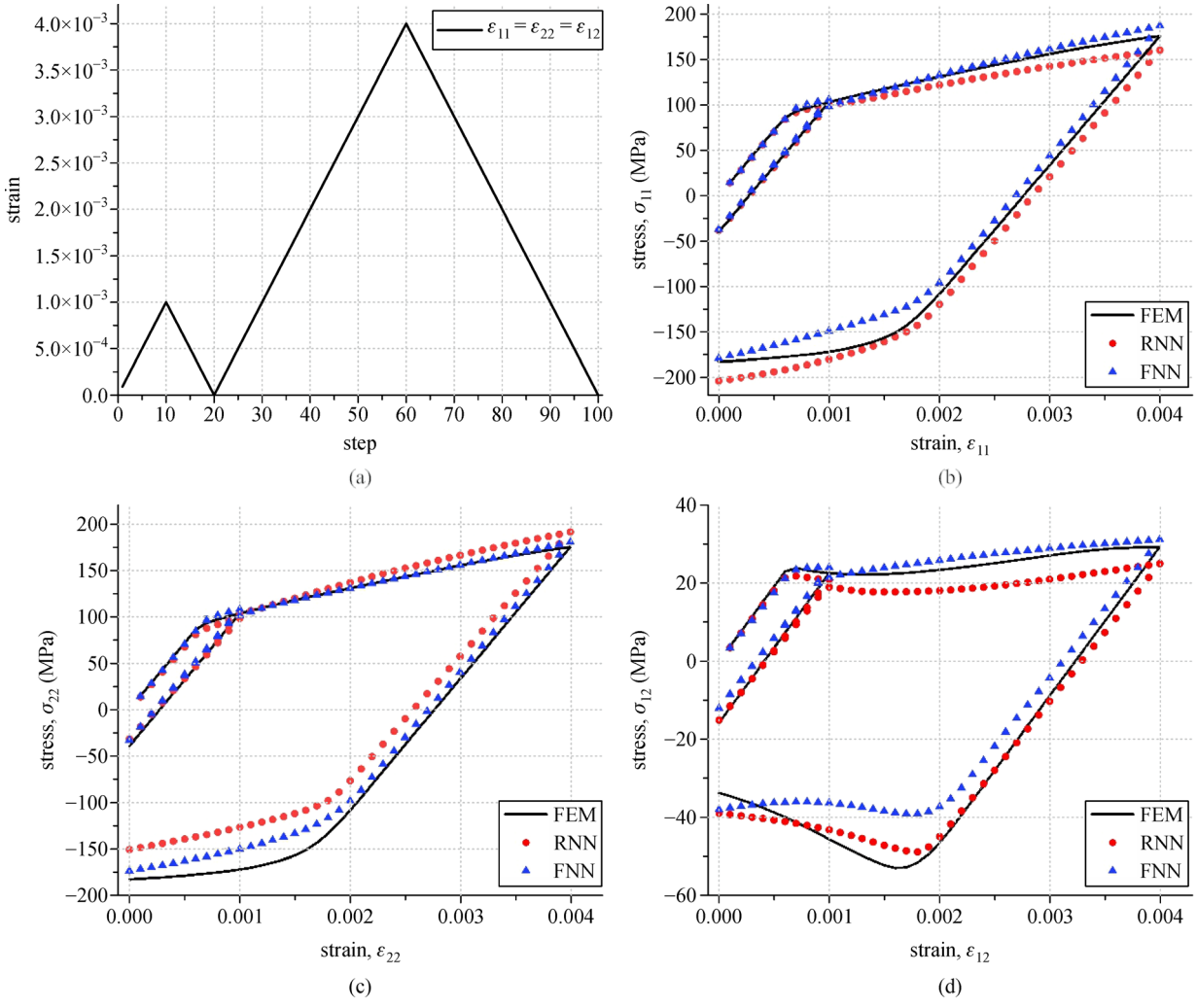

Fig. 18 Third loading path and corresponding stress-strain curves. (a) The loading paths; (b) the stress-strain curves (σ_{11}-ε_{11}); (c) the stress-strain curves (σ_{22}-ε_{22}); (d) the stress-strain curves (σ_{12}-ε_{12}).

Table 4 Calculation time of FEM, FNN model and RNN model

FEM	data acquisition	FNN training	FNN prediction	RNN training	RNN prediction
2400 s	48000 s (2400∗200 s)	1089 s	3 s	1846.8 s	4 s

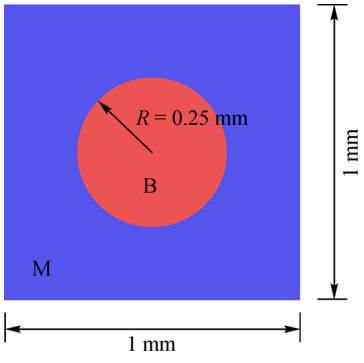

Fig. 19 Two-dimensional RVE model (RVE2).

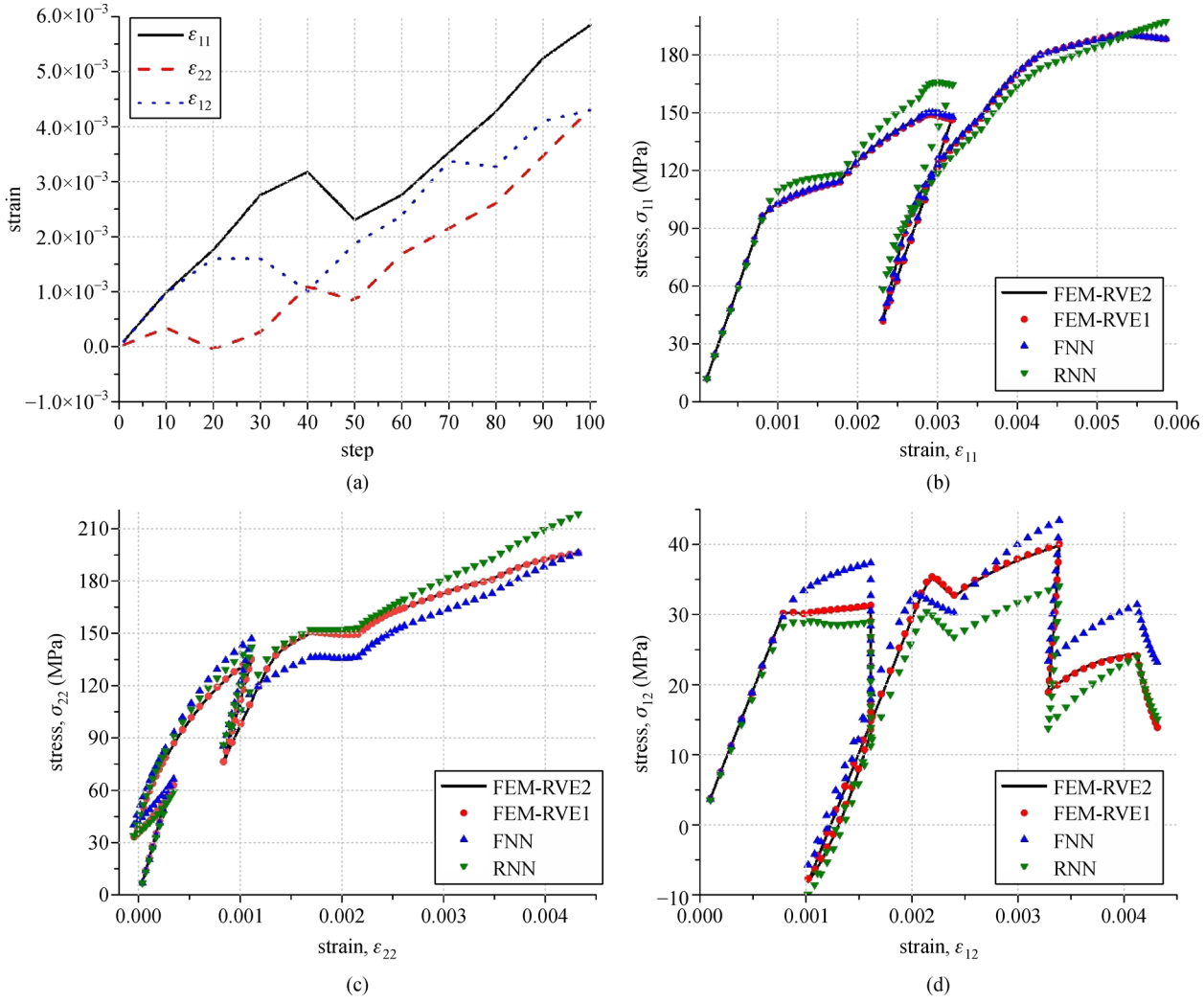

Fig. 20 A loading path and corresponding stress-strain curves. (a) The loading paths; (b) the stress-strain curves (σ_{11}-ε_{11}); (c) the stress-strain curves (σ_{22}-ε_{22}); (d) the stress-strain curves (σ_{12}-ε_{12}).

Table 5 Calculation time of FEM, FNN model and RNN model

FEM	data acquisition	FNN training	FNN prediction	RNN training	RNN prediction
2160 s	43200 s (2160* 200 s)	1089 s	4.8 s	1846.8 s	4.2 s

been proposed to solve partial differential equations, providing a deeper perspective to combine computational mechanics and ML, which needs further study.

Acknowledgements The authors acknowledge the support from the National Natural Science Foundation of China (Grant No. 11772234).

References

1. Fritzen F, Marfia S, Sepe V. Reduced order modeling in nonlinear homogenization: A comparativestudy. Computers & Structures, 2015, 157: 114–131
2. Kerfriden P, Goury O, Rabczuk T, Bordas S P A. A partitioned model order reduction approach to rationalise computational expenses in nonlinear fracture mechanics. Computer Methods in Applied Mechanics and Engineering, 2013, 256: 169–188
3. Michel J C, Suquet P. Nonuniform transformation field analysis. International Journal of Solids and Structures, 2003, 40(25): 6937–6955
4. Roussette S, Michel J C, Suquet P. Nonuniform transformation field analysis of elastic-viscoplastic composites. Composites Science and Technology, 2009, 69(1): 22–27
5. Liu Z, Bessa M, Liu W K. Self-consistent clustering analysis: An efficient multi-scale scheme for inelastic heterogeneous materials. Computer Methods in Applied Mechanics and Engineering, 2016,

306: 319–341

6. Liu Z, Fleming M, Liu W K. Microstructural material database for self-consistent clustering analysis of elastoplastic strain softening materials. Computer Methods in Applied Mechanics and Engineering, 2018, 330: 547–577
7. Geers M, Yvonnet J. Multiscale modeling of microstructure-property relations. MRS Bulletin, 2016, 41(08): 610–616
8. Liu Z, Wu C, Koishi M. A deep material network for multiscale topology learning and accelerated nonlinear modeling of heterogeneous materials. Computer Methods in Applied Mechanics and Engineering, 2019, 345: 1138–1168
9. Ghaboussi J, Garrett J Jr, Wu X. Knowledge-based modeling of material behavior with neural networks. Journal of Engineering Mechanics, 1991, 117(1): 132–153
10. Ghaboussi J, Pecknold D A, Zhang M, Haj-Ali R M. Autoprogressive training of neural network constitutive models. International Journal for Numerical Methods in Engineering, 1998, 42(1): 105–126
11. Huber N, Tsakmakis C. Determination of constitutive properties from spherical indentation data using neural networks. Part I: The case of pure kinematic hardening in plasticity laws. Journal of the Mechanics and Physics of Solids, 1999, 47(7): 1569–1588
12. Huber N, Tsakmakis C. Determination of constitutive properties from spherical indentation data using neural networks. Part II: Plasticity with nonlinear isotropic and kinematic hardening. Journal of the Mechanics and Physics of Solids, 1999, 47(7): 1589–1607
13. Pernot S, Lamarque C H. Application of neural networks to the modelling of some constitutive laws. Neural Networks, 1999, 12(2): 371–392
14. Haj-Ali R, Pecknold D A, Ghaboussi J, Voyiadjis G Z. Simulated micromechanical models using artificial neural networks. Journal of Engineering Mechanics, 2001, 127(7): 730–738
15. Rumelhart D E, Hinton G E, Williams R J. Learning Internal Representations by Error Propagation. Technical Report. California University San Diego La Jolla Inst for Cognitive Science. 1985
16. Dimiduk D M, Holm E A, Niezgoda S R. Perspectives on the impact of machine learning, deep learning, and artificial intelligence on materials, processes, and structures engineering. Integrating Materials and Manufacturing Innovation, 2018, 7(3): 157–172
17. Unger J F, Könke C. Coupling of scales in a multiscale simulation using neural networks. Computers & Structures, 2008, 86(21–22): 1994–2003
18. Bessa M, Bostanabad R, Liu Z, Hu A, Apley D W, Brinson C, Chen W, Liu W K. A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality. Computer Methods in Applied Mechanics and Engineering, 2017, 320: 633–667
19. Le B, Yvonnet J, He Q C. Computational homogenization of nonlinear elastic materials using neural networks. International Journal for Numerical Methods in Engineering, 2015, 104(12): 1061–1084
20. Lefik M, Boso D, Schrefler B. Artificial neural networks in numerical modelling of composites. Computer Methods in Applied Mechanics and Engineering, 2009, 198(21–26): 1785–1804
21. Zhu J H, Zaman M M, Anderson S A. Modeling of soil behavior with a recurrent neural network. Canadian Geotechnical Journal, 1998, 35(5): 858–872
22. Wang K, Sun W. A multiscale multi-permeability poroplasticity model linked by recursive homogenizations and deep learning. Computer Methods in Applied Mechanics and Engineering, 2018, 334: 337–380
23. Feyel F, Chaboche J L. FE^2 multiscale approach for modelling the elastoviscoplastic behaviour of long fibre SiC/Ti composite materials. Computer Methods in Applied Mechanics and Engineering, 2000, 183(3–4): 309–330
24. Zhu H, Wang Q, Zhuang X. A nonlinear semi-concurrent multiscale method for fractures. International Journal of Impact Engineering, 2016, 87: 65–82
25. Fish J. Practical Multiscaling. 1st ed. UK: John Wiley & Sons, 2014
26. Yuan Z, Fish J. Toward realization of computational homogenization in practice. International Journal for Numerical Methods in Engineering, 2008, 73(3): 361–380
27. Cho K, Van Merriënboer B, Bahdanau D, Bengio Y. On the properties of neural machine translation: Encoder-decoder approaches. 2014, arXiv preprint arXiv:1409.1259
28. Jung S, Ghaboussi J. Neural network constitutive model for rate-dependent materials. Computers & Structures, 2006, 84(15–16): 955–963
29. Lefik M, Schrefler B. Artificial neural network as an incremental non-linear constitutive model for a finite element code. Computer Methods in Applied Mechanics and Engineering, 2003, 192(28–30): 3265–3283
30. Zhu J, Chew D A, Lv S, Wu W. Optimization method for building envelope design to minimize carbon emissions of building operational energy consumption using orthogonal experimental design (OED). Habitat International, 2013, 37: 148–154
31. Furukawa T, Yagawa G. Implicit constitutive modelling for viscoplasticity using neural networks. International Journal for Numerical Methods in Engineering, 1998, 43(2): 195–219
32. Furukawa T, Hoffman M. Accurate cyclic plastic analysis using a neural network material model. Engineering Analysis with Boundary Elements, 2004, 28(3): 195–204
33. Yun G J, Ghaboussi J, Elnashai A S. A new neural network-based model for hysteretic behavior of materials. International Journal for Numerical Methods in Engineering, 2008, 73(4): 447–469
34. Samaniego E, Anitescu C, Goswami S, Nguyen-Thanh V M, Guo H, Hamdia K, Zhuang X, Rabczuk T. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. Computer Methods in Applied Mechanics and Engineering, 2020, 362: 112790
35. Nguyen-Thanh V M, Zhuang X, Rabczuk T. A deep energy method for finite deformation hyperelasticity. European Journal of Mechanics. A, Solids, 2020, 80: 103874
36. Guo H, Zhuang X, Rabczuk T. A deep collocation method for the bending analysis of Kirchhoff plate. Computers, Materials & Continua, 2019, 59(2): 433–456
37. Anitescu C, Atroshchenko E, Alajlan N, Rabczuk T. Artificial neural network methods for the solution of second order boundary value problems. Computers, Materials & Continua, 2019, 59(1): 345–359