Archives of Computational Methods in Engineering
https://doi.org/10.1007/s11831-024-10199-z

REVIEW

Machine Learning Aided Modeling of Granular Materials: A Review

Mengqi Wang[1] · Krishna Kumar[2] · Y. T. Feng[1] · Tongming Qu[3] · Min Wang[4]

Received: 19 June 2024 / Accepted: 17 October 2024
© The Author(s) 2024

Abstract
Artificial intelligence (AI) has become a buzzy word since Google's AlphaGo beat a world champion in 2017. In the past five years, machine learning as a subset of the broader category of AI has obtained considerable attention in the research community of granular materials. This work offers a detailed review of the recent advances in machine learning-aided studies of granular materials from the particle-particle interaction at the grain level to the macroscopic simulations of granular flow. This work will start with the application of machine learning in the microscopic particle-particle interaction and associated contact models. Then, different neural networks for learning the constitutive behaviour of granular materials will be reviewed and compared. Finally, the macroscopic simulations of practical engineering or boundary value problems based on the combination of neural networks and numerical methods are discussed. We hope readers will have a clear idea of the development of machine learning-aided modelling of granular materials via this comprehensive review work.

1 Introduction

The granular material, as a macroscopic continuum, showcases complicated features, involving anisotropy [125, 155], strain localization [155, 158], non-coaxiality [154], and path-and-states dependence [2, 41, 79], under external loading due to their micro-discrete nature. Traditional numerical techniques, such as the finite element method (FEM), finite differential method (FDM), material point method (MPM), discrete element method (DEM), and smoothed particle hydrodynamics (SPH), have been widely employed to investigate the micro/discrete and macro/continuous duality [173] of granular media. Wherein the FEM, SPH, and MPM solve the mechanical responses of granular materials from the macroscopic scale, while the DEM focuses on the mechanical behaviour of granular media at the microscale.

In the mesh-based numerical method, such as FEM [196], the research domain is discretized into finite element meshes that incorporate Gaussian points or grids, and the local mechanical response of the material is characterized by continuum-theory-based phenomenological models embedded in each Gaussian point. The classic constitutive models used in these methods include linear elasticity (e.g. Hooke's law), nonlinear elasticity (e.g. Duncan-Chang [43]), the elastic-perfectly-plastic model (e.g. Mohr-Coulomb model [153], Drucker-Prager model, and hardening soil (HS) model [23]), and the critical-state-based model (e.g. modified cam-clay MCC [137] UH [179, 180], and hypoplastic model [118, 177]), etc.

Different from the mesh-based method, the MPM, a hybrid Eulerian-Lagrangian meshless approach [71, 186], governs the macroscopic deformation of the target body via a set of discretized material points which carry local physical features (e.g. mass, density, and velocity) of the material. In MPM, the information stored on each material point is first projected to the node of the Eulerian mesh on which the motion equation is solved. Then the updated nodal kinematic features are interpolated back to the corresponding material points to update the deformation of the material, making the

✉ Y. T. Feng
y.feng@swansea.ac.uk

✉ Min Wang
minw@lanl.gov

1 Zienkiewicz Centre for Computational Engineering, Faculty of Science and Engineering, Swansea University, Swansea, Wales SA1 8EP, UK

2 Department of Civil, Architecture and Environmental Engineering, University of Texas at Austin, Austin, Texas 78701, USA

3 Department of Civil and Environmental Engineering, Hong Kong University of Science and Technology, Clearwater Bay, Kowloon, Hong Kong SAR, China

4 Fluid Dynamics and Solid Mechanics Group, Theoretical Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, USA

Published online: 29 October 2024

MPM especially superior in solving the large deformation and flow problems of granular materials.

Unlike the continuum-based methods, the DEM, proposed in Cundall and Strack's works [37–39, 147] in the 1970 s, represents the overall behaviour of particle systems by explicitly modelling interactions between particles, where adhesive/frictional contact plays a crucial role. In the early framework of the DEM, the discrete particle is normally simplified as discs for two-dimensional (2D) problems. With the advancement of non-spherical models [22, 54, 94, 165, 172] and corresponding contact theories [47, 50, 194] over the last four decades, the DEM has been firmly established as one powerful numerical tool to model the deformation [116, 151] or flow [42, 185] of the collection of grains from the microscopic scale in engineering and scientific problems.

However, these methods have their intrinsic deficiencies. For mesh-based methods, most constitutive models they leveraged tend to homogenize or smear out the underlying discrete features of granular assemblies [68], which limits their ability to simulate catastrophic instabilities [180] (e.g. landslides and liquefaction) caused by collective particle motion. Furthermore, to perfectly fit one certain experimental phenomenon, constitutive models are increasingly sophisticated and inevitably introduce assumptions [180, 183] or arguments without physical meanings but need much effort to calibrate [129, 182, 191], which limits their further applications in other tests. It is also worth mentioning that the constitutive model developed for granular flow is more complicated than the above-mentioned ones. Typical work can be found in [187], where the effect of particle rotation at the grain level results in the Jaumann derivative in the evolution equation of the macroscopic contact stress. Additionally, when addressing large deformation problems, the local mesh may suffer from severe distortion and thus deteriorate modelling results. While the mesh distortion problem can be avoided in MPM, it requires the information exchange process between the background (Eulerian) grid and material points, resulting in data breaches and significantly increasing computational complexity.

For DEM, although the discrete nature of granular particles can be taken into account, it still confronts some challenges. One is the prohibitive computational expense primarily attributed to the contact detection procedure in DEM simulations of particle collections [171], and this issue is further exacerbated in nonspherical particle assemblies where contact detection involves an iteration-based optimization method [94, 105]. Related to this, to minimize the computational cost, irregular shape particles are typically simplified as spheres (3D) or circles (2D) [9, 93] but may lead to the spuriousness of the bulk mechanical properties in particle assemblies.

In addition to the aforementioned numerical methods used for the modelling of granular media, the requirement of amounts of grain-scale computation has also fostered the emergence of multiscale approaches, such as the FEM-DEM [4, 5, 68, 69, 121] and MPM-DEM [104] approaches, which bridges the micro-features to the computation of macroscopic response of grain materials. Although these methods can compensate for defects of continuum methods to some extent, they are also subjected to the problem of prohibitive computing costs in large-scale simulations.

More recently, the machine learning (ML) method resurged in various fields [3, 98, 99, 166] seems a promising scheme to circumvent the aforementioned shortcomings of traditional numerical methods in simulating the behaviour of granular materials because of its following advantages: 1) the ML or neural network-based surrogate model can directly extract mechanical features of granular materials from raw data without any assumption; 2) due to its remarkable high-dimension mapping capability [40, 78], ML models can approximate desired constitutive laws at high precision by sufficient training data rather than sophisticated formula and well-calibrated physically meaningless parameters; 3) ML models are highly computationally efficient and can instantaneously update the materials' micro or macro-states of materials according to the received deformation information once their training parameters are determined. Resorting to these advantages of the ML method, it is possible to develop the ML-based stress–strain model to replace the traditional constitutive model used in continuum-level computational approaches. Meanwhile, employing ML algorithms to establish surrogate contact/interaction models for both micro-grains or macroscopic material points also shows promise in alleviating the computational burden for mesh-based and meshless numerical methods (e.g. MPM and DEM).

This paper primarily aims to provide one comprehensive review of the latest advances in ML-assisted granular material modelling at both microscopic and macroscopic levels from the following aspects: 1) the application of ML models in microscopic grain scale computation; 2) the development of ML-based constitutive models of granular materials; 3) the development of ML-aided macroscopic simulation of granular materials, including both the macroscopic kinematic features-based ML model and the application of the ML method in mesh-based numerical methods (e.g. the FEM-ML framework).

The remaining paper is structured as follows: Sect. 2 compares and summarises the architecture and features of some typically used neural networks in ML-aided modelling of granular materials. A brief review of grain information-based ML simulations, including the ML-based inter-particle contact feature and kinematic feature modelling of grain systems, is provided in Sect. 3 from both their advantages and deficiencies of each aspect. Section 4 showcases the latest development of ML-based constitutive studies of granular materials from aspects of the used training data sources and training methodologies of different ML frameworks. Furthermore, the prediction capability of varying ML models for stress–strain response under different loading paths is also intuitively assessed in this section. Section 5 reviews several commonly used macroscopic simulation schemes of granular materials and their coupling approaches with the ML methods, followed by a specific biaxial modeling example with the FEM-ML scheme. The discussion and concluding remarks are made in the last two sections.

2 Typical Neural Networks in ML-Aided Modeling of Granular Materials

A thorough comprehension of different neural networks is indispensable for effectively applying them to specific tasks. In this section, seven frequently used ML models in the simulation of granular materials are introduced, including the multi-layer perceptron (MLP), the basic recurrent neural network (RNN) as well as its family members, i.e. the long-short-term memory (LSTM) and gated recurrent unit (GRU), temporary convolutional neural networks (TCNN), convolutional neural network (CNN), and graph neural networks (GNN). According to their unique properties, they are grouped into three categories: 1) single-step-based networks, 2) multi-step-based or time-sequence networks, and 3) spatial information-based neural networks.

2.1 The Single-Time Step-Based Neural Network

A notable characteristic of single-step-based neural networks like the MLP is that their prediction relies solely on the input of the current time step, necessitating a strict surjective relationship between the input and output of the network.

Figure 1 presents the architecture of the MLP with an input layer, hidden layer, and output layer, in between connected by trainable parameters (weight matrixes $\mathbf{W_1}, \mathbf{W_2}$, and bias vectors $\mathbf{b_1}, \mathbf{b_2}$). The training process of the MLP consists of two parts: the feedforward and backpropagation procedures. To obtain the prediction values O, i.e. $(\hat{y}_1, ..., \hat{y}_m)$, the feedforward process of input data $\mathbf{x}(x_1, x_2, ..., x_n)$ can be expressed as:

$$\mathbf{H} = g\left(\mathbf{W}_1\mathbf{x} + \boldsymbol{b}_1\right) \tag{1}$$

$$\mathbf{O} = g\left(\mathbf{W}_2\mathbf{H} + \boldsymbol{b}_2\right) \tag{2}$$

where $\mathbf{H}$ means the output of the hidden layer; f and g are activation functions used in hidden and output layers, respectively. After finishing the feedforward process, the error between the predicted and target values is calculated and returned to each layer to update the trainable parameters

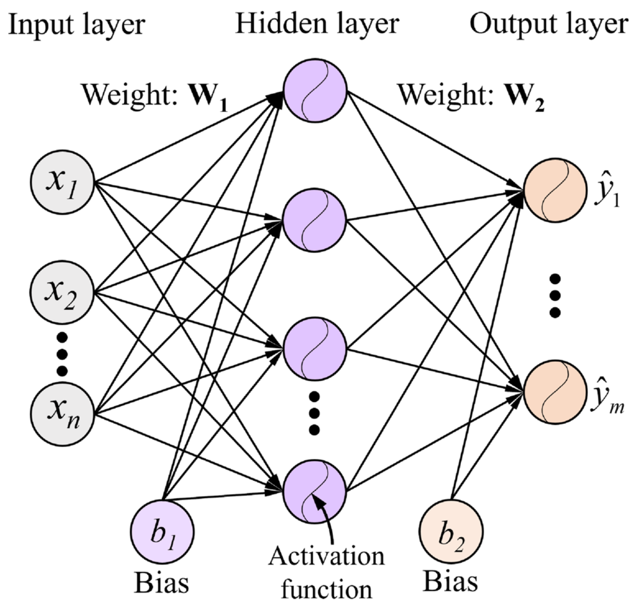


Fig. 1 The architecture of the MLP with one hidden layer

via the backpropagation algorithm. The whole process is iterated until the computed errors reach a minimum value or remain constant. The simple architecture and strong capability of mapping extremely intricate functions [97, 98, 138] make the MLP widely applied in ML-assisted numerical techniques for both regression and classification tasks [81, 95].

However, the MLP also has one apparent shortcoming. In history-dependent problems, the MLP has no architecture to record history variables to distinguish history states, and thus highly relies on the artificially added history information.

2.2 The Time-Sequence Neural Network

The most notable feature of time-sequence ML models involves incorporating historical information to forecast the current time step. This means the output of the neural network depends not only on the current input but also on past information, making them well-suited for tackling time series forecasting tasks.

2.2.1 Recurrent Neural Networks

One prominent example of a time-sequence ML model is the basic RNN, which achieves this function by substituting each node of the hidden layer in the MLP with an RNN neuron [46, 168]. Such history-dependent property makes it originally prevalent in language modeling [64, 149] and machine translation [10, 150].

As shown in Fig. 2, the input vector $\mathbf{x}(\mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(t)})$ is fed into each RNN neuron featured with the recurrent connection (the orange dash line), which can be envisioned as a succession of unrolled cells across time steps, and output the hidden state $\mathbf{H}^{(t)}$ of the current time step. In each cell, the hidden state of the last time step and the input of the current time step are considered together to generate the hidden state of the next time step until the hidden state of time step t is obtained, enabling the history information to be integrated into the output layer to predict current time step.

The feedforward process in one RNN neuron can be expressed as:

$$\mathbf{H}^{(t)} = \tanh\left(\mathbf{W}^{(t)}\mathbf{x}^{(t)} + \mathbf{U}^{(t)}\mathbf{H}^{(t-1)} + \boldsymbol{b}\right) \tag{3}$$

$$\mathbf{O}^{(t)} = g\left(\mathbf{W}_2\mathbf{H}^{(t)} + \boldsymbol{b}_2\right) \tag{4}$$

where tanh is the activation in each RNN cell; the weight matrices connecting the input and RNN layer, as well as the adjacent cells, are denoted by $\mathbf{W}^{(t)}$ and $\mathbf{U}^{(t-1)}$, respectively; $\boldsymbol{b}$ denotes the bias between the input layer and the RNN layer.

However, the basic RNN architectures normally suffer from gradient vanishing or exploding problems along the timeline, resulting in their weak capability to capture long-term dependencies. Therefore, more elaborate RNN architectures, such as the long short-term memory (LSTM) [77] and the gated recurrent (GRU) [31], are innovated and more commonly used.

As illustrated in Fig. 3a, which shows the memory cell of LSTM, the key distinction between the standard RNN and LSTM is the introduction of internal state $\mathbf{C}_t$ and the novel inclusion of multiplicative gates i.e. the input gate $\mathbf{I}_t$, input node $\tilde{\mathbf{C}}_t$, forget gate $\mathbf{F}_t$, and output gate $\mathbf{O}_t$, in each RNN neuron, which can be calculated as:

$$\mathbf{F}_t = \sigma\left(\mathbf{W}_F\mathbf{x}^{(t)} + \mathbf{U}_F\mathbf{H}^{(t-1)} + \mathbf{b}_F\right) \tag{5}$$

$$\mathbf{I}_t = \sigma\left(\mathbf{W}_I\mathbf{x}^{(t)} + \mathbf{U}_I\mathbf{H}^{(t-1)} + \mathbf{b}_I\right) \tag{6}$$

$$\mathbf{O}_t = \sigma\left(\mathbf{W}_O\mathbf{x}^{(t)} + \mathbf{U}_O\mathbf{H}^{(t-1)} + \mathbf{b}_O\right) \tag{7}$$

$$\tilde{\mathbf{C}}_t = \tanh\left(\mathbf{W}_C\mathbf{x}^{(t)} + \mathbf{U}_C\mathbf{H}^{(t-1)} + \mathbf{b}_C\right) \tag{8}$$

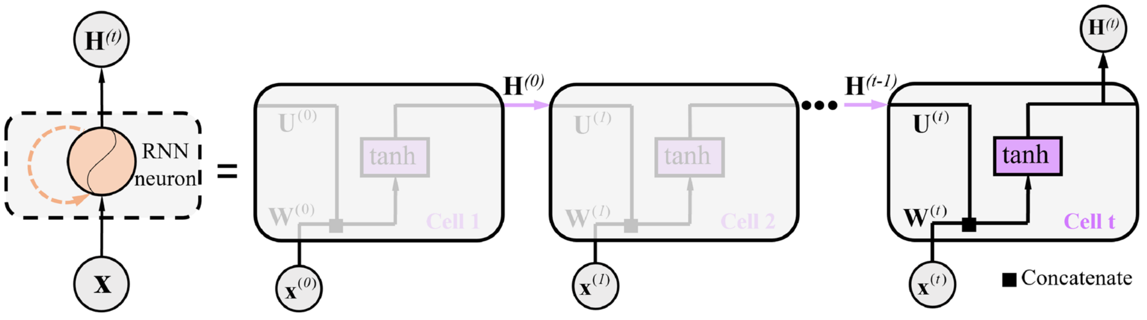


Fig. 2 The recurrent neuron in the basic RNN

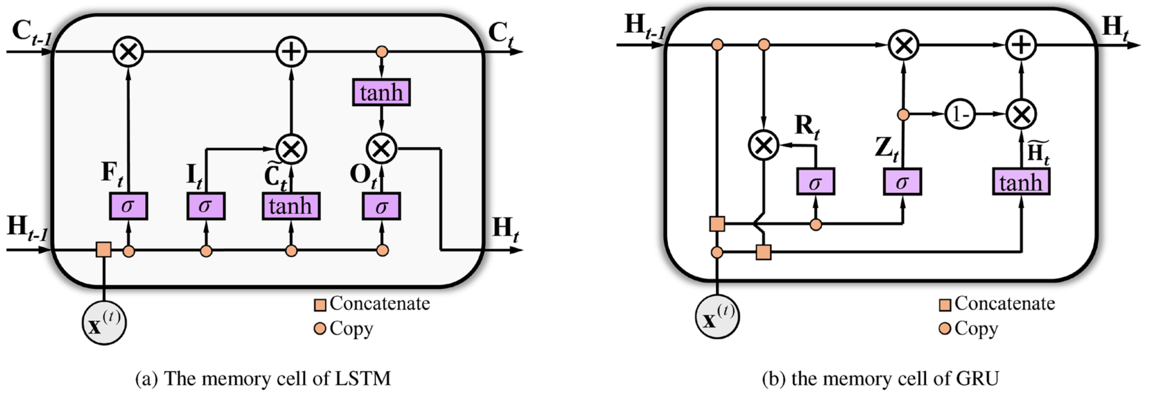


Fig. 3 The different memory cells in RNNs

where $\mathbf{I}_t$ and $\tilde{\mathbf{C}}_t$ determine how much new information is adopted and $\mathbf{F}_t$ governs how much old internal state $\mathbf{C}_{t-1}$ is inherited to update the new internal state $\mathbf{C}_t$:

$$\mathbf{C}_t = \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \tilde{\mathbf{C}}_t \tag{9}$$

and then, the current hidden state $\mathbf{H}^{(t)}$ which relies on the obtained internal state $\mathbf{C}_t$ and output gate $\mathbf{O}_t$ is updated by:

$$\mathbf{H}^{(t)} = \mathbf{O}_t \odot \tanh\left(\mathbf{C}_t\right) \tag{10}$$

where $\odot$ denotes the elementwise product operator; σ here represents the sigmoid activation function.

On the other hand, the gated recurrent unit (GRU) neural network shown in Fig. 3b provides a streamlined version of LSTM with a comparable performance but speeds up computation by only introducing two gates: update gate $\mathbf{Z}_t$ and reset gate $\mathbf{R}_t$ to the standard RNN cell. The output of these two gates can be expressed as:

$$\mathbf{R}_t = \sigma\left(\mathbf{W}_R\mathbf{x}^{(t)} + \mathbf{U}_R\mathbf{H}^{(t-1)} + \boldsymbol{b}_R\right) \tag{11}$$

$$\mathbf{Z}_t = \sigma\left(\mathbf{W}_Z\mathbf{x}^{(t)} + \mathbf{U}_Z\mathbf{H}^{(t-1)} + \boldsymbol{b}_Z\right) \tag{12}$$

where $\mathbf{R}_t$ determines how much previous hidden state is kept in the current candidate hidden state $\tilde{\mathbf{H}}^{(t)}$:

$$\tilde{\mathbf{H}}^{(t)} = \tanh\left[\mathbf{W}\mathbf{x}^{(t)} + \mathbf{U}\left(\mathbf{R}_t \odot \mathbf{H}^{(t-1)}\right)\right] \tag{13}$$

and $\mathbf{Z}_t$ controls the extent to which the new hidden state $\mathbf{H}^{(t)}$ matches the old state $\mathbf{H}^{(t-1)}$ and how similar it is to the new candidate state $\tilde{\mathbf{H}}^{(t)}$, which can be expressed as:

$$\mathbf{H}^{(t)} = \mathbf{Z}_t \odot \mathbf{H}^{(t-1)} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{H}}^{(t)} \tag{14}$$

Incorporating more sophisticated gate architectures in the standard RNN can effectively alleviate the gradient vanishing or explosion problem, but such structures also lead to the rapid growth of training parameters and excessive memory usage to store the result of each cell with the increase of the time step. Furthermore, the nature that the prediction of the current hidden state must wait for the completion of its predecessors in RNNs limits their parallelism.

2.2.2 The Temporal Convolutional Neural Network

The temporal convolutional neural network (TCNN) [11], a variant of the convolutional neural network (CNN) which will be introduced in detail later in Sect. 2.3, is another ML model that can be used in time-sequence forecasting problems. Different from the RNN which uses the memory cells to integrate history information, the TCNN extracts the past information by scanning the input data along the time-step direction using filters with the shape of *kernel size* × *Depth*, in which trainable parameters are stored.

Figure 4 provides a detailed feedforward process in the TCNN. The input x is composed of a 2D array with one representing time steps and the other representing depth (i.e. the number of columns of the input data). To guarantee only the past and current data are involved and an equal length between the input and output columns of each convolution layer, the causal convolution (CC) in which the forefront of input data is padded with zero along the direction of time sequence. Where the number of the padded zero line is equal to $kernelsize - 1$. In the CC procedure, n filters will simultaneously make dot product calculations with the corresponding elements of input data along the direction of the time step with fixed stride s and obtain n corresponding output series $(o_1, o_2, \dots, o_n)$. The obtained output vectors are then flattened to one column and fed into the output neuron to predict the value of the current time step.

As the TCNN filters are independent, it is easy to parallelize the training process across GPU cores [11]. Furthermore, the number of training parameters is solely dependent on the number of filters, which means when the time steps of the

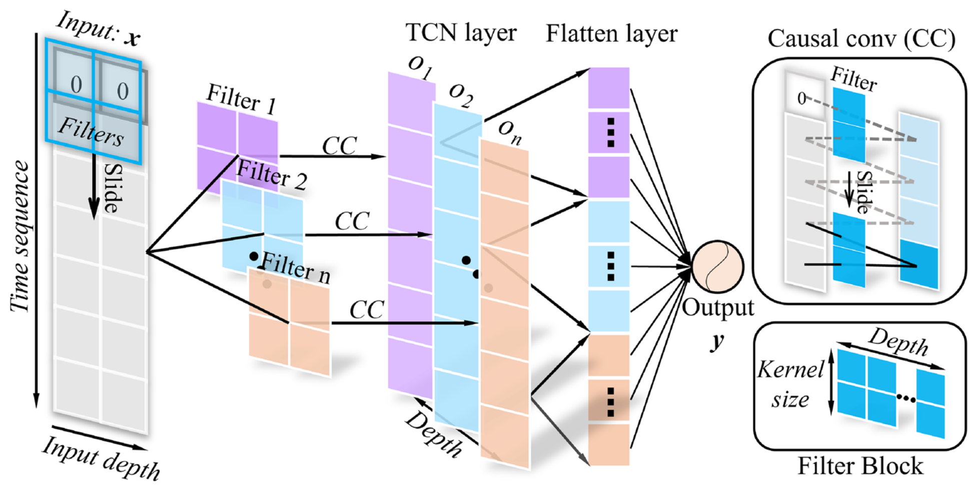


Fig. 4 The feedforward process in the temporal convolution neural network

training data increase, there are no extra training parameters introduced. In addition, the transfer learning process is easy to conduct between two TCNN models as long as the number and shape of filters remain constant.

However, the TCNN also has some notable disadvantages. Unlike the LSTM or GRU in which the needed history information is adjustable to predict the current time step, the TCNN cannot filter out the redundant historical information when transferring a model from a domain where only little memory is needed (i.e., small *kernel size*) to a domain where much longer memory (i.e., large *kernel size*) is required. In addition, the development of one TCNN normally needs several CC layers, which may need a large memory to store the trained parameters, resulting in a lower training efficiency.

2.3 The Geometry Information-Based Neural Networks

The prediction of the granular mechanical response is traditionally considered a time-sequential problem and mainly treats granular information as vectors to develop the ML surrogate model, which ignores the rich spatial features of granular assembly. The recent development of ML technology also enables researchers to directly extract physical information of particles from figures or graphics to predict the behaviour of granular media. The two most commonly used geometry information-based neural networks are the CNN and the graph neural network (GNN), respectively.

2.3.1 The Convolutional Neural Network

The advent of CNN [57, 167], has revolutionized fields like image recognition [34, 35] and video analysis [8, 83]. Leveraging CNNs, it becomes feasible to predict granular properties by encoding the structural characteristics of particle assemblies into a pixel matrix. Similar to the TCNN which extracts the temporal features with filters, the CNN collects spatial features of figures by filters.

Figure 5 illustrates the whole feedforward process in CNN. Unlike the TCNN, which requires input data in a sequential time-based format, CNN regards the image with abundant spatial information as the input data; thus, the time dimension is not required. Instead, the input is represented by its height (h), width (w), and channel (c) dimensions, where c represents the number of input images. Correspondingly, the dimension of the filter is changed to 3D but with the same height and width as the kernel size k. Furthermore, unlike TCNN, where the filter can only move along the direction of the time step, the filter in CNN scans the input data from both the height and width directions at a fixed stride. It is worth noting that to keep the constant shape between the input and output, zero-padding around the input pixel is still required, similar to the approach used in TCNN.

Compared to TCNN, CNNs have some unique features. The first is its interpretability, as CNNs can be visualized to show which part is significant for the prediction. The second notable feature is its location invariance. Filters are trained to detect various spatial or shape features, such as edges or corners, enabling CNNs to recognize objects in images regardless of their position.

However, CNNs also have some limitations. For example, CNNs are not suitable for non-grid input data. Furthermore, their performance can be impacted when input images are occluded or contain significant noise. In addition, training one CNN to recognize meaningful patterns normally needs

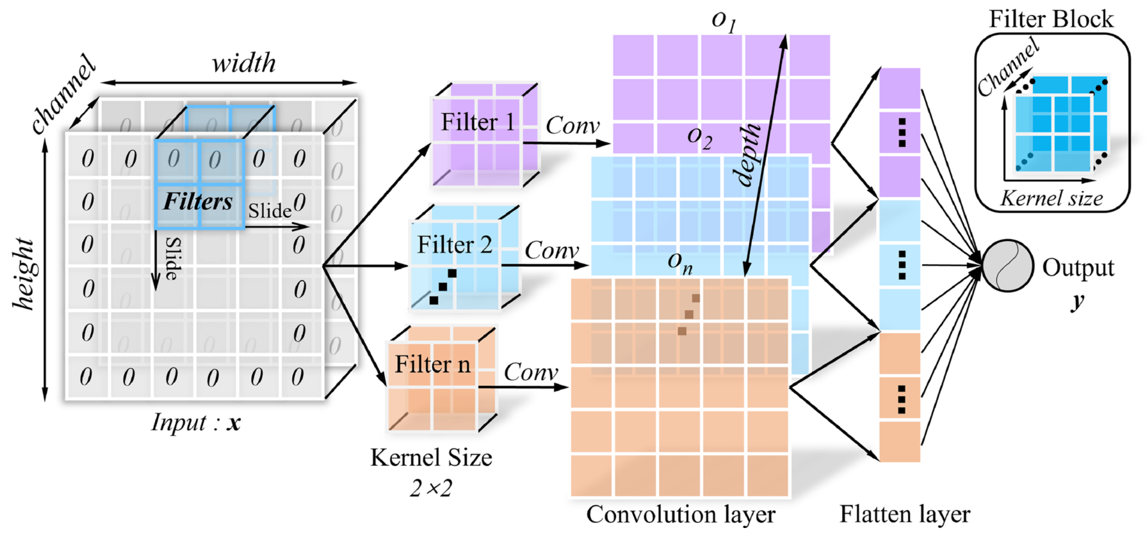


Fig. 5 The feedforward process of the convolutional neural network

a large amount of labelled data, which is a great challenge when data is expensive to acquire.

2.3.2 Graph Neural Network

In addition to CNN, the GNN [120, 140, 176] are also capable of learning the structural information and topological features of graphs, which makes them a powerful tool for capturing the dynamic features of grain assemblies [6, 32].

An example architecture of the GNN is illustrated in Fig. 6. The GNN takes one graph consisting of six nodes as the input. In the input layer, also called the 0^{th} layer, the node feature of each vertex is represented by x_v, and all node information $\boldsymbol{h}_v^0\left(h_1^0, h_2^0, \dots, h_6^0\right)$ is transported to the GNN layer1. In the GNN layer1, the node feature is alternately updated and the newly-generated output $\boldsymbol{h}_v^1(h_1^1, h_2^1, \dots, h_6^1)$ is passed to the next GNN layer. This process is repeated until the output layer acquires the prediction result z_v where $z_v = \boldsymbol{h}_v^K$, and K denotes the output layer.

In GNN, the feature of each node h_v^k is updated by the message passing (MP) process, which aggregates the information received from their adjacent nodes u. There are normally three different ways to integrate the message from the neighbouring nodes, i.e. mean, sum, and max. Regarding the mean method as one example, the MP procedure of the node v in the k^{th} GNN layer can be expressed as:

$$h_v^k = f\left(\mathbf{W}^k \sum_{u\in N(v)} \frac{h_u^{k-1}}{N(v)} + \mathbf{b}^k h_v^{k-1}, \forall k \in \{1, \dots, K\}\right) \tag{15}$$

where h_u^{k-1} represents the output features of nodes that connect to the vertex v from the $(k-1)^{th}$ layer. $N(v)$ denotes the number of adjacent nodes of vertex v. The item $\sum_{u\in N(v)} \frac{h_u^{k-1}}{N(v)}$ aims to aggregate the received message of node v from the previous layer in an average way. $\mathbf{W}^k$ and $\mathbf{b}^k$ are trainable parameters stored on each edge in the k^{th} GNN layer. f represents the activation function (e.g. ReLU).

Compared with the MLP, the RNNs, CNNs, and the GNN have some distinct advantages. Different from RNNs and CNNs, whose input data needs to be arranged in one certain order, the GNN acquires knowledge by updating the message stored in each node and thus disregarding the order of input data. In addition, the interaction relationship between different objects can be captured by the edges of graphs, but networks, like the MLP and RNNs, cannot explicitly reflect this dependency relationship. Therefore, the GNN is quite suitable for analyzing kinematic features of structural scenarios, e.g. granular systems.

However, it is worth noting that the GNN also confronts many challenges. One obvious limitation is its vulnerability to the modification of graph structure. When nodes and edges are added or removed, the GNN cannot adaptively adjust the network structure. Furthermore, similar to CNNs,

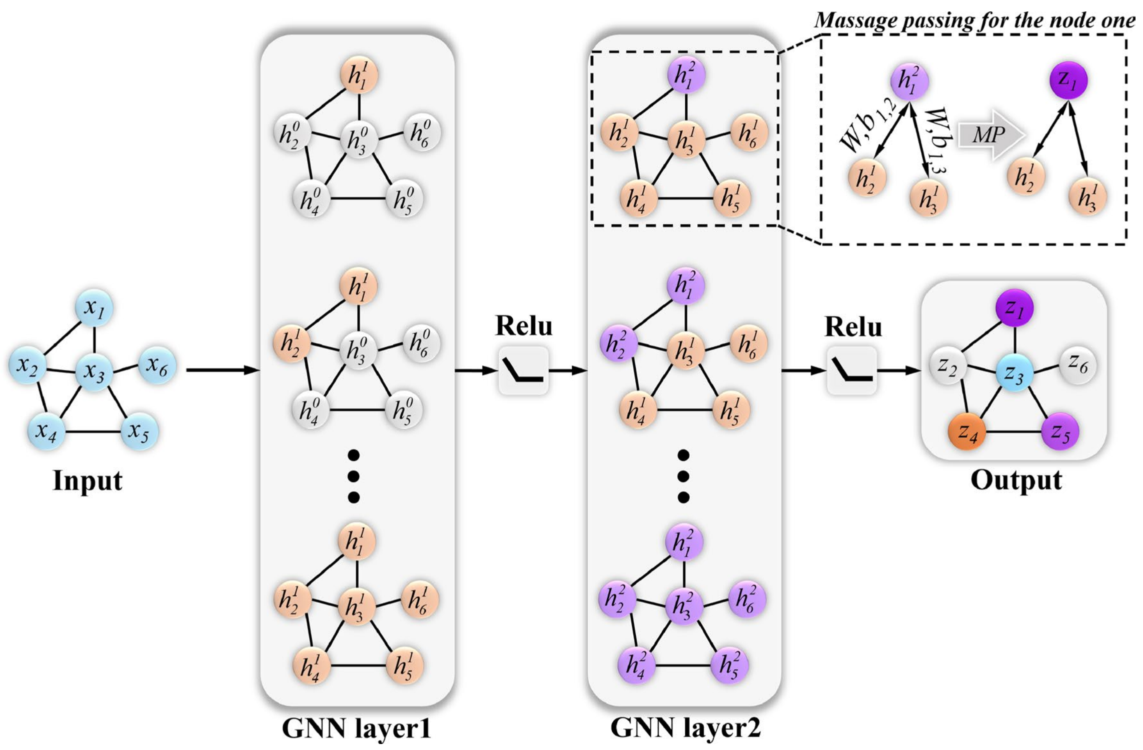


Fig. 6 The architecture of the graph neural network

the GNN is also not robust to noise. Introducing a slight noise into the graph via node perturbation or addition/deletion of edges can cause an adversarial impact on the output of GNNs. In addition, in graphs consisting of numerous nodes, a large amount of training parameters are required to represent relationships between adjacent objects, which results in a low training efficiency of GNN.

2.4 Summary

The networks used in the ML-aided granular material simulation have their unique architectures, advantages, and shortcomings, which are summarized in Table 1. It is found that all these neural networks excel at solving non-linear or high-dimensional problems. Meanwhile, to enhance their versatility in addressing various complex tasks, neural network architectures tend to become more sophisticated, for example, the evolution of basic RNN to the LSTM and GRU. The growing structural complexity of neural networks does enhance their ability to approximate or predict more intricate functions but meanwhile increases training parameters, potentially reducing the training efficiency.

Furthermore, different neural networks possess distinct advantages and limitations. For instance, the MLP is particularly useful in approximating one-to-one or many-to-one mappings, while RNNs (including LSTM and GRU) and TCNNs are more adept at time series forecasting tasks, owing to their unique architectures. While CNNs and GNNs perform well at extracting spatial features from the input data. Instead, the focus should be on identifying the most suitable neural network for the designated task based on its specific strengths and weaknesses.

In addition to the seven typical neural networks used in ML-assist granular materials modelling, other neural networks, such as the radial basis function neural network (RBFNN), Bi-LSTM, residual CNN, etc., are also developed based on the aforementioned typical neural networks. These

Table 1 Features of seven typically used neural networks in the ML-aided granular materials simulation

Neural networks	Advantages	Disadvantages
MLP	(1) Simple architecture; (2) High training efficiency	(1) Requiring artificially added history variables in time-sequence problems; (2) Gradient vanishing or explosion; (3) Sensitive to noise and irregularities
RNN	(1) No extra history variables are needed in sequential prediction; (2) Strong non-linear mapping capability	(1) More complex network architecture than MLP; (2) More training parameters than MLP; (3) Gradient vanishing or explosion; (4) Weak ability to record long-history information; (5) Weak parallelism
LSTM	(1) No extra history variables are needed in sequential prediction; (2) Strong non-linear mapping capability; (3) Eliminate the gradient vanishing or explosion by the gate structure; (4) Strong ability to capture long-term dependencies	(1) Increased structural complexity than RNN; (2) More training parameters than RNN; (3) Weak parallelism
GRU	(1) Similar to LSTM; (2) Simpler gate structures than LSTM	Similar to LSTM
TCNN	(1) No extra history variables are needed in sequential prediction; (2) Strong non-linear mapping capability; (3) Excellent parallelism; (4) Good portability of trained parameters; (5) Suitable for longer history information	(1) Numerous training parameters; (2) Weak ability to filter out the redundant history information

CNN	(1) Strong high-dimensional mapping capability;
	(2) Excellent parallelism;
	(3) Good portability of trained parameters;
	(4) Good visualization;
	(5) Location invariance of input;
	(6) Strong ability to get rich spatial features
		(1) Numerous training parameters;
		(2) Not available in non-grid input data;
		(3) Weaker resistance to noise;
		(4) Requirement for large amounts of labelled data

GNN	(1) Available in non-grid input data; (2) Strong ability to get rich spatial features; (3) Strong ability to reflect the relationship of adjacent objects	(1) Numerous training parameters; (2) Vulnerability to the modification of graph structure; (3) Less resistance to noise

networks inherit the strengths and limitations of their predecessors, but will not be discussed in detail here.

3 The Microscopic Grain Information-Based ML Models

In the field of granular mechanics, DEM has been very popular for modelling the mechanical behaviour of various granular materials and related engineering problems, such as landslides [109], shear deformation of soil and sand [92, 152], failure of tunnel surface [184] and fluidized bed [111]. While these methods can reflect the discrete nature of granular media from the grain scale to a certain degree, they typically suffer from intensive computational costs. The advent of deep learning methods offers a potential way to alleviate the computational burdens by integrating ML models with these conventional (microscopic) grain/particle-based techniques. In this section, a concise review of relevant studies in this area is presented. Before that, a brief introduction to the basic framework of the DEM is given.

3.1 The Basic Framework of the Discrete Element Method

In contrast to the continuum approach, the DEM represents granular material as an assembly of distinct particle entities, and the overall (macroscopic) behaviour of the system is governed by inter-particle contacts over time, making it superior to address the large-deformation problem.

Figure 7 illustrates the fundamental computational steps of the DEM, encompassing the following stages:

(a) System initialization: This step involves 1) assigning material properties λ (e.g. mass m and friction coefficient μ) to particles within the current system; and 2) initializing the geometry features $\boldsymbol{X}^{(t)}$ (e.g. shape, size, and positions) [36, 51, 94, 106, 126] and state parameters $\chi^{(t)}$ (e.g. velocity $\boldsymbol{v}^{(t)}$, angle velocity $\boldsymbol{w}^{(t)}$, acceleration $\boldsymbol{a}^{(t)}$, and angle acceleration $\boldsymbol{\alpha}^{(t)}$ to each particle.

(b) Contact detection and resolution: identifying pairs of particles that are in contact through collision detection algorithms [49]. Subsequently, computing the contact features (e.g. contact norm $\boldsymbol{n}$, tangent $\boldsymbol{t}$, and corresponding inter-particle overlaps δ_n, $\Delta\delta_t$) between each pair using collision resolution algorithms.

(c) Contact force and torque calculation: utilizing the acquired contact features and contact model [39, 76, 85], the resultant contact force $\mathbf{F}^{(t)}$ and contact torque $\mathbf{M}^{(t)}$, which respectively govern the translational and rotational motion of particles, are calculated.

(d) State variable update: the acceleration $\boldsymbol{a}^{(t+\Delta t)}$ and the angular acceleration $\boldsymbol{\alpha}^{(t+\Delta t)}$ are respectively updated with Newton's second law with obtained contact force and torque. This update process further refreshes the state variables, including both the $\boldsymbol{v}^{(t+\Delta t)}$ and $\boldsymbol{w}^{(t+\Delta t)}$.

(e) Position update: the geometry information of the particle (i.e. their location) is updated based on newly obtained state variables. After the particle positions are updated within the current time step, a new iteration will begin.

3.2 The ML-Aided Discrete Element Modeling

In the traditional DEM calculation process, the contact detection and resolution process are the most computationally intensive steps [95, 171]. Therefore, leveraging the superior computational efficiency inherent in ML models, the development of ML-based models for contact detection and resolution shows significant potential in accelerating the calculation process of the DEM.

Within the ML-enhanced DEM framework, the contact detection and resolution processes are achieved through a classification and regression neural network, respectively. As shown in Fig. 8 and Fig. 9, the key difference between these two networks is their output layers. In the classification network, the output of the final layer is 0 or 1, indicating the contact status of two particles (one considered as an object grain and the other as a cue particle). While the regression network outputs contact features such as contact

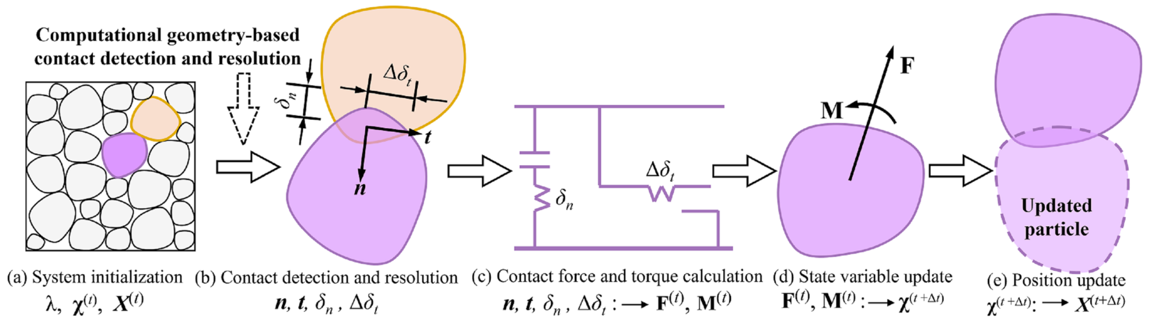

Computational geometry-based contact detection and resolution
δ_n
$\Delta\delta_t$
t
n
δ_n
$\Delta\delta_t$
M
F
Updated particle
(a) System initialization
λ, $\chi^{(t)}$, $\boldsymbol{X}^{(t)}$
(b) Contact detection and resolution
$\boldsymbol{n}$, $\boldsymbol{t}$, δ_n, $\Delta\delta_t$
(c) Contact force and torque calculation
$\boldsymbol{n}, \boldsymbol{t}, \delta_n, \Delta\delta_t : \longrightarrow \mathbf{F}^{(t)}, \mathbf{M}^{(t)}$
(d) State variable update
$\mathbf{F}^{(t)}, \mathbf{M}^{(t)} : \longrightarrow \chi^{(t+\Delta t)}$
(e) Position update
$\chi^{(t+\Delta t)} : \longrightarrow \boldsymbol{X}^{(t+\Delta t)}$

Fig. 7 The computation process in DEM

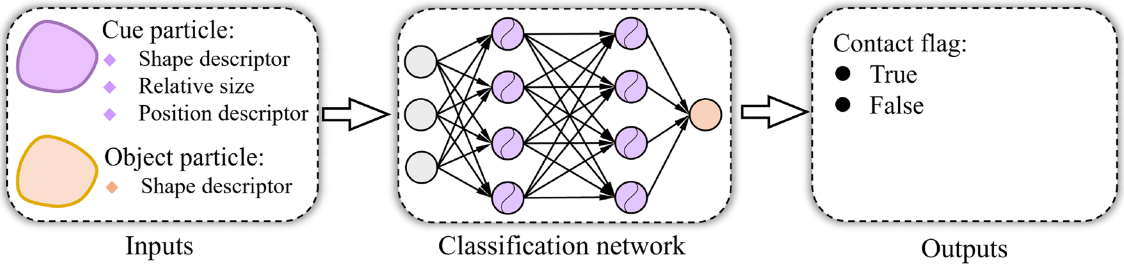


Fig. 8 The classification neural network for contact detection [95]

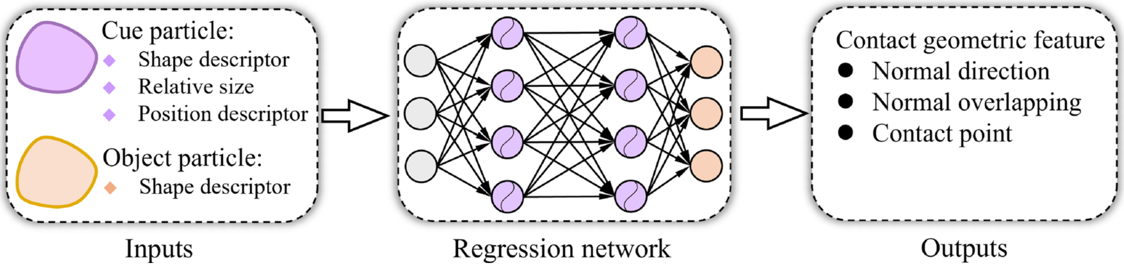


Fig. 9 The regression neural network for contact resolution [95]

point, normal, and overlap between two particles given their requisite geometric features.

Lai et al. [95] extracted the contact status and contact features of particles with elliptical and arbitrary shapes by a classification network and a regression network, respectively. Both these two networks take the same parameters as input, including the shape parameters of the object grain, and the size, shape as well as position parameters of the cue particles. The obtained two networks are then embedded into the DEM algorithm to model cases, e.g. random packing, iodometric compression, angle of repose, packing, and compression of arbitrarily irregular-shaped particles. Similarly, Hwang et al. [81] used one classification and one regression ANN to respectively predict the contact states (i.e. detached or intersected) and contact features of two identical irregular particles, including the mean contact point, averaged norm vector of overlapped vertices, and inter-penetration depth. In their work, to train these two networks, the labelled contact properties of two particles were computed by the deepest point method according to their relative position and orientation which were fed into ML models as the input.

3.3 The ML-Based Grain-Level Kinematic Features Simulations

The evolution of the microstructure in granular materials, influenced by the kinematic features of particles, significantly affects the mechanical behaviour of the material. Therefore, many researchers have endeavoured to develop ML models that can directly capture the particle motion laws without calculating the contact forces using empirical or analytical contact models.

Based on the ML method, both the collision law of sparse objects and the deformation of dense grain media have been widely investigated. For instance, in the work of Katerina et al [53], the trajectory of the billiard is predicted via a CNN which takes the current and previous image information of the system as input and outputs the velocity of the goal object in the future time steps. Meanwhile, a novel object-centric prediction method is used to enhance the translational invariance of the acquired physical laws by the ML model. Wu et al. [175] integrated the ML model into the physical engine to perceive dynamic features of objects in the future time step according to received physical information (e.g. position, friction coefficient, shape) of objects at previous time steps. Battaglia et al [18] embedded the physical state of the system at the previous step into one graph to construct the interaction network which can reason the collision rules of objects in the complicated system interact in the future step. Based on the ML method, Chang et al. [28] built a Neural Physics Engine (NPE) which takes the past pair velocity of both the goal and its adjacent objects as input to predict the velocity of objects at the next time step in the system.

On the other hand, ML models trained with the data generated by DEM simulations have been employed to accelerate the computing of DEM simulations by replacing the contact models with ML models. In the work of

Ummenhofer et al [156] and Lu et al [111], a CNN that can predict the collision laws of inter-particles was constructed with continuous filters to substitute the direct computation of particle-particle/boundary collisions in DEM modelling to accelerate the simulation of grains flow. Li et al [102] designed a GNN that takes the acquired static microstructure of grain packing from DEM simulation as input to predict the contact force of the grain system under the uniaxial compression condition. Cheng et al [30] used the particle and contact kinematical data obtained from a DEM simulator to train the GNN which can estimate the contact forces of granular assemblies under the uniaxial compression condition. Bapst et al [14] trained GNN with the solely initial position information of the grains system to represent the long-term evolution of a glassy system under the shear condition. Mayr et al [119] develop a Boundary-Graph Neural Network (BGNN) to model the interaction of particles with complex boundary conditions. Virtual nodes are inserted into the graph dynamically to represent the boundary surface regions within the cutoff radius of particles and have features encoding triangle orientation representing the boundary. The authors successfully replicate the simulation of 3D granular flows through hoppers, rotating drums, and mixtures. Kumar et al. [90]hypothesize that the GNN messages encode latent representations that preserve the underlying interaction laws between particles in a DEM simulation. The sparse representation of the GNN messages $\left(\mathrm{e}_{\mathrm{k}} \leftarrow \phi\left(\mathrm{e}_{\mathrm{k}}, \mathrm{v}_{\{\mathrm{r}_{\mathrm{k}}\}}, \mathrm{v}_{\{\mathrm{s}_{\mathrm{k}}\}}, \mathrm{u}\right)\right)$ is a learned linear combination of the true forces. To learn a maximally sparse message representation, the authors sort the message vector components by standard deviation and enforce an L1 regularization, forcing the GNN to describe the messages in a minimal vector space. With this approach, they successfully recover the fundamental linear spring interaction law $F_n = k_n \cdot \mathrm{abs}(\Delta x - r_i - r_j)$ based only on the kinematics of the DEM particles.

We can see most kinematic features-based ML models take the position and velocity of individual grain at previous and current time steps as input and directly output the acceleration of the corresponding particle in the next step to update the current grain state of the whole system, which bypasses the contact force and torque calculation, thereby significantly improving the computational efficiency. However, the existing research indicates that the majority of obtained models are trained using data collected from granular systems composed of circular or spherical particles, and thus ignore the influence of particle shape on the grain trajectory.

3.4 Summary

This section summarizes the work related to the grain information-based ML models from two aspects. The first is developing the ML-based contact model based on contact state and contact geometric features in particle-based numerical methods, which has received comparatively less attention from researchers. The second aspect is leveraging appropriate ML models, e.g. (CNN or GNN), to predict the kinematic features for both sparse and dense grain systems at the microscopic scale. Both of these two directions show promise in reducing the intensive computation cost in particle/grain-based numerical techniques. A summary of the advantages and limitations of ML-based microscopic grain information models is provided in Table 2.

Compared to the traditional contact models, ML-based contact models offer one significant advantage by directly outputting contact features and relationships between two particles given their positions, bypassing the traditional contact detection and resolution process, which accelerates the overall computational process of the DEM. Furthermore, the ML-based contact models have a strong capability to

Table 2 An overview for the ML-based microscopic grain information models

ML-aided microscopic modelling	Advantages	Disadvantages
ML-based contact models	(1) High computational efficiency compared to traditional contact models; (2) Strong ability to account for the influence of particle shape; (3) User friendly	(1) The training data normally incorporates contact assumptions; (2) Limited by the completeness of the training data
ML-aided grain-level kinematic feature simulation	(1) Lower computational complexity; (2) High computational efficiency	(1) Error accumulation problem; (2) Ignoring the influence of particle rotation; (3) Weaker suitability to the system where particles are added or removed

account for the influence of particle shape, as the trained ML model can instantly provide contact features when given the necessary geometric information of particles, regardless of their shape. This represents an improvement over traditional contact detection methods, which often simplify grains into spheres (3D) or circles (2D) to improve computational efficiency but overlook the influence of the rotation resistance of particles. Additionally, although more advanced contact detection algorithms and contact models with rigorous theoretical foundations have been developed, their implementation remains challenging for engineers and researchers unfamiliar with computational geometry [47, 48].

In ML-aided grain-level kinematic feature simulations, the fine-tuned geometry information networks can simultaneously predict the acceleration of all grains based on their positions from the previous time step. This approach eliminates the need for the iterative calculation of particle accelerations typically required in traditional discrete element modeling, thereby significantly enhancing simulation efficiency.

However, it is also important to acknowledge the limitations associated with each aspect. The development of ML-based contact models is primarily hindered by the completeness and quality of the training data, including contact geometry information. The contact state and contact features are directly related to the relative position and geometry shape of two particles. To create an ML model that can accurately predict contact information for any contact position between two particles, the training data must encompass a wide range of contact conditions, which typically requires significant time and effort to generate, especially in grains with complex shapes. In addition, the training data generation relies on the theoretical contact models used. For example, the definition of contact geometry features, e.g. the overlap distance and the direction of contact norm, are empirical and can vary in different contact models. This disparity further limits the development of ML-based contact models.

On the other hand, when employing GNNs or CNNs to predict the rollouts of particle systems, error accumulation becomes unavoidable. This occurs because the dynamic update at the current time step depends on the particle positions from the previous step. Furthermore, apart from the position and translation velocity, particle rotation, and angular velocity are freedoms of individual particle/grain of granular materials. It seems that current kinematic features-based ML models are unable to reflect these fundamental physics. Additionally, it should be noted that geometry information-based networks, such as GNNs, are not well-suited for systems in which the number of particles changes during the simulation. This limitation arises from the inherent structure of graph-based networks, which require a fixed number of nodes to maintain consistency throughout the simulation.

In addition to the aforementioned discussion, several promising avenues for future research are also proposed in this section, aiming to address current limitations and enhancing existing methodologies:

(1) Enhancing training data for ML-based contact models: The critical challenge in developing one universal ML-based contact model is the quality and completeness of the training data. Consequently, future research could focus on developing new methodologies to define contact features beyond traditional empirical models, thereby minimizing inherent assumptions in the training datasets and improving the overall data quality. Additionally, efforts could be directed toward generating more comprehensive and diverse training datasets to enhance the generalization capability of ML-based contact models across a wider range of contact scenarios.
(2) Mitigating error accumulation in particle system rollouts: Error accumulation is a key limitation when utilising geometry information-based networks, particularly in long-term simulations of particle systems. Therefore, future works could explore error correction mechanisms, such as active learning, to dynamically adjust the model prediction during simulations to eliminate error propagation.
(3) Incorporating rotational dynamics into ML-based kinematic feature models: Current ML-based models for grain-level kinematic feature simulations focus primarily on positional and translational velocities, ignoring rotational dynamics. Future work could focus on integrating rotational degrees of freedom, such as particle rotation and angular velocity, into ML models, which may involve extending current CNN or GNN architectures to account for these additional physical factors.

4 The ML-Based Constitutive Models of Granular Materials

The development of the ML-based constitutive model of granular materials can be traced back to the 1990 s [60, 124, 143], and the development of the ML-based constitutive model of granular materials is undeniably one of the most prominent subjects in ML-assist numerical methods. The construction of ML surrogate stress–strain models for grain media depends on numerous factors, such as the hyperparameters, optimization algorithm, loss function, etc., used in neural networks [190]. However, the two fundamental factors that govern the performance of the ML-based constitutive model are the data resource and the feature selection of input–output corresponding to the used networks.

Table 3 The development of ML-based constitutive models for granular materials

Material	Reference	Experiment	Loading	Drained	Data source	Neural network
Sand	[143]	Triaxial	M	D	Experiment	MLP
Sand	[60]	Triaxial	M	D+U	Experiment	MLP
Soil	[195]	Triaxial	M	D	Experiment	MLP
Sand	[124]	Triaxial	M	D	Experiment	MLP
–	[17]	/	C	/	Synthetic data, and experiment	MLP
Coarse sand	[136]	Triaxial	M		Experiment	RNN
–	[16]	/	C	/	Synthetic data	MLP
Lateritic gravel	[70]	Triaxial	M	D	Experiment	MLP
Sand	[13]	Triaxial	M	U	Experiment	MLP
Ballast	[142]	Triaxial	M	D	Experiment	MLP
–	[55]	Triaxial	M	U	Synthetic data (MCC)	MLP
Sand	[73]	Triaxial	M	D	Experiment	MLP
Soil	[74]	Triaxial	M	U	Experiment	MLP
Lateritic gravel	[84]	Triaxial	M	D	Experiment	MLP
Soil	[114]	Triaxial	M	U	Experiment	MLP
Sand	[141]	Direct shear	M	/	Experiment	MLP
Rockfill	[7]	Triaxial	M	D	Experiment	MLP
Sand	[134]	Triaxial	M	D	Experiment	MLP
–	[146]	Triaxial	M	D	Synthetic data (HS)	MLP
Sand	[88]	Triaxial	M	D	Experiment	MLP
–	[100]	/	M	/	Synthetic data (DEM)	MLP
–	[161]	Simple shear	C	/	Synthetic data (DEM)	LSTM
/	[160]	Tension-shear	C	/	Synthetic data (DEM)	GRU
–	[188]	/	M	/	Synthetic data (MCC)	LSTM
–	[192]	Triaxial	C	D + U	Synthetic data (EM)	LSTM
–	[128]	Triaxial	C	/	Synthetic data (DEM)	GRU
–	[163]	Triaxial	C	/	Synthetic data (DEM)	TCNN
–	[115]	Triaxial	M	/	Synthetic data (DEM)	LSTM

4.1 The Data Sources

Table 3 offers a summary of partial works on ML-based constitutive models of granular materials over the past decades, there are mainly two types of data resources used when developing the ML constitutive models of granular materials. One is the experiment data, and the other is synthetic data.

4.1.1 The Experiment Data

The experiment data, which directly reflects the stress–strain response of granular materials, implicitly encapsulates the most authentic constitutive laws without any assumption, and thus the ML models developed from the experiment data can reveal the most essential mechanical features of granular materials. As listed in Table 3, the mechanical response of different granular materials, such as soil, sand, clay, ballast, and rockfill, has been investigated by different neural networks, where most ML models focus on the mechanical behaviour of granular materials under the drained and undrained triaxial test, and the remaining research dedicates to develop the ML surrogate models which can represent the mechanical features of granular media under direct shearing [141], simple shearing [161], and tension-shear [160].

While the experiment data can provide reliable inputs for neural networks to extract underlying principles governing the behaviour of materials, the limitations of the experiment data should also be taken into consideration. The training of neural networks normally requires a sufficient amount of data samples, making a purely experimental data-driven approach expensive. Additionally, restricted by the experimental facility, most experiment data used for training ML models are generated under specific shearing or triaxial compression conditions, covering only a partial range of stress–strain space and material types, and thus the robustness of trained ML models is limited.

4.1.2 The Synthetic Data

Compared to experiment data, synthetic data can be a cost-effective alternative to experimental data to develop ML-based constitutive models for granular materials, as experimental constraints do not bind it and can span a wider range of stress–strain space. As demonstrated in Table 3, the synthetic data generally can be acquired in two approaches. The first is the phenomenological constitutive models, such as the critical state-based models [180, 183] and the deviatoric hardening model [123, 127], and the other is the particle-based numerical techniques, such as the discrete element method (DEM).

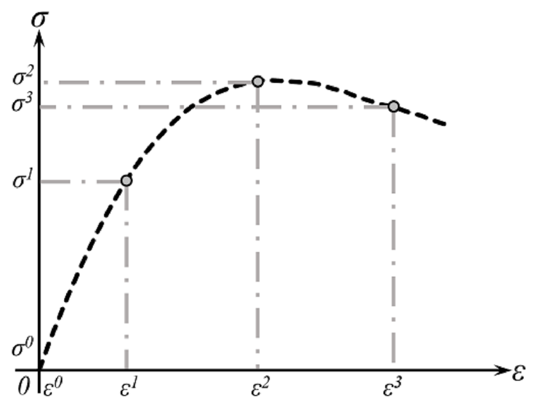
Given these advantages, it is possible to generate extensive amounts of synthetic data encompassing various materials and more extensive stress–strain space to establish more robust machine learning models. Ma et al. [115] provides an example of this, where one ML model which is capable of simulating the stress–strain response of granular materials with different particle size distributions (PSDs) and initial void ratio (e_0) under random loading paths was obtained through DEM-generated data. In addition, the development of the synthetic data-based ML model can also provide prior knowledge for constructing experiment data-based machine learning models. In reference [16], several mapping methods based on synthetic data were compared before selecting the true sequential dynamic mapping method for simulating the cyclic behaviour of soils with experiment data.

While ML models derived from synthetic data can capture the fundamental mechanical characteristics of granular materials under specific loading paths, they are limited in uncovering deeper constitutive laws, since synthetic data are generated under some assumptions (e.g. the homogenization in theoretical models) and simplification (e.g. the simplified shape of grain in DEM), which results in the loss of some intrinsic physical information of granular materials. However, there is no doubt that synthetic data could be a cost-effective supplement to experimental data in the development of ML-based constitutive models.

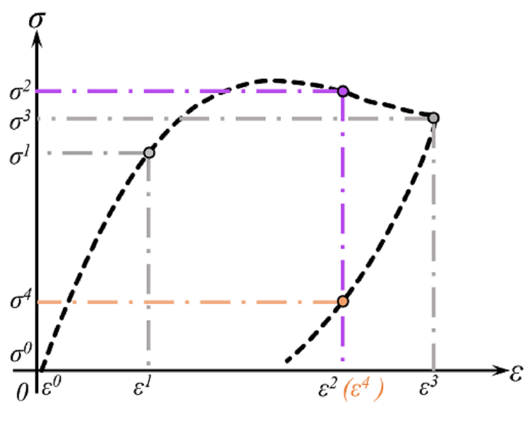
4.2 The Training Strategy for Different ML-Based Constitutive Models

4.2.1 The History-Dependent Features of ML-Based Constitutive Models

Under the quasi-static condition, as shown in Fig. 10a, the relationship between the strain tensor $\boldsymbol{\varepsilon}^{(t)}$ and stress tensor



(a) The monotonous loading case.



(b) The reversal loading case

Fig. 10 Different loading paths in 1D plastic deformation

$\boldsymbol{\sigma}^{(t)}$ of each time step in the monotonous loading cases keeps the one-to-one mapping which can be pursued as:

$$\boldsymbol{\sigma}^{(t)} = \boldsymbol{f}\left(\boldsymbol{\varepsilon}^{(t)}, \mathbf{W}, \mathbf{b}\right) \tag{16}$$

However, when subjected to loading reversal, as demonstrated in Fig. 10b, a reversal loading may result in two identical strains (e.g. $\varepsilon^2 = \varepsilon^4$) corresponding to different stress states, and therefore, it is necessary to introduce the history variables to differentiate the loading state (i.e. the unloading and reloading).

Different from the traditional theoretical constitutive models, in which loading states can be explicitly formulated, ML models depend on other methods to differentiate the loading history under cycling loading conditions. Time-sequence neural networks draw history information from their input data consisting of discrete strain–stress data sequences:

$$\hat{\boldsymbol{\sigma}}^{(t)} = \boldsymbol{f}^{ML}\left(\left\{\boldsymbol{\varepsilon}^{(t-n)}, \ldots, \boldsymbol{\varepsilon}^{(t-2)}, \boldsymbol{\varepsilon}^{(t-1)}, \boldsymbol{\varepsilon}^{(t)}\right\}, \mathbf{W}, \mathbf{b}\right) \tag{17}$$

For instance, RNNs learn loading history by recording the hidden state encoded in the input data with RNN neurons, while TCNN identifies different loading states by the causal convolution calculation with trainable filters.

However, single-step-based neural networks like the MLP cannot distinguish the loading history itself. Therefore, they need to introduce artificially added history/internal variables as extra input to transform the one-to-many mapping into a surjective between the input and output variables, which can be expressed as:

$$\hat{\boldsymbol{\sigma}}^{(t)} = f^{\mathrm{MLP}}\left(\boldsymbol{\varepsilon}^{(t)}, \varphi^{(t)}, \mathbf{W}, \mathbf{b}\right) \tag{18}$$

where $\boldsymbol{\sigma}^{(t)}$ and $\boldsymbol{\varepsilon}^{(t)}$ represent the current stress and strain states of granular materials, $\boldsymbol{\varphi}^{(t)}$ means the history variables and consists of $\{\varphi_1^{(t)}, \ldots, \varphi_m^{(t)}\}$; $\mathbf{W}$ and $\mathbf{b}$ mean trainable parameters of the ML model.

4.2.2 The Selection of Input Features in ML Models

In addition to the data resource, selecting the necessary information from the obtained data as the input and output features to train the neural network is also a key aspect in the development of ML models. Generally, the input variable of the neural network is normally a combination of the following three types of parameters: 1) the material parameters, such as the PSDs, and the relative density D_r of the granular assembly, representing the basic intrinsic properties of materials; 2) the state parameters, e.g. the void ratio e, the mean effective stress p, the deviatoric stress q, the stress $\boldsymbol{\sigma}$, and strain $\boldsymbol{\varepsilon}$ at the current time step t, which govern the evolution of the stress–strain relationship; 3) the history parameters which is used to differentiate loading states.

The variables used as the input and output of the data-driven model vary according to specific problems. As listed in Table 3, ML models are constructed with single-step-based or time-sequence neural networks to simulate the behaviour of granular materials with cycling or monotonous loading data. Thanks to their specific architectures, time-sequence neural networks like the LSTM, GRU, and TCNN inherently acquire historical information from their input data, which comprises a sequence of discrete data [115, 128, 163]. Thus their input variables always consist of material and state parameters whether in monotonous or cycling loading

In single-step-based networks such as MLP, to consider the influence of the loading history on the constitutive relationship, there are generally two approaches to selecting history variables. The first one is regarding the predicted state parameters of the last time step as the history variables to predict the current state, such as works [16, 73, 143, 146]. As illustrated in Fig. 11a, the current prediction stress $\hat{\boldsymbol{\sigma}}^{(t)}$ not only relies on the current state variable strain $\boldsymbol{\varepsilon}^{(t)}$ but also the state variable of the strain $\boldsymbol{\varepsilon}^{(t-1)}$ and the predicted stress $\hat{\boldsymbol{\sigma}}^{(t)}$ at the $(t-1)^{th}$ time step. The process can be formulated as:

$$\hat{\boldsymbol{\sigma}}^{(t)} = f^{\mathrm{ML}}\left(\lambda, g(\boldsymbol{\varepsilon}^{(t)}, \boldsymbol{\sigma}^{(t)}), \varphi\left\{\boldsymbol{\varepsilon}^{(t-1)}, \hat{\boldsymbol{\sigma}}^{(t-1)}\right\}, \mathbf{W}, \mathbf{b}\right) \tag{19}$$

where λ represents the vector consisting of material parameters. However, one inevitable issue is that any predicted error of the previous stress state by neural network can lead to error accumulation in the ML system.

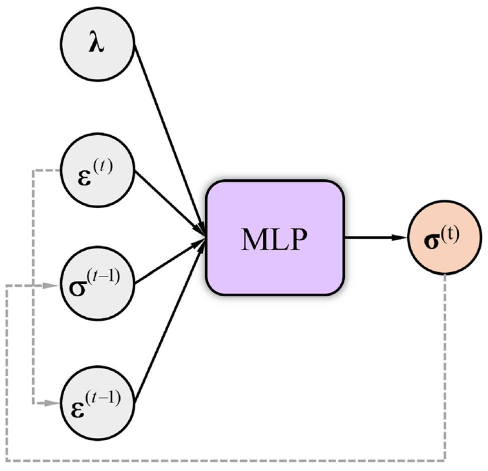
As demonstrated in Fig. 11b, an alternative scheme is directly extracting history variables $\boldsymbol{\varphi}^{(\mathrm{t})}$ from the state variable of the current step, such as using the absolute accumulated strain increment $\Delta\boldsymbol{\varepsilon}^{(\mathrm{t})}$ of the current time step as the history variable [66, 80], which can be computed as:

$$\boldsymbol{\varphi}^{(t)} = \begin{cases} \Delta\boldsymbol{\varepsilon}^{(t)} = 0 & t = 0 \\ \Delta\boldsymbol{\varepsilon}^{(t)} = \sum_{k=1}^{t} \left|\boldsymbol{\varepsilon}^{(k)} - \boldsymbol{\varepsilon}^{(k-1)}\right|, & t \geq 1 \end{cases} \tag{20}$$

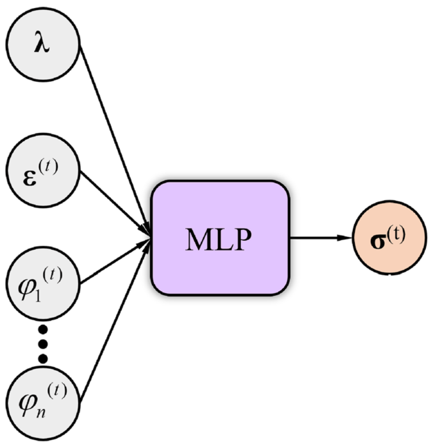
where $\boldsymbol{\varepsilon}_{\mathrm{ij}}^{(\mathrm{t})}$ represents the strain component at the current time step of t.

4.3 Examples of the ML-Based Stress–Strain Models Derived from DEM Data

This section provides some intuitive demonstration of the performance of different neural networks in predicting the stress–strain response of granular materials. Four different loading paths: 1) the conventional triaxial compression (CTC), 2) the isobaric axisymmetric triaxial loading, i.e., constant-p (CP), 3) the intermediate principal stress coefficient of tri-axial loading, i.e., constant-b (CB), and 4) random strain loading, are leveraged to generate the



(a) Regarding state variables as history variables (Hashash et al., 2004).



(b) Regarding parameterized variables as history variables (Huang et al., 2020)

Fig. 11 Different two methods for the selection of history variables in the MLP

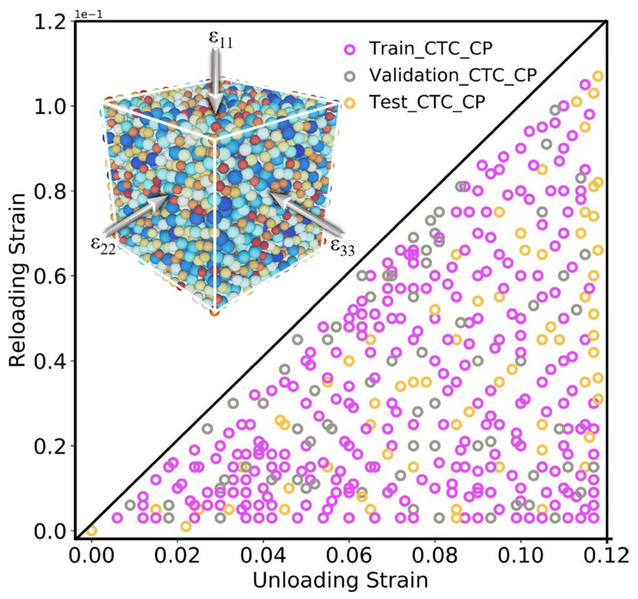


Fig. 12 The stress–strain space of cases with one-cycle CTC and CP loading path

training data with the same RVE, where $p = \frac{\sigma_1+\sigma_2+\sigma_3}{3}$ and $b = \frac{\sigma_2-\sigma_3}{\sigma_1-\sigma_3}$. The relevant work [163] shows that time-sequential neural networks, including GRU, LSTM, and TCNN have similar performance in predicting the constitutive laws of granular materials. Therefore, only one of the time-sequential neural networks (GRU) and the single-step-based MLP are trained and tested with the generated data.

4.3.1 The Stress–Strain Space of the Training Data

Both the CTC and CP cases include one-cycle loading. As shown in Fig. 12, a set of unloading-reloading points are adopted based on the principle that the unloading strain is bigger than the reloading strain to control the position of the hysteresis loop. In CB loading cases, only the monotonous loading condition is involved and the *b*-value is set to be a different value, increasing from 0 to 1 with an interval of 0.05, to control the loading paths.

Eventually, there are 440 loading cases generated under the CTC, CP, and CB loading conditions. Wherein 360 loading curves, involving these three loading conditions are randomly selected as the training and validation sets; the left 80 cases comprise the test set.

4.3.2 The Prediction Results

As mentioned in Sect. 4.2.2, there are normally two types of history variables available for the MLP to distinguish from different loading states: 1) one is the predicted state variables from the last time step, and 2) the other is the parameterized variables from the current state variables. To demonstrate the performance of different history variables in MLP, two MLPs are trained and tested with the same data but different history variables in this section. Wherein the first MLP employs the strain $\boldsymbol{\varepsilon}^{(t-1)}$ and predicted stress tensor $\hat{\boldsymbol{\sigma}}^{(t-1)}$ at the last time step as the history variables to predict the current stress [72], and the other utilizes the absolute accumulated strain increment $\Delta\boldsymbol{\varepsilon}_{\mathrm{ij}}^{(\mathrm{t})}$ as the history variables [80], which can be respectively formulated as:

$$\hat{\boldsymbol{\sigma}}^{(t)} = f^{\mathrm{MLP1}}\left(\boldsymbol{\varepsilon}^{(t)}, \left\{\boldsymbol{\varepsilon}^{(t-1)}, \hat{\boldsymbol{\sigma}}^{(t-1)}\right\}, \mathbf{W}, \mathbf{b}\right) \tag{21}$$

$$\hat{\boldsymbol{\sigma}}^{(t)} = f^{\mathrm{MLP2}}\left(\boldsymbol{\varepsilon}^{(t)}, \Delta\boldsymbol{\varepsilon}^{(t)}, \mathbf{W}, \mathbf{b}\right) \tag{22}$$

Furthermore, one of the time-sequence neural networks (i.e. GRU) is also trained and tested with the same data. As a time-sequential network, the history-loading information is encapsulated in the input strain sequence $\left\{\varepsilon^{(\mathrm{t-n+1})}, \varepsilon^{(\mathrm{t-n+2})}, \ldots, \varepsilon^{(\mathrm{t})}\right\}$ of GRU and can be identified by its RNN cells. Therefore, there is no need for the assistance of the extra history variable to learn the constitutive laws of granular materials, which can be expressed as:

$$\hat{\boldsymbol{\sigma}}^{(t)} = \boldsymbol{f}^{\mathrm{GRU}}\left(\left\{\boldsymbol{\varepsilon}^{(t-n+1)}, \boldsymbol{\varepsilon}^{(t-n+2)}, \ldots, \boldsymbol{\varepsilon}^{(t)}\right\}, \mathbf{W}, \mathbf{b}\right) \tag{23}$$

where n represents the time step of the input strain sequence. The hyperparameters used in GRU and MLP are listed in Table 4.

Figure 13 gives an intuitive demonstration of the performance of the trained MLP1 by six representative prediction cases. In CTC and CP loading cases, in which one reversal loading is included, although the MLP1 can capture the constitutive laws of granular materials in partial cases (CTC case 2), an error accumulation phenomenon as mentioned in Sect. 4.2.2 occurs, especially in reloading stage, see CTC case 1, CP case 1, and CP case 2. Furthermore, it is also found that uncontrollable accumulated errors can eventually result in the breakdown of the entire prediction system (see CP case 2), while controllable accumulated errors can be gradually diminished (see CP case 1) due to the relationship between the strain and stress changes from the one-to-many to a surjective after the reversal loading. Additionally, the trained MLP1 performs well in all CB cases, as the relationship between stress and strain remains strictly surjective in these cases, and thus the selection of the history variables has less influence on the performance of the MLP1.

The identical cases in Fig. 13 are also employed to test the trained MLP2 and GRU. The obtained results are showcased in Fig. 14, and the predicted average mean absolute error (MAE) of these two models for the 80 test cases is listed in Table 5. The outcomes, as illustrated in Fig. 14 and detailed in Table 5, reveal the MLP can achieve comparable performance to the GRU using the parameterized history variables. Furthermore, a comparison between the prediction results in Fig. 14 and their counterparts in Fig. 13 indicates that the use of parameterized history variables effectively mitigates the problem of error accumulation.

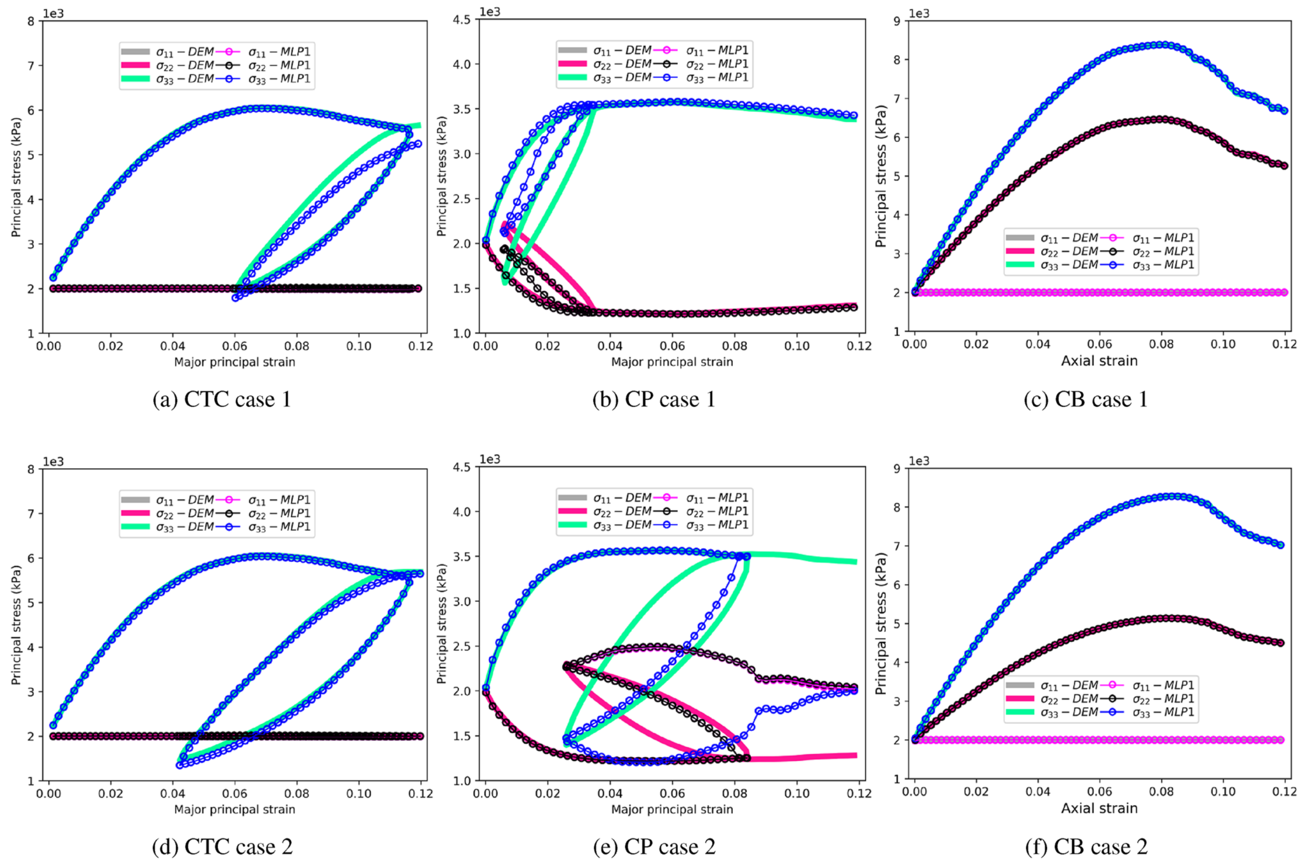
From the perspective of engineering application, the strain intervals are normally different and random. Therefore, the sensitivity analysis of different neural networks to the loading strain interval is also conducted in this work. The strain intervals of 80 test cases are randomized, and the trained MLP2 and GRU are tested again with the newly generated test data. The prediction results of the same cases in Fig. 13 and Fig. 14 are plotted in Fig. 15. The results illustrate that the single-step-based MLP remains effective in capturing the stress–strain response despite variations in the loading interval. In contrast, the time-sequential neural network GRU exhibits notable sensitivity to changes in the loading interval, resulting in poorer performance under such conditions.

4.4 Summary

This section provides an overview of ML-based constitutive models for granular materials, focusing on two key aspects: the sources of training data and the training strategies employed by different neural networks, and a concise summary of the advantages and limitations of each aspect is provided in Table 6. In addition, Sect. 4.3 presents a detailed case study that illustrates the predictive capabilities of

Table 4 Hyperparameters used in GRU and MLP

ML models	Model architecture	The number of **W**, **b**	Learning rate	Batch size	Time steps of input	Active function	Optimizer	epoch	Time (s/epoch)
GRU	3 GRU Layers /50 Units + 1 Dense layer /50 Units + 1 Dense layer /3 Units	39,003	0.001	128	40	Relu	Adam	1000	24
MLP	9 Dense Layers /20 Units + 1 Dense layer /3 Units	22703	0.001	128	–	Relu	Adam	1000	4



(a) CTC case 1 (b) CP case 1 (c) CB case 1

(d) CTC case 2 (e) CP case 2 (f) CB case 2

Fig. 13 The prediction results of MLP1

Table 5 The prediction results of all test cases with the GRU and MLP

Loading paths	ML models	Average MAE
CB	GRU	0.0140
	MLP2	0.0159
CP	GRU	0.0174
	MLP2	0.0183
CTC	GRU	0.0168
	MLP2	0.0185

various neural networks under different loading conditions. The following conclusions can be drawn from the analysis:

The datasets used to train ML-based constitutive models can be categorized into two types: experimental data and synthetic data. Experimental data are normally generated under conditions that closely resemble real-world loading scenarios. However, these datasets are typically costly and time-consuming to collect, resulting in limited coverage of the sampling space. Additionally, experimental data often contain noise introduced by measurement devices, which can adversely affect the training of neural networks.

In contrast, the data obtained from existing phenomenon-based constitutive models are more cost-effective, enabling the generation of diverse and extensive datasets that cover a wide range of loading scenarios. This diversity can help produce more general neural network models. However, their fidelity fully depends on the selected constitutive models, which may have restrictions in reproducing certain characteristics of granular materials. To the best knowledge of the authors, no constitutive model is perfect. For the data obtained from DEM simulations, reliability is also a problem that we need to pay attention to. Apart from the accuracy of the method itself, the ad-hoc/arbitrary selection of microscopic parameters for the grain/particle is primarily calibrated based on certain cases, which may be not applicable to a generic problem.

On the other hand, the constitutive law of granular materials is inherently state and history-dependent, and both single-step and time-sequence (multi-step) neural networks can be employed to predict such constitutive behavior. Single-step-based neural networks, such as MLPs, offer the advantage of a simple architecture, which typically leads to high training and prediction efficiency. However, these models lack the inherent ability to capture

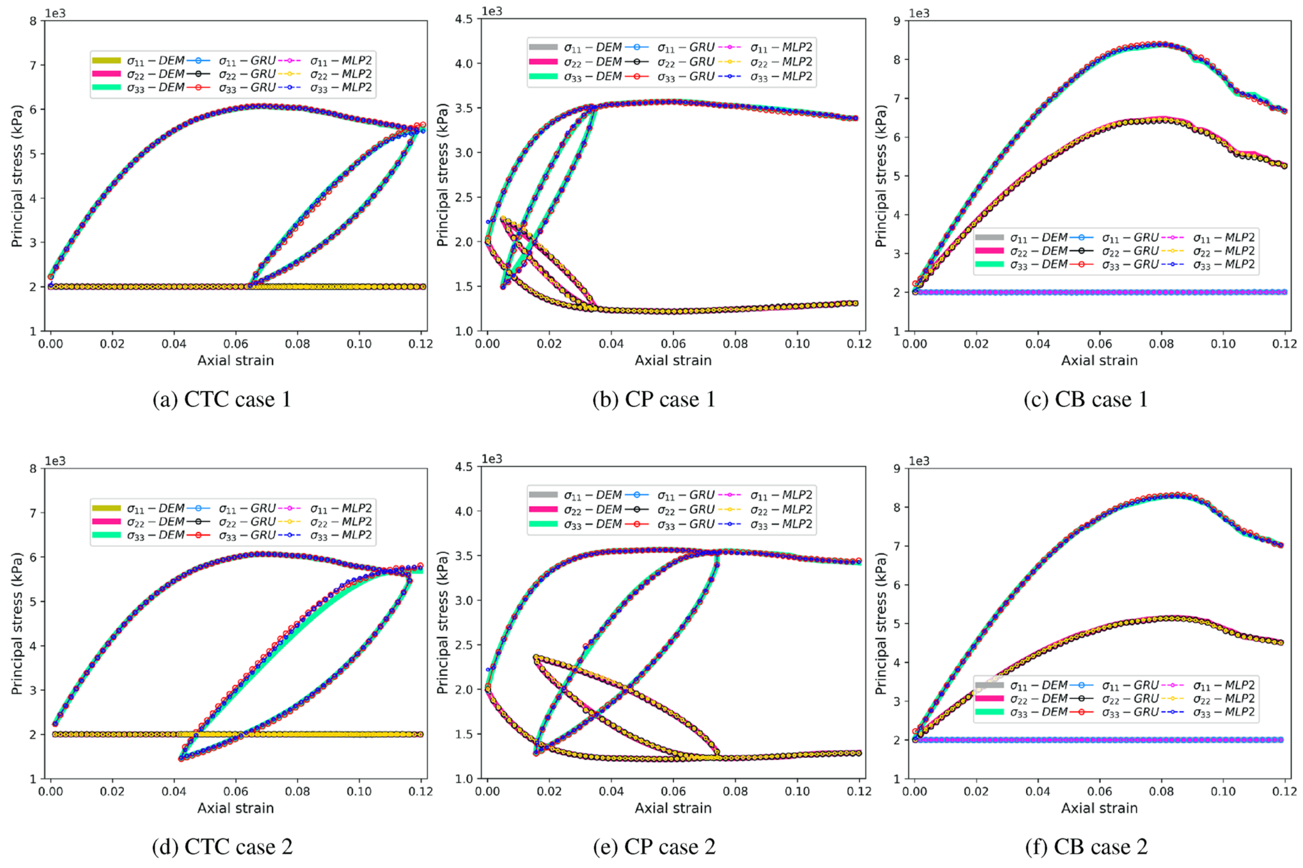


Fig. 14 The prediction results of GRU and MLP2

loading history, necessitating the inclusion of artificially added internal or history variables. There are generally two approaches for incorporating history variables in MLPs: (a) state parameters predicted at the previous time step, which can lead to an accumulation of errors over time, and (b) variables extracted directly from the current state parameters, which aim to provide more immediate historical context without the propagation of previous errors.

Due to the unique architecture, time-sequence neural networks are inherently capable of distinguishing different loading histories without requiring additional assistance. This architecture enhances their ability to capture long-term historical dependencies and perform complex non-linear mappings. However, the increased complexity of these networks, compared to single-step-based models, results in more training parameters. Consequently, time-sequence neural networks are more challenging to tune and exhibit lower training and prediction efficiency. Additionally, according to the prediction result showcased in Sect. 4.3, it is also found that time-sequential networks are sensitive to the change of strain intervals, while the single-step-based network still performs well under the same loading cases, indicating that time-sequential networks are more suitable to be applied in the data with a fixed loading step.

Based on the discussion of ML-based constitutive models of granular materials, some promising avenues for future research, particularly from the perspectives of transfer learning and optimization for training strategy, could be potential ways to solve the challenges mentioned above:

(1) Application of transfer learning in ML-based constitutive Modeling: Given the high cost and limited availability of experimental data, transfer learning offers a promising approach to enhance the generalization capabilities of ML-based constitutive models. By leveraging knowledge from pre-trained models on synthetic datasets, transfer learning can fine-tune these models using smaller amounts of high-fidelity experimental data, which has the potential to improve the accuracy and authenticity of ML-based constitutive models across a broader range of loading conditions.
(2) Optimization of training strategies: Single-step-based neural networks are inherently limited in their ability to capture loading history, necessitating the development of parameterization schemes for internal vari-

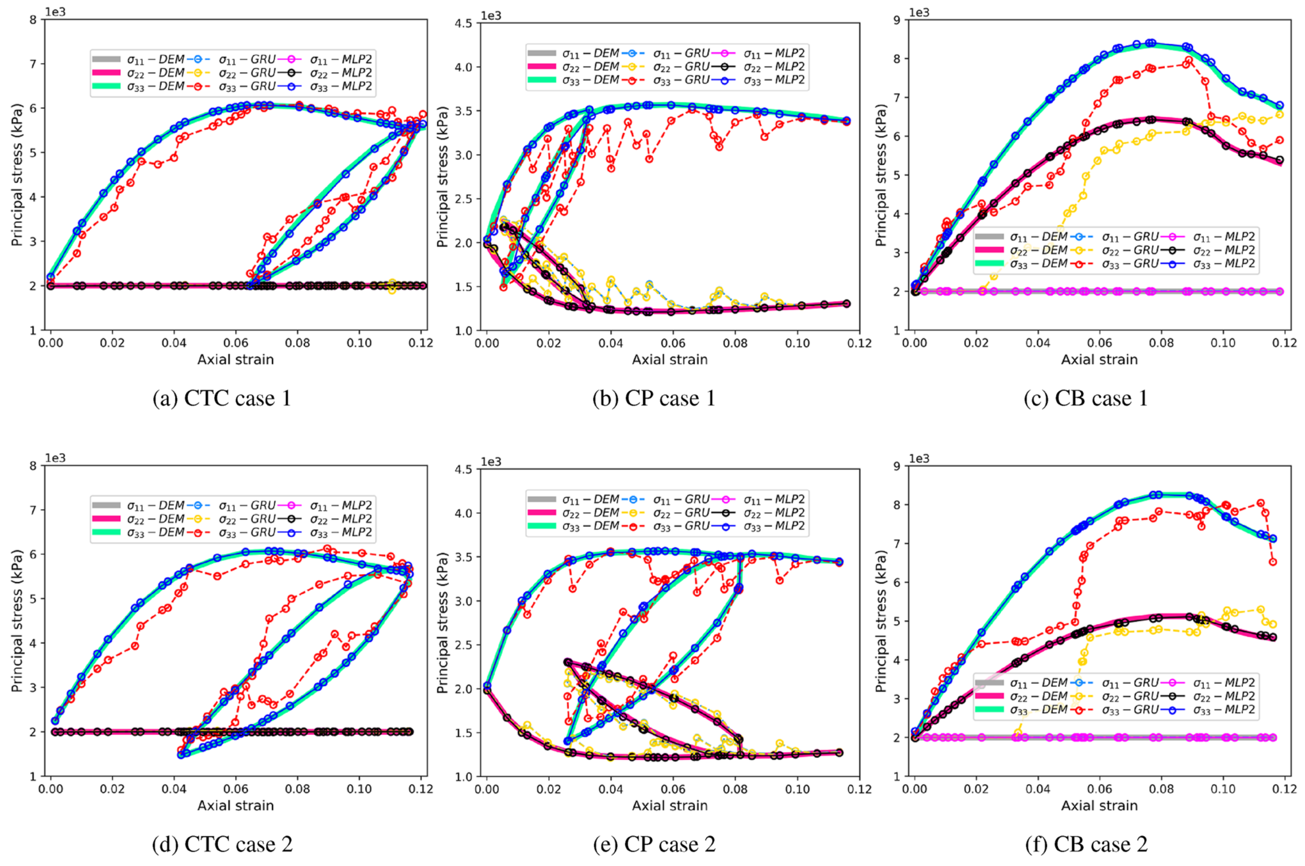


Fig. 15 The prediction results of GRU and MLP2 in cases with different loading intervals

Table 6 A concise summary of the ML-based constitutive models for granular materials

ML-based constitutive models	Types	Advantages	Disadvantages
Data sources used	Experiment data	(1) Reflecting the most authentic and realistic constitutive laws of granular materials; (2) Representative of real-world loading conditions	(1) High cost and time-consuming; (2) Incorporating noise; (3) Incomplete datasets
	Synthetic data	(1) Easily generated in large quantities; (2) Wide coverage of different loading scenarios	(1) Smearing out some intrinsic features of granular materials; (2) Lack of authenticity of experimental data
Neural network type	Single-step-based	(1) Simple architecture; (2) High training and prediction efficiency; (3) Robust to the change of loading step	(1) Requirement for artificially added internal/history variables; (2) Error accumulation problem

	Multi-step-based	(1) No extra history variables are required in identifying the loading history;
		(2) Stronger non-linear mapping capability than single-step-based networks
			(1) Complex structure;
			(2) Lower training efficiency;
			(3) Sensitive to the change of the loading interval

ables which aim to represent loading history using the fewest physically meaningful variables possible. Additionally, future research could focus on improving the performance of time-sequence neural networks under varying loading intervals. Approaches such as interpolation or data augmentation may serve as effective strategies to enhance the resilience of these networks to non-uniform loading intervals, thereby broadening their applicability in real-world scenarios.

5 The ML-Aided Macroscopic Simulation of Granular Materials

Apart from the use in depicting the microscopic interparticle contact characteristics and corresponding contact forces, ML models have also been integrated with continuum-based particle methods or mesh-based simulation approaches to predict the macroscale deformation of granular systems. According to the coupled numerical methods, the ML-aided macroscopic simulation can be categorized into two types: 1) ML simulation based on the interpolation of macroscopic particle kinematic features, and 2) the macroscopic simulation using ML models as constitutive models (e.g. the MPM/FEM-ML framework) in this review.

5.1 The Macroscopic Particle Kinematic Features-Based ML Simulation

Over the past decades, various continuum-based particle methods, such as the particle finite element method (PFEM) [122], element-free Galerkin method (EFGM) [19], smoothed particle hydrodynamics (SPH), and material point method (MPM), have been developed to model the deformation features of either fluid or solid. Among these methods, the SPH and MPM stand out as particularly representative and have been extensively utilized across a broad range of engineering applications. Therefore, our primary focus in this section centres on reviewing the integration of data-driven methods with SPH and MPM to depict the macroscopic deformation of granular materials. Before that, a concise introduction to SPH and MPM will be given.

5.1.1 The Continuum-Based Particle Numerical Methods

In both MPM and SPH, the kinematic features of macroscopic material points or SPH particles govern the macroscale deformation of the material under consideration. The standard MPM was originally proposed in the literature [148] and is a particle-based Eulerian-Lagrangian method that has gained increased attention in geomechanics [15, 52, 144], especially in addressing large-strain problems of granular materials [20, 21, 170].

As shown in Fig. 16, the computational process of the MPM consists of four steps: 1) In the ***particle-to-node*** stage, the continuum body Ω is discretized into a total of n_p material points located on the background mesh with a total of n_n nodes. The physical variables encapsulated in material points at the time step t, e.g. mass m_p, velocity $\mathbf{v}_p^t$, and acceleration $\mathbf{a}_p^t$, are projected to the corresponding nodes via the shape function $N_i(\mathbf{x})(i = 1, 2, 3, ..., n_n)$ to compute nodal mass $m_i^t = \sum_{p=1}^{n^p} N_i\left(\mathbf{x}_p^t\right)m_p$, nodal momentum $\mathbf{q}_i^t = \sum_{p=1}^{n^p} m_p\mathbf{v}_p^t N_i\left(\mathbf{x}_p^t\right)$, nodal force $\mathbf{f}_i^t = \sum_{p=1}^{n^p} m_p\mathbf{a}_p^t N_i\left(\mathbf{x}_p^t\right)$

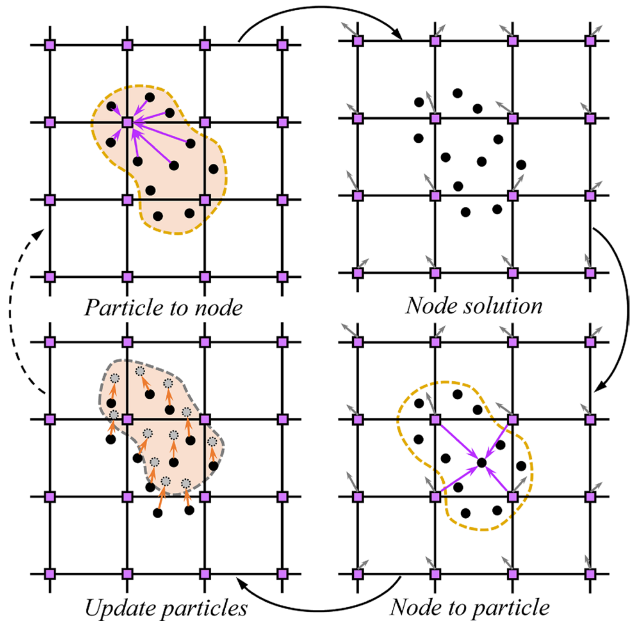


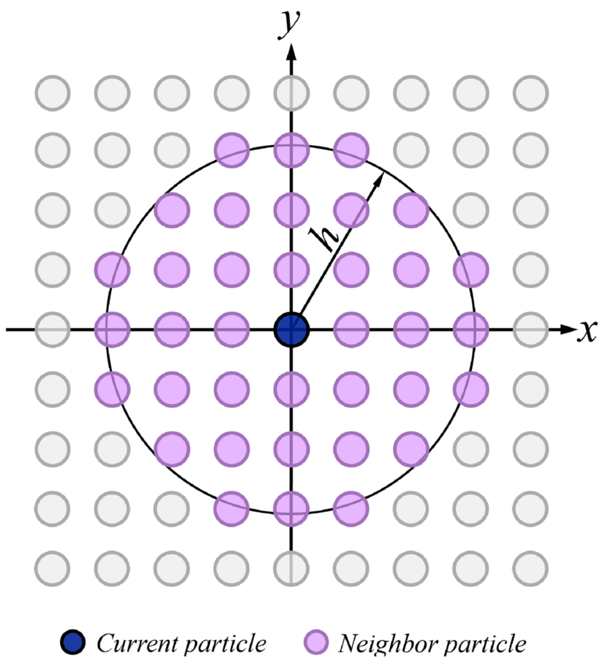
Fig. 16 The calculation process of the MPM

as well as nodal velocity $\mathbf{v}_i^t = \mathbf{q}_i^t/m_i^t$ and acceleration $\mathbf{a}_i^t = \mathbf{f}_i^t/m_i^t$ at the time step of t. Where $\mathbf{x}_p^t$ represents the location of the p^{th} material point at the t time step. Then, 2) the nodal displacement $\mathbf{u}_\mathrm{i}^{t+1}$, velocity $\mathbf{v}_\mathrm{i}^{t+1}$, and $\mathbf{a}_\mathrm{i}^{t+1}$ at the following time step are solved with the governing equation in the ***node solution*** step, which is similar to the updated Lagrangian FEM process [12, 174]. 3) the obtained solution is interpolated back to the material points in the ***node-to-particle*** stage and 4) the dynamic information of material points including their position $\mathbf{x}_p^{t+1}$, velocity $\mathbf{v}_p^{t+1}$, and acceleration $\mathbf{a}^{t+1}$, is calculated and the Lagrangian mesh is reset for the next iteration in the ***update particles*** step.

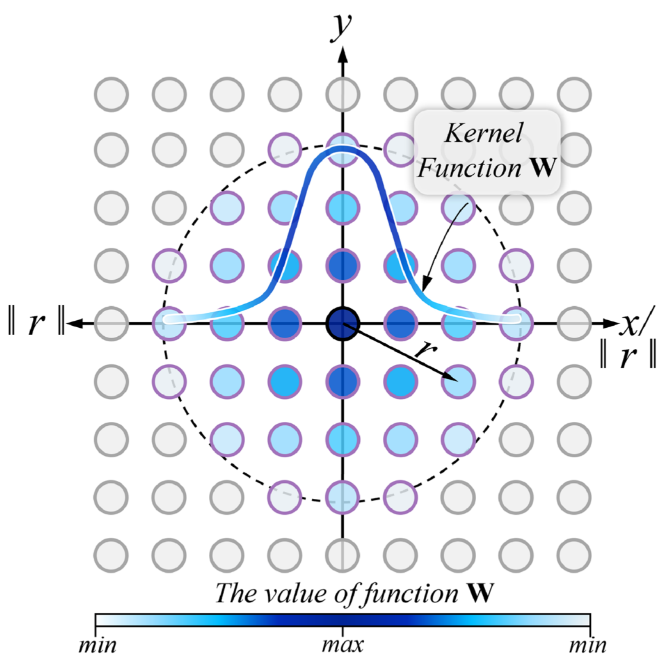
SPH, initially developed for astrophysical problems [62, 113], is another commonly used mesh-free Lagrangian method and has also been used to model the deformation of granular materials [24–26, 159]. Similar to the MPM, as demonstrated in Fig. 17a, the SPH method discretizes the research domain into a set of macroscopic interpolation particles to represent physical properties $\mathrm{A}\left(\mathbf{x}_\mathrm{i}^t\right)$ of specific point $\mathbf{x}_\mathrm{i}^t$ at the current time step t, and each physical variable, e.g. density $\rho\left(\mathbf{x}_i^t\right)$, velocity $\mathbf{v}\left(\mathbf{x}_i^t\right)$, and pressure $\mathbf{p}\left(\mathbf{x}_i^t\right)$, is impacted by its neighbour particles $\mathbf{x}_j^t$ within an interaction radius (h). By using the smooth function [108] (also referred to as kernel function) as depicted in Fig. 17b, field variables at any given point can be estimated by interpolation calculations:

$$\mathbf{A}\left(\mathbf{x}_\mathrm{i}^\mathrm{t}\right) = \sum_j \mathbf{A}\left(\mathbf{x}_\mathrm{j}^\mathrm{t}\right)\frac{m_j}{\rho_j}\mathbf{W}\left(\mathbf{x}_\mathrm{i}^\mathrm{t} - \mathbf{x}_\mathrm{j}^\mathrm{t}, h\right) \tag{24}$$

where m_j, ρ_j represent the mass and density of the j^{th} neighbour particle, and $\|r\| = \left\|\mathbf{x}_\mathbf{i}^t - \mathbf{x}_j^t\right\|$. Then, the dynamic infor-



(a) The discretized system with an interaction radius h.



(b) The weight contribution following the kernel function $\mathbf{W}$

Fig. 17 2D SPH particles

mation of each SPH particle can be iteratively computed by forces (e.g. body forces, viscosity forces, etc) acting on them.

Relying on macroscopic particle kinematic features to govern the deformation of the target domain makes MPM and SPH effectively avoid mesh entanglement and distortion, showcasing notable superiority in handling large-strain issues. However, updating the dynamic features for each particle is computationally significant, particularly in simulations involving a substantial number of particles.

5.1.2 The Framework of Macroscopic Kinematic Features-Based ML Models

The advancement of data-driven simulators in the past two decades [145] offers one promising way to solve the above-mentioned problem in MPM and SPH methods. Given one specific position, ML-based simulators can output necessary dynamic features simultaneously, making them far superior in computational efficiency [75] bypassing the complex physical solution process.

In MPM and SPH methods, the system consists of an unordered set of macroscopic interpolation particles including rich interaction relations with one another. These particle features and particle-particle interaction can be naturally expressed with the graph that can be modelled by the graph network simulators (GNS) but not convenient to be transferred into the regular sequences or vectors which single-step, time-sequential networks [169], and specific tree networks [91] can extract knowledge from.

GNS operates on graphs to learn the physics of the dynamic system and predict rollouts. The graph network spans the system domain with nodes representing a collection of particles and the links connecting the nodes representing the local interaction between the material points. As shown in Fig. 18, GNS $\left(S_\theta\right)$ consists of two parts: one is the learnable approximator d_Θ, and the other is the updater. The current physical state of each particle in MPM or SPH simulations is represented as a set of $\mathbf{x}_\mathrm{i}^\mathrm{t} \in \mathbf{X}^\mathrm{t}$ in S_θ. Where $\mathbf{x}_\mathbf{i}^\mathrm{t}$ is a vector, normally consisting of position $\mathbf{p}_\mathrm{i}^\mathrm{t}$, velocity $\dot{\mathbf{p}}_\mathbf{i}^\mathrm{t}$, boundary information $\mathbf{b}_\mathrm{i}^\mathrm{t}$, type $\mathbf{f}_\mathrm{i}$ (i.e. deformable or rigid) of each material or SPH point, which can be expressed as:

$$\mathbf{x}_\mathrm{i}^\mathrm{t} = \left[\mathbf{p}_\mathrm{i}, \left\{\dot{\mathbf{p}}_\mathrm{i}^{\mathrm{t-n+1}}, \dot{\mathbf{p}}_\mathrm{i}^{\mathrm{t-n+2}}, \dots, \dot{\mathbf{p}}_\mathrm{i}^\mathrm{t}\right\}, \mathbf{b}_\mathrm{i}^\mathrm{t}, \mathbf{f}_\mathrm{i}\right] \tag{25}$$

where n represents the previous time step. Instead of the physical solving process, the approximator d_Θ takes the current state of the particle system $\mathbf{X}^t$ as input to predict its dynamic feature $\mathbf{Y}^t$. The updater then computes the physical state $\mathbf{X}^{t+1}$ at the next time step with the $\mathbf{X}^t$ and predicted $\mathbf{Y}^t$.

The approximator d_Θ comprises three parts, including *Encoder*, *Processor*, and *Decoder* as demonstrated in Fig. 18. In the prediction process of d_Θ, the *Encoder* is responsible for embedding $\mathbf{X}^t$ into the initial latent graph $G_0\left(\mathbf{V}_0, \mathbf{E}_0\right)$ by assigning the node to each particle (i.e. node encoder δ_θ^v) and adding edges $\mathbf{r}_{\mathrm{i,j}}^t$ between particles within

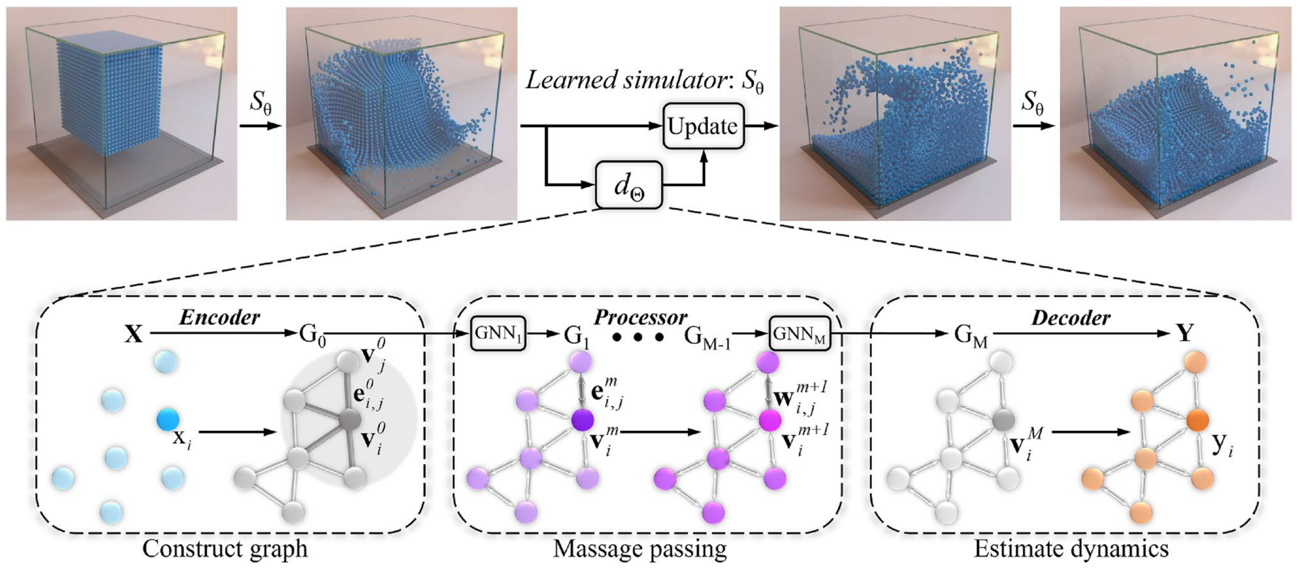


Fig. 18 The graph network simulators [139]

one interaction radius R (i.e. edge encoder δ_θ^e), which is formulated as:

$$\mathbf{v}_i^t = \delta_\Theta^v\left(\mathbf{x}_i^t\right) \tag{26}$$

$$\mathbf{e}_{i,j}^t = \delta_\Theta^v\left(\mathbf{r}_{i,j}^t\right) \tag{27}$$

where $\mathbf{v}_i^t \in \mathbf{V}_0$ represents the node state in the graph and $\mathbf{e}_{i,j}^t \in \mathbf{E}_0$ means the pair-wise relationship between nodes in the graph. Then, the *Processor*, consisting of M GNN layers, performs an M times message-passing process on G_0. Finally, the Decoder outputs the dynamic features of the system $\mathbf{y}_i^t \in \mathbf{Y}^t$ with the final graph G_M from the *Processor* to update the velocity $\dot{\mathbf{p}}_i^{t+1}$ and position $\mathbf{p}_i^{t+1}$ of each node by:

$$\dot{\mathbf{p}}_i^{t+1} = \dot{\mathbf{p}}_i^t + \mathbf{y}_i^t \Delta t \tag{28}$$

$$\mathbf{p}_i^{t+1} = \mathbf{p}_i^t + \dot{\mathbf{p}}_i^{t+1} \Delta t \tag{29}$$

So far, the application of GNS in particle-based numerical methods (e.g. SPH and MPM) is still in the nascent stage. For example, ML models trained with the data generated from MPM simulations have been used to effectively predict the macroscale deformation of granular systems. Gonzalez et al. [139] developed a graph network-based simulator (GNS) via the 'encode-process-decode' scheme to model the dynamic features of physical systems of different materials, including sand and goop, by the data generated from both the MPM and SPH simulations. Aoyama et al. [6] constructed one GNN with the data generated from the MPM simulator to predict the stacked shape of the grain collection within different containers. Choi et al. [32] introduced physics-inspired inductive biases, such as an inertial frame that allows learning algorithms to prioritize one solution (constant gravitational acceleration) over another, reducing learning time. The GNS implementation uses semi-implicit Euler integration to update the next state based on the predicted accelerations. The GNS was trained on 26 CB-Geo MPM simulations of granular flow trajectories with 2500 particles each with 400 timesteps. At each time step, a graph network is created by linking all the material points within an interaction radius of 0.03 m, which results in connecting each node to 128 neighbouring particles. GNS uses the velocities from the previous five steps to predict the acceleration at the next time step. Figure 19 shows the GNS rollout prediction compared to MPM simulations. GNS successfully replicates the granular flow within a 5% error in the trajectory estimation for a condition outside the training regime. [33]

5.2 The Macroscopic Modelling with ML Models as Constitutive Models

In addition to the works mentioned above, ML models can also feature the mechanical response of the material at each Gaussian integral point of finite elements in mesh-based numerical methods, such as the FEM. In the following, we will focus on the combination of ML with the most popular FEM and related FEM-DEM multiscale technique, because the idea and workflow of the combination of other methods with ML are similar. In the framework of FEM, the ML model can substitute the phenomenological model or the representative volume element (RVE) embedded in each

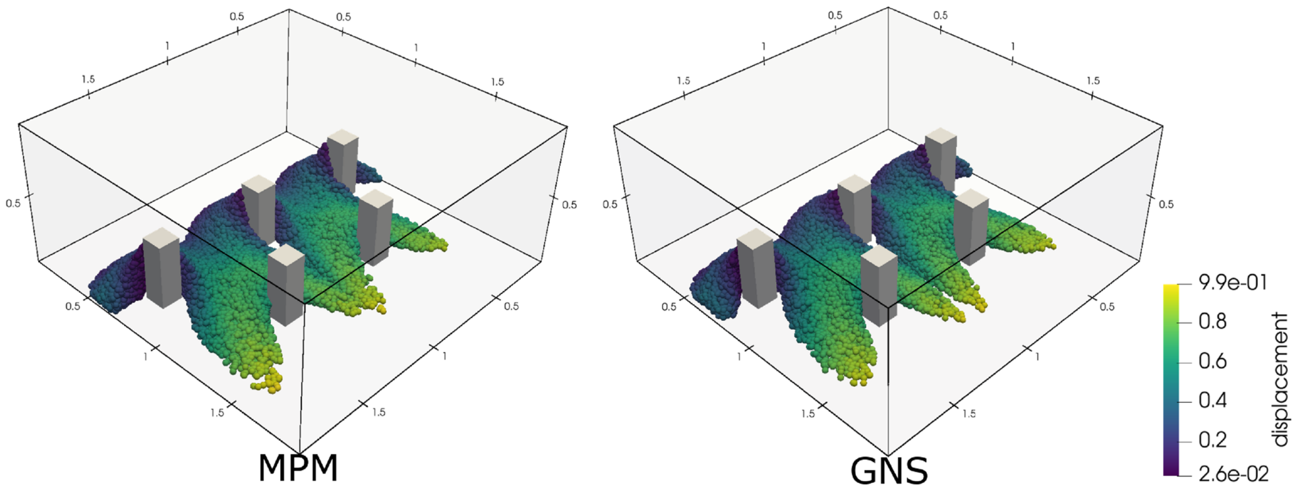


Fig. 19 Comparison of granular flow interaction with a barrier between MPM and GNS. The colour represents the magnitude of displacement. The GNS prediction is outside the training regime

Gaussian point of the FEM to provide the stress–strain response when solving boundary value problems (BVPs). In the following subsections, the basic framework of the FEM and FEM-DEM method is first introduced concisely, and then the relevant work about the development of the FEM-ML framework will be discussed in detail.

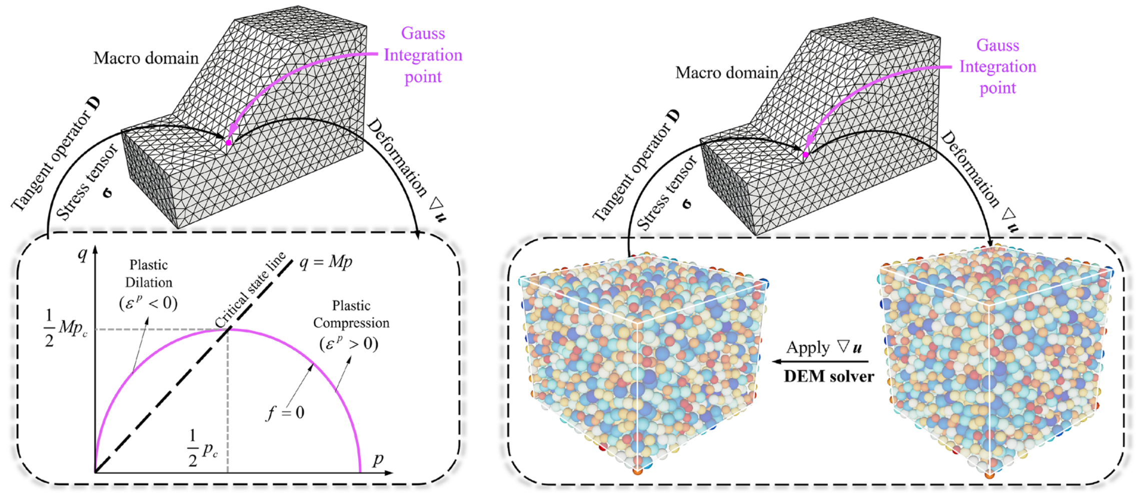
5.2.1 The Framework of the FEM and the Coupled FEM-DEM

In the FEM, the whole macro domain Ω is first discretized into a finite number of subdomains (elements) and the phenomenological model (e.g. critical state-based model shown in Fig. 20a) is embedded into Gaussian points of each FE element to describe the local stress–strain relationship. Without considering the body force, the macroscopic deformation of the geometry domain can be obtained by solving the nodal information of each FE element by the governing equation:

$$\int_{\Omega} \mathbf{B}^T \boldsymbol{\sigma}_t \, d\Omega = \int_{\Gamma} \mathbf{N}^{\mathbf{T}} \mathbf{t} \, d\Gamma = \mathbf{F}_t \tag{30}$$

where $\boldsymbol{\sigma}_t$ and $\mathbf{F}_t$ represent the stress tensor and external force of the current load step; $\mathbf{t}$ is the boundary traction imposed on the Neumann boundary Γ of the Ω; $\mathbf{N}$ represents the shape function; $\mathbf{B} = \mathbf{LN}$, and $\mathbf{L}$ is the differential operator.

In each loading step, as shown in Fig. 20a, the finite element algorithm fulfil the governing formulation via the



(a) The FEM with critical state-based constitutive model. (b) The FEM-DEM multiscale method

Fig. 20 The basic framework of FEM and FEM-DEM

following steps: 1) The gradient of nodal displacement $\bigtriangledown \mathbf{u}_t$ is calculated and passed into the constitutive model of each Gaussian point to output the nodal stress $\boldsymbol{\sigma}_t$ and tangential operator $\mathbf{D}_t$. 2) Given the $\mathbf{D}_t$ (also referred to as the constitutive matrix), $\boldsymbol{\sigma}_t$ and boundary condition, the displacement increment $\Delta\mathbf{u}_t$ is calculated. 3) The nodal displacement is updated according to the obtained $\Delta\mathbf{u}_t$. However, granular materials normally perform high plasticity. Therefore, the Newton–Raphson method is leveraged to iteratively solve the $\Delta\mathbf{u}_t$, ensuring the residual force $\mathbf{R}_t$ of the system remains minimized, which can be expressed as:

$$\mathbf{R}_t = \int_\Gamma \mathbf{N}^\mathbf{T}\mathbf{t}\, d\Gamma - \int_\Omega \mathbf{B}^T \boldsymbol{\sigma}_{(t,m-1)}\, d\Omega = \mathbf{K}_t \Delta\mathbf{u}_{(t,m)} \tag{31}$$

where $\Delta\mathbf{u}_{(t,m)}$ represents the solved displacement increment at the m^{th} iteration of the current load step, and the global stiffness matrix $\mathbf{K}_t$ is assembled by:

$$\mathbf{K}_\mathrm{t} = \int_\Omega \mathbf{B}^\mathbf{T}\mathbf{D}_t\mathbf{B}dV \tag{32}$$

The iteration is repeatedly executed until the $\mathbf{R}_t$ converges to one tiny value.

As shown in Fig. 20b, in the coupled FEM-DEM algorithm, the local constitutive relationship is provided by the representative volume element (RVE) solved by the DEM. The nodal displacement is calculated by the FEM solver and imposed on each RVE as their boundary conditions (i.e. $\bigtriangledown \mathbf{u}_t$). Then, with received boundary conditions, the stress $\boldsymbol{\sigma}_t$ and tangential operator $\mathbf{D}_t$ of each RVE is calculated based on various homogenization methods (e.g. Voigt's hypothesis [89], Reuss's hypothesis [27, 181]). Finally, the solved $\boldsymbol{\sigma}_t$ and $\mathbf{D}_t$ by the DEM are returned to the FEM solver to calculate the displacement increment of each nodal to update the nodal displacement.

Although both the FEM and FEM-DEM can predict the macroscopic characteristics of granular materials, they have their respective limitations. The constitutive models used in the FEM are continuum theory-based, potentially obscuring the discrete nature of granular materials. Furthermore, models utilized in FEM often incorporate multiple variables that lack physical significance but require extensive calibration efforts. Moreover, constitutive models tend to be sophisticated to accommodate specific phenomena and are thus difficult to apply to new data cases. In addition, once the constitutive model is determined in FEM, it remains unadjustable, which restricts the further applicability of FEM to new tests.

Regarding the FEM-DEM techniques, although they can offset the deficiency of the FEM in capturing the discrete features of materials to some extent, it still suffers from the issue of excessive computational costs.

It is worth noting that to avoid the deficiency of FEM in simulating material large deformation, MPM-DEM is also proposed as a promising way for the investigation of large deformation of granular materials [103]. To overcome the computational issues of RVE encountered in large deformation, an adaptive RVE model was recently proposed [129, 164]. Similar to the FEM-DEM framework, the MPM-ML framework can also be developed.

5.2.2 The FEM-ML Framework

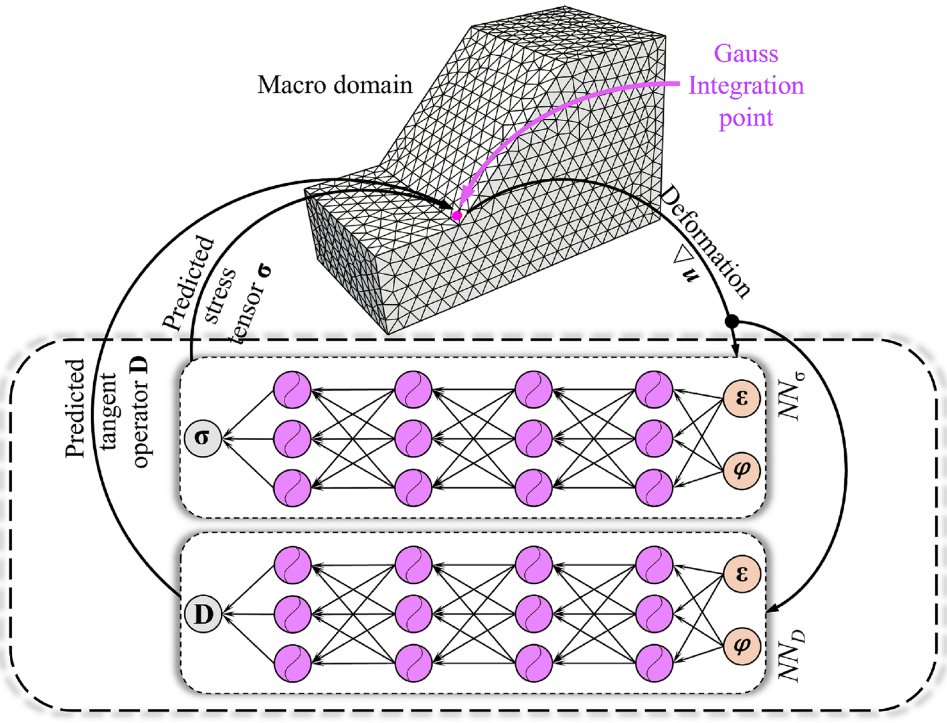
The ML model is featured with ongoing improvement and high computational efficiency in prediction. Thus it is a potential way to circumvent the challenges mentioned above of traditional numerical methods by integrating ML surrogate models into them. As illustrated in Fig. 21, in FEM or coupled FEM-DEM, replacing constitutive models or RVEs, the trained neural networks offer the necessary tangential operator or stress state according to the received local deformation information of each FE element to accelerate the computational process in BVPs.

Generally, according to the data used to develop the neural networks, the FEM-ML framework can be categorized into two different types: 1) one is the phenomenological model data-based FEM-ML framework, and 2) the other is the DEM data-based FEM-ML framework, which is demonstrated in the following sections in detail, respectively.

Over the past decades, numerous efforts have been made to represent the material response with ML models in BVPs. The earlier work of constitutive data-based FEM-ML can be traced back to the 1990 s, in which the trained ML model functions for the stress calculation in FEM to simulate the material macroscopic behaviour under triaxial [143], biaxial, uniaxial cycled compression [58], and un-uniform loading conditions [59]. Subsequently, numerous scholars have carried out related works. For instance, literature [72] develops one NN model as an alternative to the constitutive model to accelerate the efficient convergence of the Newton iteration in the FE algorithm; In reference [86], one rate-dependent NN constitutive model is developed to capture the time-dependent feature of geomaterials in the FEM; Xu et al. [178] compute the tangential matrix with a Cholesky-factored symmetric positive-definite neural network. Based on a dimensionality reduction method (Proper Orthogonal Decomposition), Huang et al. [80] described the intrinsic history-dependency feature of plastic materials with the feedforward neural network and states variables in FEM; Nikolaos N et al. [157] leveraged the deep learning network to replicate one elastoplastic model, where the plastic flow and yield surface of the model are accurately portrayed through a series of deep network predictions, to obtain the seamless FEM simulation.

Similarly, numerous endeavours have been given to the DEM data-based FEM-ML framework to tackle multiscale simulations. For example, Le et al. [96] utilized the neural

Fig. 21 The framework of the FEM-ML algorithm



network to approximate the effective potential, which is used to derive the stress and tangent elastic tensor of materials in the multiscale computation of non-linear elastic materials, to reduce the computational cost. The recurrent neural network is trained with DEM data and integrated into Gaussian points of the FE meshes in the works of Simons et al. [61] and Logarzo et al. [110] to improve the efficiency of the multiscale scheme; The performance of the Gaussian process and ANN as the agent in the multi-scale BVP was compared in the work of Fuhg et al. [56]; Replacing RVEs, Guan et al [66] utilized the feedforward neural network to predict the material matrix and local stress to complete the nonlinear iteration in the multiscale computation process; Qu et al. [130] developed an active learning-based data-driven modeling framework which can pritorise the most informative data for a trained model and make it be progressively improved via interactive model training and data labelling; Qu et al. [131] further developed a transfer learning strategy that can combine the use of well-established phenomenological models, numerical models and physical experiments for data-driven material modeling, thereby reducing the data demands for certain material modeling tasks; Replacing the DEM solver embedded in Gaussian point with one well-trained neural network, Rangel et al. [133] proposed a data-driven FVE-ML multiscale framework to explore the thermomechanical behavior of granular materials subjected to thermal expansion.

However, the development of the FEM-ML framework still confronts some problems, which will be discussed from the aspects of the training data and neural networks used. The completeness of the training sample, e.g. fixed time step and the limited stress–strain space of training data, deteriorate computational stability [110, 189] and the robustness of the FEM-ML framework. Furthermore, in the DEM data-based FEM-ML framework, there are no explicit history variables are calculated to feature the loading states in DEM solvers, which performs the history and state-dependent feature of the particle assembly by fabric tensor or energy-based parameters [44, 87]. Therefore, parameterizing history states into the macroscopic loading information (e.g. strain and stress) to develop the DEM data-based FEM-ML framework is also a tricky issue, especially when employing the single-step-based neural network to construct the FEM-ML framework. Although multi-timestep deep learning models can avoid this problem [65, 67], they are incompatible with the standard FEM computational framework, which operates on a single step.

5.3 An Example: The DEM Data-Based FEM-ML Modelling

A detailed study on time-sequential networks and constitutive data-based FEM-ML framework can be found in the literature [65]. Therefore, in this section, we offer an example of the development of an MLP-based FEM-ML framework with the DEM data where there are no explicit internal variables available via a 2D biaxial compression simulation.

5.3.1 The Process of Developing the FEM-ML Framework

To develop the FEM-ML framework, one biaxial compression FEM-DEM multiscale simulation is performed, and the stress $\boldsymbol{\sigma}^{(t)}$, strain $\boldsymbol{\varepsilon}^{(t)}$, internal variable $\boldsymbol{\varphi}^{(t)}$, as well as tangential operator $\mathbf{D}_t$ of each RVE, are recorded as the training data of ML models. Wherein the absolute accumulated strain increment $\Delta\boldsymbol{\varepsilon}^{(t)}$ is adopted as the history [80] according to the result in Sect. 4.3.2.

As illustrated in Fig. 22, during the multiscale modelling process, the bottom boundary of the coarse-meshed geometry is fixed in the vertical and horizontal directions, and both the left and right boundary of the mesh are confined with the constant horizontal pressure ($P_0 = 100$ KPa). A displacement-controlled axial loading is applied to the top surface of the rectangular domain with a fixed velocity (0.001 m/load step) until the axial strain arrives at 0.05 under the reversal loading 1(*Reld1*) path shown in Fig. 23. The parameters used in each RVE directly refer to the work of Guo et al [68]. The detailed grain-scale parameters are listed in Table 7.

Different from continuum-based phenomenological models, whose constitutive function is sufficiently smooth and can be obtained by automatic differentiation (AD) methods [1, 60, 61, 72, 86], the stress–strain curve of granular materials has dramatic disturbance, and thus the AD method cannot provide the proper tangential operator during the Newton–Raphson iteration in FEM. Alternatively, the method proposed in the work of Guan et al. [66] is resorted in this case, i.e. using the neural network to provide the material matrix. Consequently, two MLP models $\left(NN_\sigma \text{ and } NN_{\mathrm{D}}\right)$ are trained to provide the necessary stress $\boldsymbol{\sigma}^{(t)}$ and material matrix $\mathbf{D}_t$ for the FE algorithm, which can be expressed as:

$$\sigma^{(t)} = NN_\sigma\left(\boldsymbol{\varepsilon}^{(t)}, \Delta\boldsymbol{\varepsilon}^{(t)}, \mathbf{W}, \mathbf{b}\right) \tag{33}$$

$$\mathrm{D}_t = NN_D\left(\boldsymbol{\varepsilon}^{(t)}, \Delta\boldsymbol{\varepsilon}^{(t)}, \mathbf{W}, \mathbf{b}\right) \tag{34}$$

Table 7 The detailed grain-scale parameters of each RVE in the FEM-DEM scheme

Parameter	Value
Particle number	450
Particle size range (mm)	(6 ~12)
Young's modulus (MPa)	600
Density (kg/m^3)	2650
Particle frictional coefficient	0.5
Damping ratio	0.1

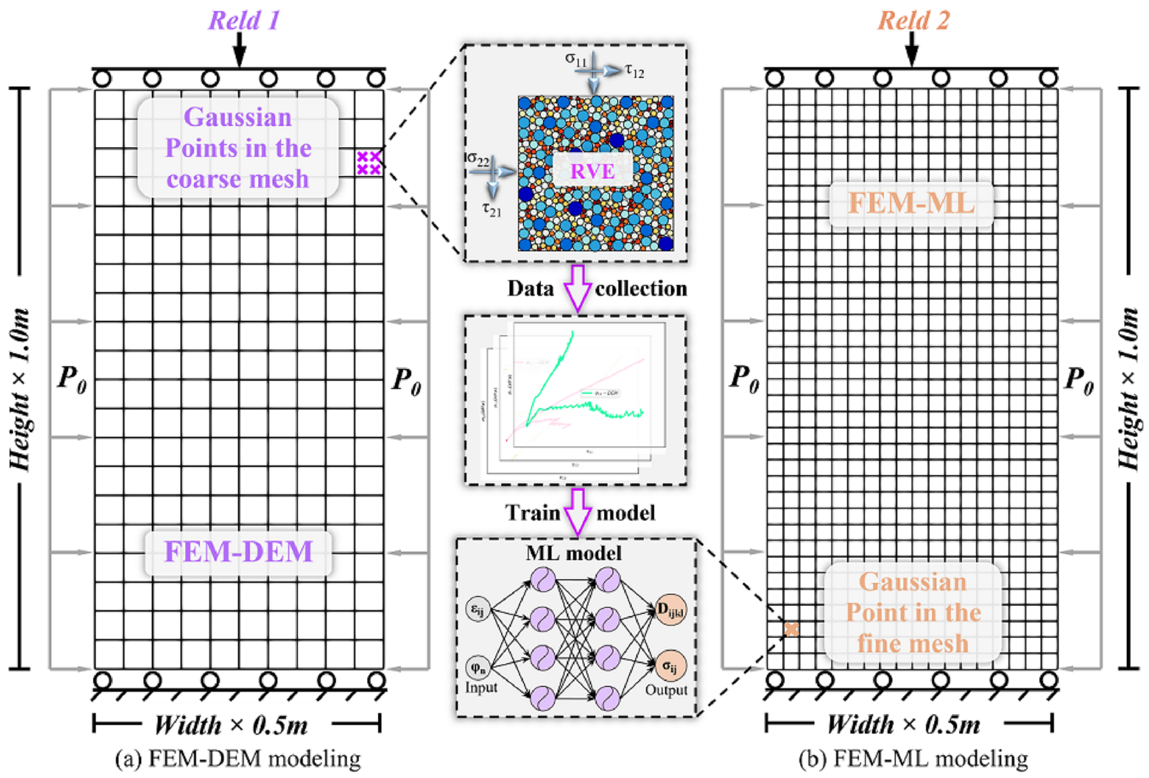


Fig. 22 An illustration for the DEM data-based FEM-ML framework

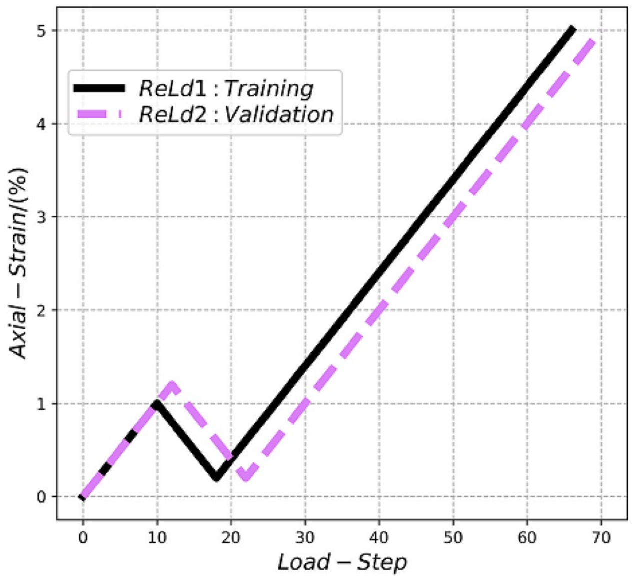


Fig. 23 Two different reversal loading paths

As demonstrated in Fig. 22, the trained networks are embedded into each Gaussian point of the fine mesh to construct the FEM-ML framework, which is then leveraged to solve the same BVP of biaxial compression but under the *Reld2* path showcased in Fig. 23.

5.3.2 The Prediction Results with the Developed FEM-ML Scheme

As one controlled experiment, an additional FEM-DEM simulation is conducted with identical boundary conditions and *Reld2* path within the finely-meshed sample to validate the performance of the developed FEM-ML scheme. Therefore, a total of three biaxial compression simulations are implemented. The detailed experimental parameters of each simulation are listed in Table 8. The solved shear strain and axial displacement fields by both the FEM-ML approach and the corresponding FEM-DEM scheme are compared and demonstrated in Fig. 24 and Fig. 25.

Figure 24 demonstrates the solved axial displacement fields and top force (TF) of the fine-meshed sample by both the FEM-ML and FEM-DEM approaches. Where the axial displacement at some special loading moments (i.e. the 2 nd , 22 nd, and 70 th load steps) are respectively plotted in Figs. 24a, 24b, and 24c; Fig. 24d illustrates the change of the top force of the sample during the simulation process. Similarly, the shear strain fields at the 2 nd , 22 nd, and 70 th load steps along with the volume strain (VS) of the geometry domain in the FEM-DEM and FEM-ML modelling are showcased in Fig. 25.

The results showcased in Fig. 24-25 demonstrate that the FEM-ML scheme can still accurately model the deformation feature of granular materials in the same BVP, although the mesh density is increased and the loading path is changed, which proves that the well-developed FEM-ML approach possesses the extrapolation capability to some extent.

5.4 Summary

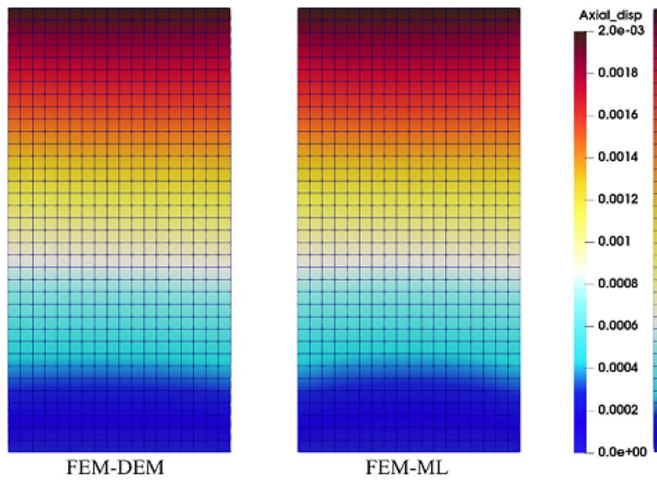
This section focuses on recent developments in coupling ML techniques with continuum-based numerical methods, which are categorized into two main areas: the integration of ML models in 1) meshless methods, such as SPH and the MPM, and 2) mesh-based methods, including the FEM and multiscale FEM-DEM frameworks.

Table 9 presents a concise summary of the advantages and limitations of each aspect. In the context of continuum-based numerical methods, the extensive spatial interactions among interpolation particles underscore the suitability of geometry information-based neural networks. These networks facilitate permutation-invariant learning, which enables them to efficiently capture complex spatial relationships among particles and bypass complicated physical equations. Consequently, compared to traditional continuum-based particle methods, the geometry information network-based simulator significantly reduces the computational complexity when simulating the deformation of granular materials, leading to a remarkable improvement in overall simulation efficiency.

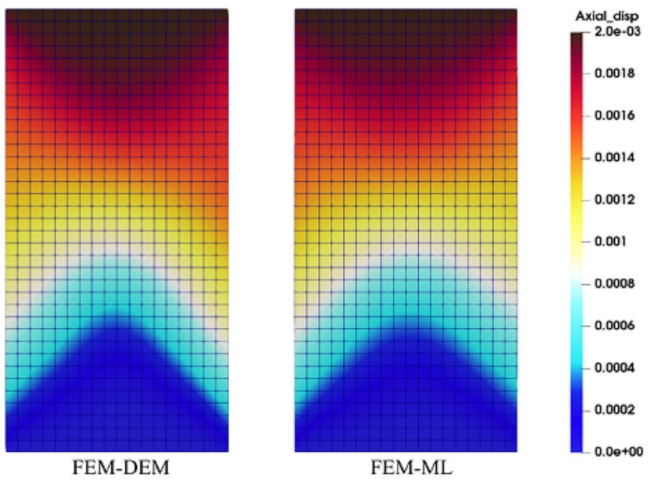
On the other hand, compared to the traditional FEM, especially the FEM-DEM approach, the superior computational efficiency of the ML-based constitutive model mentioned in Sect. 4.4 can significantly reduce the computational cost and time of the FEM-ML multiscale method. Additionally, due to the adaptive nature of ML surrogate models, the performance of the FEM-ML scheme can be continuously

Table 8 The experimental parameters of the three biaxial compression modelling

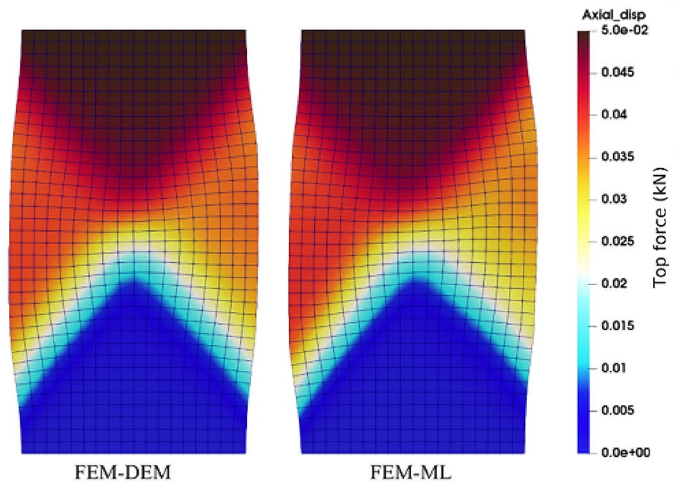
No.	Mesh	Method	Loading path	Data for using	Total load steps	Time consuming	Unloading step	Reloading step	The number of RVEs
1	Coarse	FEM-DEM	Reld1	Training	66	2.12 h	10	18	800
2	Fine	FEM-DEM	Reld2	Validation	70	9.65 h	12	22	2592
3	Fine	FEM-ML	Reld2	Testing	70	0.016 h	12	22	2592



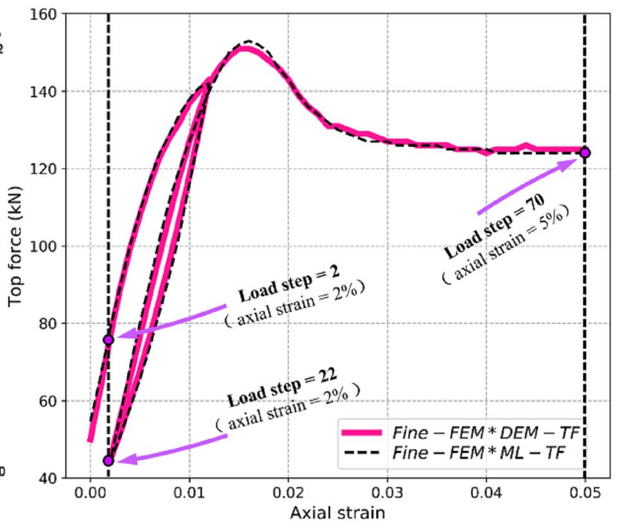
(a) Load step=2



(b) Load step=22



(c) Load step=70



(d) The top force

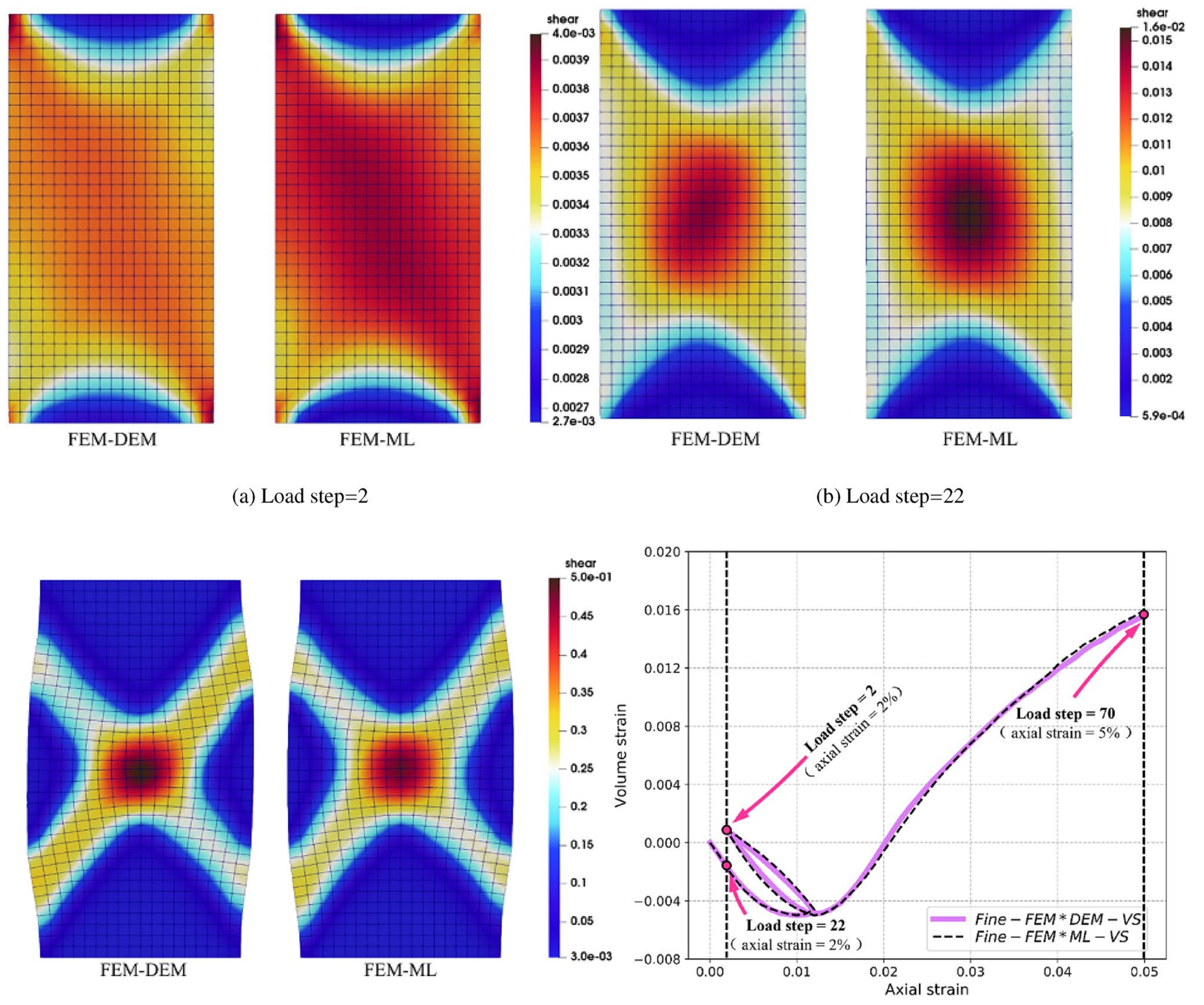
Fig. 24 The solved axial displacement field under the *Reld*2 path in fine meshes [162]

enhanced given sufficient training data, enabling it to handle increasingly complex tasks more effectively.

However, it is important to note that training geometry information-based networks typically requires significant computational memory due to the large number of particles in a single simulation case. Furthermore, when using Graph Neural Networks (GNNs) to predict system rollouts over extended periods, error accumulation becomes a challenge. This issue arises because the dynamic information at each time step is updated based on the particle positions from the preceding time step, leading to the propagation of errors throughout the simulation.

In the development of the FEM-ML scheme, both single-step and multi-step surrogate models serve as alternatives to phenomenological models or RVEs embedded at Gaussian points within the FEM framework. While time-sequence neural networks can naturally capture the history-dependent constitutive behavior of granular materials, they are sensitive to changes in loading intervals, and their multi-step nature is not aligned with the conventional FEM algorithms. This discrepancy requires additional efforts to integrate time-sequence networks into a FEM solver. In contrast, single-step neural networks are inherently compatible with the FEM algorithms and robust to variations in loading intervals. However, they still face the challenge of identifying appropriate internal variables to capture different loading histories adequately.

Additionally, the generalization capability of current FEM-ML strategies is often limited by the completeness of training data. For instance, the simulation result obtained in



(a) Load step=2

(b) Load step=22

(c) Load step=70

(d) The top force

Fig. 25 The solved shear strain field under the *Reld*2 path in fine meshes [162]

Sect. 5.3 demonstrates the developed FEM-ML approach has a certain extrapolation capability. Although the mesh density of the solving domain increases and a different but similar loading path is applied, the FEM-ML scheme can still acquire a comparable solution to the corresponding FEM-DEM simulation. However, when encountering a new situation where the geometry domain and boundary conditions vary considerably, the performance of the current FEM-ML solver will deteriorate [162]. This limitation makes many FEM-ML approaches case-specific and restricts their applicability to more general scenarios.

Based on the summary presented above, several promising directions for future research in ML-aided macroscopic simulation methods can be identified. These avenues include employing active learning to identify informative data for training trustworthy ML models, Bayesian or Gaussian processes for uncertainty quantification, and the development of hybrid computational frameworks that integrate neural networks with traditional numerical methods, which are detailed as follows:

1). Application of active learning to mitigate error accumulation: One of the key challenges in using ML-based simulators, particularly GNS, is the accumulation of errors during long-term system rollouts. Active learning could be employed to dynamically select new data points that the model finds most uncertain, enabling iterative retraining of the model to correct errors. This approach would improve the robustness of ML models over extended simulations and ensure more reliable results, especially in complex, multi-step scenarios.

Table 9 The overview of advantages and limitations of ML-aided macroscopic simulation methods

ML-aided macroscopic simulation	Types	Advantages	Disadvantages
Geometry information network-based ML simulator	GNS	(1) Strong generalization capability across meshless numerical methods; (2) Low computational complexity; (3) High computational efficiency	(1) Requirement for large training memory; (2) Error accumulation problem; (3) Weaker suitability to systems where interpolation points are added or removed
Data-driven FEM-ML multiscale methods	Phenomenological model data-based DEM data-based	(1) Higher computational efficiency (2) Potential for ongoing enhancement	(1) Requirement for artificially added internal variables; (2) Limited generalization ability

2). Incorporating Bayesian or Gaussian processes for reliability assessment: To improve the reliability of ML-based constitutive models, future research could incorporate Bayesian networks or Gaussian processes, which provide probabilistic estimates along with predictions. These approaches enable uncertainty quantification in model outputs, allowing researchers to assess the confidence in predictions. Incorporating such probabilistic models would enhance the trustworthiness of simulations by identifying areas where predictions may be less reliable and require further retraining efforts.

3). Development of hybrid computational schemes: Utilizing the insights from Bayesian or Gaussian processes, future research could focus on developing hybrid computational schemes that integrate the strengths of ML models with traditional numerical methods. In such frameworks, ML models could be applied in regions where their predictions have the highest certainty, while traditional methods could be used in areas where ML models exhibit lower reliability. This approach would improve computational efficiency while maintaining high accuracy, offering a robust solution for complex simulations with high efficiency and precision.

6 Discussion

The weak extrapolation capability of the ML models is one primary reason that limits their further application in modelling granular materials. Research efforts have been focused on developing advanced techniques that specifically address the relevant issue, such as the proposal of Physics-informed neural networks (PINNs) and Fourier neural operators(FNOs).

The idea behind PINNs [132] is to enforce the neural network to satisfy some known physical laws or governing equations by introducing a loss function that incorporates these physical constraints. The total loss function for a PINN is typically a combination of a data loss term $L_{\text{data}}(\theta)$ and a physics loss term $L_{\text{physics}}(\theta)$, which can be expressed as: $L(\theta) = L_{\text{data}}(\theta) + \lambda L_{\text{physics}}(\theta)$. Here, λ is a hyperparameter that balances the importance of data fidelity and physics consistency. θ is a set of parameters (weights and biases). The data loss term can be a standard mean squared error (MSE) that compares predictions with observed data. The physics loss term enforces that predictions of the network satisfy the underlying physics, expressed as a partial differential equation (PDE) or other constraints using a differential operator representing the governing equations, which can be computed using automatic differentiation. For example, in the work of Eghbalian et al. [45], plasticity concepts like additive strain decomposition and hypoelasticity are incorporated into the network design to capture the pressure-dependent yielding and dilatancy behaviours of sand. This physics-informed structure enables more efficient training with improved generalization compared to standard neural networks, as evidenced by the PINN's stable predictions across diverse loading paths unseen in the training data. Although the PINN can improve the robustness of the ML model to a certain extent, they are still largely restricted to the boundary conditions on which they were trained and are not generalizable beyond the training regime. Additionally, because many granular media-related phenomena are not easily simplified using PDEs or Ordinary Differential Equations (ODEs), the application of PINNs in simulating granular media remains limited. Several examples in this field include using PINNs or neural operator for consolidation analysis in soils [117, 193] and for modelling particle aggregation and breakage processes in chemical engineering [29]. Both examples rely on the prerequisite that PDEs have been well-established to characterize these problems.

On the other hand, operator learning, particularly through the lens of FNOs [101, 135], offers a transformative approach to modelling in scientific computing. Traditional deep learning models are designed to learn mappings between finite-dimensional vectors, but operator learning seeks to understand the mappings between entire function spaces. This is achieved by training on pairs of input–output functions, rather than discrete data points. The incorporation of spectral methods allows the model to operate in infinite-dimensional function spaces without being constrained by mesh resolutions. By representing functions in the Fourier space, FNOs can efficiently learn operators by applying Fourier transforms and diagonal matrix multiplications within their layers. This not only reduces the need for excessive parametrization but also ensures that the learned operator behaves smoothly across the function space. Furthermore, the inherent properties of the Fourier space, such as smoothness and regularization, enable FNOs to efficiently capture global patterns and generalize well within the scope of the training data, making them a promising tool for a wide range of scientific applications.

Another significant advancement in operator learning is the Deep Operator Network (DeepONet; [63, 112, 117]). This architecture is grounded in the Universal Approximation Theorem for Operators, which extends the classical neural network theory to function spaces. The theorem posits that certain neural network architectures can approximate a wide class of nonlinear operators between function spaces with arbitrary precision. DeepONet implements this theorem through a novel two-part structure: a branch network and a trunk network. The branch network processes input functions, while the trunk network handles the evaluation points of output functions. DeepONet approximates operators by representing them as a sum of basis functions multiplied by coefficients, implemented through a dot product operation between the outputs of these networks. The trunk network learns the basis functions, while the branch network computes the coefficients based on the input function. This architecture allows DeepONet to solve multi-scale problems [107] and complex non-linear systems [117].

While FNOs and DeepONet operate in general Banach spaces, the Basis-to-Basis (B2B) approach [82] leverages the geometric properties of Hilbert spaces to generate more interpretable and generalizable operators. By exploiting the inner product structure, spectral theory, and optimization geometry of Hilbert spaces, B2B learns basis functions for both input and output spaces, and then maps between their coefficients. This structure allows B2B to handle variable input locations, interpolate effectively, and potentially extrapolate beyond the training domain, while offering improved interpretability and analytical tractability.

Although relatively unexplored in the space of granular media, operator learning approaches like FNO, DeepONet, and B2B have the potential to revolutionize and accelerate numerical simulations in this field. These methods offer promising avenues for modeling complex granular systems, predicting their behavior under various conditions, and potentially uncovering new insights into the fundamental physics governing granular media.

7 Conclusion

This work reviewed the major advancement in ML-aided modelling of granular media, specifically including the application of the ML method in microscopic grain scale computation, the development of the data-driven constitutive model for granular materials, and the ML-aided macroscopic simulation of granular media.

On the microscopic scale, we mainly focus on the recent development of two different types of grain information-based ML models. The first one is the contact information-based ML model, and the other is the grain-level kinematic features-based ML model. The high computation efficiency of the ML algorithm makes these models promising ways to accelerate the computational process of particle/grain-based numerical techniques, and the black-box characteristics make these ML models very user-friendly. However, it is essential to recognize that the development of such grain contact information-based ML models is also subjected to certain challenges. The variations of particle shape, diverse contact situations, and the absence of consistent contact theories make it nearly impossible to generate a high-quality and comprehensive training set that covers all contact conditions. Consequently, this limitation hinders the generalization of the ML-based contact surrogate models in practical engineering applications. On the other hand, it has been observed that a majority of existing kinematic features-based ML models ignored some essential physical motion information of grains, e.g. particle rotation, and angular velocity. These two main issues are the primary obstacles impeding the continued advancement of particle information-based ML models at the grain level.

Both time-sequential and single-step-based neural networks can be leveraged to derive the constitutive relationship of granular materials using various types of training data. Wherein the distinctive architecture of these time-sequence neural networks inherently enables them to capture the history-dependent stress–strain response of granular materials. For single-step-based networks, to achieve comparable performance to time-sequence neural networks, the incorporation of artificially introduced internal/historical variables is necessary to identify loading states. Since the ML-based constitutive models directly acquire the constitutive laws from generated stress–strain data, they can naturally circumvent assumption and intricate mathematical formulation and evolve continuously with the expansion of the training dataset, making them naturally overcome the problems confronted by the traditional continuum-theory-based phenomenological constitutive models. However, the progress of ML-based constitutive models is primarily hindered by their inherently weaker extrapolation ability of neural networks. This limitation implies that the training set for a general ML-based constitutive model has to encompass diverse stress–strain paths and various material types to the greatest extent, which is challenging to achieve. Therefore, most existing ML constitutive models can only accurately replicate the stress–strain response of specific granular materials under certain loading conditions.

Utilizing various neural networks, data-driven simulators or ML-aided numerical solvers have been developed to accelerate the macroscopic simulation process of granular materials. Due to their high computational efficiency and black-box characteristics, ML models bypass the most time-consuming process in traditional physical solving processes and directly provide essential state parameters for each solving iteration in the numerical algorithm, thereby accelerating the overall computation process. However, similar to the ML-based constitutive model and grain information-based ML simulations, the ML-assisted macroscopic simulation methods also suffer from the issue of weaker extrapolation capability, and most of the current works are case-by-case. Additionally, it is essential to consider the avoidance of the error accumulation problem, particularly when employing the ML model in explicit dynamic numerical methods to predict system rollouts over an extended duration. Any prediction error from previous time steps can impact subsequent predictions, potentially leading to the breakdown of the ML system.

Author contributions Mengqi Wang, Min Wang and Krishna Kumar wrote the main manuscript. Y. T. Feng and Tongming Qu reviewed the manuscript.

Data availability No datasets were generated or analysed during the current study.

Declarations

Competing Interests The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, et al. (2016) Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467
2. Alipour M, Lashkari A (2018) Sand instability under constant shear drained stress path. Int J Solids Struct 150:66–82
3. Amroune M (2021) Machine learning techniques applied to online voltage stability assessment: a review. Archi Comput Methods Eng 28:273–287
4. Andrade JE, Avila C, Hall SA, Lenoir N, Viggiani G (2011) Multiscale modeling and characterization of granular matter: from grain kinematics to continuum mechanics. J Mech Phys Solids 59:237–250
5. Andrade JE, Tu X (2009) Multiscale framework for behavior prediction in granular media. Mech Mater 41:652–669
6. Aoyama Y, Haeri A, Theodorou EA (2023) Optimal control of granular material. arXiv preprint arXiv:2302.03231
7. Araei AA (2014) Artificial neural networks for modeling drained monotonic behavior of rockfill materials. Int J Geomech 14:04014005
8. Baccouche M, Mamalet F, Wolf C, Garcia C, Baskurt A (2011) Sequential deep learning for human action recognition, in: Human Behavior Understanding: Second International Workshop, HBU 2011, Amsterdam, The Netherlands, November 16, 2011. Proceedings 2, Springer. pp. 29–39
9. Bagheri G, Bonadonna C, Manzella I, Vonlanthen P (2015) On the characterization of size and shape of irregular particles. Powder Technol 270:141–153
10. Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473
11. Bai S, Kolter JZ, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271
12. Bandara S, Soga K (2015) Coupling of soil deformation and pore fluid flow using material point method. Comput Geotech 63:199–214
13. Banimahd M, Yasrobi S, Woodward PK (2005) Artificial neural network for stress-strain behavior of sandy soils: knowledge based verification. Comput Geotech 32:377–386
14. Bapst V, Keck T, Grabska-Barwińska A, Donner C, Cubuk ED, Schoenholz SS, Obika A, Nelson AW, Back T, Hassabis D et al (2020) Unveiling the predictive power of static structure in glassy systems. Nat Phys 16:448–454
15. Bardenhagen S, Brackbill J, Sulsky D (2000) The material-point method for granular materials. Comput Methods Appl Mech Eng 187:529–541
16. Basheer I (2002) Stress-strain behavior of geomaterials in loading reversal simulated by time-delay neural networks. J Mater Civil Eng 14:270–273
17. Basheer IA (2000) Selection of methodology for neural network modeling of constitutive hystereses behavior of soils. Comput Aided Civil Infrastruct Eng 15:445–463
18. Battaglia P, Pascanu R, Lai M, Jimenez Rezende D et al (2016) Interaction networks for learning about objects, relations and physics. Adv Neural Inf Process Syst 29:4509
19. Belytschko T, Lu YY, Gu L (1994) Element-free galerkin methods. Int J Numer Methods Eng 37:229–256
20. Beuth L, Benz T, Vermeer PA, Wieckowski Z (2008) Large deformation analysis using a quasi-static material point method. J Theor Appl Mech 38:45–60

21. Beuth L, Wikeckowski Z, Vermeer P (2011) Solution of quasi-static large-strain problems by the material point method. Int J Numer Anal Methods Geomech 35:1451–1465
22. Bowman ET, Soga K, Drummond W (2001) Particle shape characterisation using fourier descriptor analysis. Geotechnique 51:545–554
23. Brinkgreve RB (2005) Selection of soil models and parameters for geotechnical engineering application.Proceedings of Geo-Frontiers 2005, in Austin, Texas, pp: 69–98
24. Bui HH, Fukagawa R, Sako K, Ohno S (2008) Lagrangian mesh-free particles method (sph) for large deformation and failure flows of geomaterial using elastic-plastic soil constitutive model. Int J Numer Anal Methods Geomech 32:1537–1570
25. Bui HH, Sako K, Fukagawa R (2007) Numerical simulation of soil-water interaction using smoothed particle hydrodynamics (sph) method. J Terramech 44:339–346
26. Bui HH, Sako K, Fukagawa R, Wells J (2008) Sph-based numerical simulations for large deformation of geomaterial considering soil-structure interaction. In: The 12th international conference of international association for computer methods and advances in geomechanics (IACMAG), pp. 570–578
27. Chang CS, Liao CL (1994) Estimates of elastic modulus for media of randomly packed granules. Appl Mech Rev. https://doi.org/10.1115/1.3122814
28. Chang MB, Ullman T, Torralba A, Tenenbaum JB (2016) A compositional object-based approach to learning physical dynamics. arXiv preprint arXiv:1612.00341
29. Chen X, Wang LG, Meng F, Luo ZH (2021) Physics-informed deep learning for modelling particle aggregation and breakage processes. Chem Eng J 426:131220
30. Cheng Z, Wang J (2022) Estimation of contact forces of granular materials under uniaxial compression based on a machine learning model. Granul Matter 24:1–14
31. Cho K, Van Merrienboer B, Bahdanau D, Bengio Y (2014) On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259
32. Choi Y, Kumar K (2024) Graph neural network-based surrogate model for granular flows. Comput Geotech 166:106015
33. Choi Y, Kumar K (2024) Inverse analysis of granular flows using differentiable graph neural network simulator. Comput Geotech 171:106374
34. Ciregan D, Meier U, Schmidhuber J (2012) Multi-column deep neural networks for image classification, In: 2012 IEEE conference on computer vision and pattern recognition, IEEE. pp. 3642–3649
35. Ciresan DC, Meier U, Masci J, Gambardella LM, Schmidhuber J (2011) Flexible, high performance convolutional neural networks for image classification. In: Twenty-second international joint conference on artificial intelligence, Citeseer
36. Cleary PW, Sawley ML (2002) Dem modelling of industrial granular flows: 3d case studies and the effect of particle shape on hopper discharge. Appl Math Model 26:89–111
37. Cundall PA (1971) A computer model for simulating progressive, large-scale movement in blocky rock system. In: Proceedings of the international symposium on rock mechanics, pp. 129–136
38. Cundall PA (1974) Rational design of tunnel supports: a computer model for rock mass behaviour using interactive graphics for the input and output of geometrical data. Technical Report
39. Cundall PA, Strack OD (1979) A discrete numerical model for granular assemblies. Geotechnique 29:47–65
40. Cybenko G (1989) Approximation by superpositions of a sigmoidal function. Math Control Signals Syst 2:303–314
41. Das SK, Das A (2019) Influence of quasi-static loading rates on crushable granular materials: a dem analysis. Powder Technol 344:393–403
42. Deen N, Annaland MVS, Van der Hoef MA, Kuipers J (2007) Review of discrete particle modeling of fluidized beds. Chem Eng Sci 62:28–44
43. Duncan JM, Chang CY (1970) Nonlinear analysis of stress and strain in soils. J Soil Mech Found Division 96:1629–1653
44. Eggersmann R, Kirchdoerfer T, Reese S, Stainier L, Ortiz M (2019) Model-free data-driven inelasticity. Comput Methods Appl Mech Eng 350:81–99
45. Eghbalian M, Pouragha M, Wan R (2023) A physics-informed deep neural network for surrogate modeling in classical elasto-plasticity. Comput Geotech 159:105472
46. Elman JL (1990) Finding structure in time. Cognit Sci 14:179–211
47. Feng Y (2021) An energy-conserving contact theory for discrete element modelling of arbitrarily shaped particles: Basic framework and general contact model. Comput Methods Appl Mech Eng 373:113454
48. Feng Y (2021) A generic energy-conserving discrete element modeling strategy for concave particles represented by surface triangular meshes. Int J Numer Methods Eng 122:2581–2597
49. Feng Y (2023) Thirty years of developments in contact modelling of non-spherical particles in dem: a selective review. Acta Mech Sinica 39:722343
50. Feng Y, Gao W (2021) On the strain energy distribution of two elastic solids under smooth contact. Powder Technol 389:376–382
51. Feng Y, Han K, Owen D (2017) A generic contact detection framework for cylindrical particles in discrete element modelling. Comput Methods Appl Mech Eng 315:632–651
52. Fern J, Rohe A, Soga K, Alonso E (2019) The material point method for geotechnical engineering: a practical guide. CRC Press, Boca Raton
53. Fragkiadaki K, Agrawal P, Levine S, Malik J (2015) Learning visual predictive models of physics for playing billiards. arXiv preprint arXiv:1511.07404
54. Fu P, Walton OR, Harvey JT (2012) Polyarc discrete element for efficiently simulating arbitrarily shaped 2d particles. Int J Numer Methods Eng 89:599–617
55. Fu Q, Hashash YM, Jung S, Ghaboussi J (2007) Integration of laboratory testing and constitutive modeling of soils. Comput Geotech 34:330–345
56. Fuhg JN, Marino M, Bouklas N (2022) Local approximate gaussian process regression for data-driven constitutive models: development and comparison with neural networks. Comput Methods Appl Mech Eng 388:114217
57. Fukushima K (1980) Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol Cybern 36:193–202
58. Ghaboussi J, Garrett J Jr, Wu X (1991) Knowledge-based modeling of material behavior with neural networks. J Eng Mech 117:132–153
59. Ghaboussi J, Pecknold DA, Zhang M, Haj-Ali RM (1998) Autoprogressive training of neural network constitutive models. Int J Numer Methods Eng 42:105–126
60. Ghaboussi J, Sidarta DE (1998) New nested adaptive neural networks (nann) for constitutive modeling. Comput Geotech 22:29–52
61. Ghavamian F, Simone A (2019) Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network. Comput Methods Appl Mech Eng 357:112594
62. Gingold RA, Monaghan JJ (1977) Smoothed particle hydrodynamics: theory and application to non-spherical stars. Mon Not Royal Astron Soc 181:375–389
63. Goswami S, Bora A, Yu Y, Karniadakis GE (2023) Physics-informed deep neural operator networks. Machine learning in

modeling and simulation: methods and applications. Springer, Cham, pp 219–254
64. Graves A (2013) Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850
65. Guan Q, Yang Z, Guo N, Hu Z (2023) Finite element geotechnical analysis incorporating deep learning-based soil model. Comput Geotech 154:105120
66. Guan S, Qu T, Feng Y, Ma G, Zhou W (2023) A machine learning-based multi-scale computational framework for granular materials. Acta Geotechnica 18:1699–1720
67. Guan S, Zhang X, Ranftl S, Qu T (2023) A neural network-based material cell for elastoplasticity and its performance in fe analyses of boundary value problems. Int J Plastic 171:103811
68. Guo N, Zhao J (2014) A coupled fem/dem approach for hierarchical multiscale modelling of granular media. Int J Numer Methods Eng 99:789–818
69. Guo N, Zhao J (2016) 3d multiscale modeling of strain localization in granular media. Comput Geotech 80:360–372
70. Habibagahi G, Bamdad A (2003) A neural network framework for mechanical behavior of unsaturated soils. Can Geotech J 40:684–693
71. Harlow FH (1964) The particle-in-cell computing method for fluid dynamics. Methods Comput Phys 3:319–343
72. Hashash Y, Jung S, Ghaboussi J (2004) Numerical implementation of a neural network based material model in finite element analysis. Int J Numer Methods Eng 59:989–1005
73. Hashash Y, Song H (2008) The integration of numerical modeling and physical measurements through inverse analysis in geotechnical engineering. KSCE J Civil Eng 12:165–176
74. He S, Li J (2009) Modeling nonlinear elastic behavior of reinforced soil using artificial neural networks. Appl Soft Comput 9:954–961
75. He S, Li Y, Feng Y, Ho S, Ravanbakhsh S, Chen W, Poczos B (2019) Learning to predict the cosmological structure formation. Proceed Nat Acad Sci 116:13825–13832
76. Hertz H (1882) Ueber die berührung fester elastischer körper
77. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9:1735–1780
78. Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2:359–366
79. Hu X, Zhang Y, Guo L, Wang J, Cai Y, Fu H, Cai Y (2018) Cyclic behavior of saturated soft clay under stress path with bidirectional shear stresses. Soil Dyn Earthq Eng 104:319–328
80. Huang D, Fuhg JN, Weißenfels C, Wriggers P (2020) A machine learning based plasticity model using proper orthogonal decomposition. Comput Methods Appl Mech Eng 365:113008
81. Hwang S, Pan J, Sunny AA, Fan LS (2022) A machine learning-based particle-particle collision model for non-spherical particles with arbitrary shape. Chem Eng Sci 251:117439
82. Ingebrand T, Thorpe AJ, Goswami S, Kumar K, Topcu U (2024) Basis-to-basis operator learning using function encoders. arXiv preprint arXiv:2410.00171
83. Ji S, Xu W, Yang M, Yu K (2012) 3d convolutional neural networks for human action recognition. IEEE Trans Pattern Anal Mach Intel 35:221–231
84. Johari A, Javadi A, Habibagahi G (2011) Modelling the mechanical behaviour of unsaturated soils using a genetic algorithm-based neural network. Comput Geotech 38:2–13
85. Johnson KL (1987) Contact mechanics. Cambridge University Press, Cambridge
86. Jung S, Ghaboussi J (2006) Neural network constitutive model for rate-dependent materials. Comput Struct 84:955–963
87. Karapiperis K, Stainier L, Ortiz M, Andrade JE (2021) Data-driven multiscale modeling in mechanics. J Mech Phys Solids 147:104239
88. Kohestani V, Hassanlourad M (2016) Modeling the mechanical behavior of carbonate sands using artificial neural networks and support vector machines. Int J Geomech 16:04015038
89. Kruyt NP, Rothenburg L (1998) Statistical theories for the elastic moduli of two-dimensional assemblies of granular materials. Int J Eng Sci 36:1127–1142
90. Kumar K, Choi Y (2023) Accelerating particle and fluid simulations with differentiable graph networks for solving forward and inverse problems, in: Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis, pp. 60–65
91. Ladicky L, Jeong S, Solenthaler B, Pollefeys M, Gross M (2015) Data-driven fluid simulations using regression forests. ACM Trans Gr (TOG) 34:1–9
92. Lai Z, Chen Q (2017) Characterization and discrete element simulation of grading and shape-dependent behavior of jsc-1a martian regolith simulant. Granul Matter 19:69
93. Lai Z, Chen Q (2019) Reconstructing granular particles from x-ray computed tomography using the tws machine learning tool and the level set method. Acta Geotech 14:1–18
94. Lai Z, Chen Q, Huang L (2020) Fourier series-based discrete element method for computational mechanics of irregular-shaped particles. Comput Methods Appl Mech Eng 362:112873
95. Lai Z, Chen Q, Huang L (2022) Machine-learning-enabled discrete element method: contact detection and resolution of irregular-shaped particles. Int J Numer Anal Methods Geomech 46:113–140
96. Le B, Yvonnet J, He QC (2015) Computational homogenization of nonlinear elastic materials using neural networks. Int J Numer Methods Eng 104:1061–1084
97. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521:436–444
98. Lee S, Ha J, Zokhirova M, Moon H, Lee J (2018) Background information of deep learning for structural engineering. Arch Comput Methods Eng 25:121–129
99. Li H, Yu H, Cao N, Tian H, Cheng S (2021) Applications of artificial intelligence in oil and gas development. Arch Comput Methods Eng 28:937–949
100. Li Z, Chow JK, Wang YH (2017) Applying the artificial neural network to predict the soil responses in the dem simulation. In: IOP Conference Series: Materials Science and Engineering, IOP Publishing. p. 012040
101. Li Z, Huang DZ, Liu B, Anandkumar A (2023) Fourier neural operator with learned deformations for pdes on general geometries. J Mach Learn Res 24:1–26
102. Li Z, Li X, Zhang H, Huang D, Zhang L (2023) The prediction of contact force networks in granular materials based on graph neural networks. J Chem Phys 158:5
103. Liang W, Zhao J (2019) Coupled mpm/dem multiscale modelling geotechnical problems involving large deformation. In: 16th Asian Regional Conference on sSoil Mechanics and Geotechnical Engineering
104. Liang W, Zhao J (2019) Multiscale modeling of large deformation in geomechanics. Int J Numer Anal Methods Geomech 43:1080–1114
105. Lim KW, Andrade JE (2013) Granular element method for computational particle mechanics. Computer Methods in Applied Mechanics and Engineering, 241, 262–274
106. Lim KW, Andrade JE (2014) Granular element method for three-dimensional discrete element calculations. Int J Numer Anal Methods Geomech 38:167–188

107. Liu L, Cai W (2021) Multiscale deeponet for nonlinear operators in oscillatory function spaces for building seismic wave responses. arXiv preprint arXiv:2111.04860
108. Liu M, Liu G (2010) Smoothed particle hydrodynamics (sph): an overview and recent developments. Arch Comput Methods Eng 17:25–76
109. Liu Z, Su L, Zhang C, Iqbal J, Hu B, Dong Z (2020) Investigation of the dynamic process of the xinmo landslide using the discrete element method. Comput Geotech 123:103561
110. Logarzo HJ, Capuano G, Rimoli JJ (2021) Smart constitutive laws: inelastic homogenization through machine learning. Comput Methods Appl Mech Eng 373:113482
111. Lu L, Gao X, Dietiker JF, Shahnam M, Rogers WA (2021) Machine learning accelerated discrete element modeling of granular flows. Chem Eng Sci 245:116832
112. Lu L, Jin P, Pang G, Zhang Z, Karniadakis GE (2021) Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. Nat Mach Intel 3:218–229
113. Lucy LB (1977) A numerical approach to the testing of the fission hypothesis. Astronom J 82:1013–1024
114. Lv Y, Nie L, Xu K (2011) Study of the neural network constitutive models for turfy soil with different decomposition degree. In: 2011 Second International Conference on Mechanic Automation and Control Engineering, IEEE. pp. 6111–6114
115. Ma G, Guan S, Wang Q, Feng Y, Zhou W (2022) A predictive deep learning framework for path-dependent mechanical behavior of granular materials. Acta Geotech 17:3463–3478
116. Ma X, Zhang DZ (2006) Statistics of particle interactions in dense granular material under uniaxial compression. J Mech Phys Solids 54:1426–1448
117. Mandl L, Goswami S, Lambers L, Ricken T (2024) Separable deeponet: Breaking the curse of dimensionality in physics-informed machine learning. arXiv preprint arXiv:2407.15887
118. Mašín D (2005) A hypoplastic constitutive model for clays. Int J Numer Anal Methods Geomech 29:311–336
119. Mayr A, Lehner S, Mayrhofer A, Kloss C, Hochreiter S, Brandstetter J (2023) Boundary graph neural networks for 3d simulations. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 9099–9107
120. Micheli A (2009) Neural network for graphs: a contextual constructive approach. IEEE Trans Neural Netw 20:498–511
121. Nitka M, Combe G, Dascalu C, Desrues J (2011) Two-scale modeling of granular materials: a dem-fem approach. Granul Matter 13:277–281
122. Oñate E, Idelsohn SR, Del Pin F, Aubry R (2004) The particle finite element method–an overview. Int J Comput Methods 1:267–307
123. Pande G, Pietruszczak S, Wang M (2020) Role of gradation curve in description of mechanical behavior of unsaturated soils. Int J Geomech 20:04019159
124. Penumadu D, Zhao R (1999) Triaxial compression behavior of sand and gravel using artificial neural networks (ann). Comput Geotech 24:207–230
125. Petalas AL, Dafalias YF, Papadimitriou AG (2020) Sanisand-f: sand constitutive model with evolving fabric anisotropy. Int J Solids Struct 188:12–31
126. Peters JF, Hopkins MA, Kala R, Wahl RE (2009) A poly-ellipsoid particle for non-spherical discrete element method. Eng Comput 26:645–657
127. Poorooshasb HB, Pietruszczak S (1985) On yielding and flow of sand; a generalized two-surface model. Comput Geotech 1:1
128. Qu T, Di S, Feng Y, Wang M, Zhao T (2021) Towards data-driven constitutive modelling for granular materials via micromechanics-informed deep learning. Int J Plastic 144:103046
129. Qu T, Feng Y, Wang M (2021) An adaptive granular representative volume element model with an evolutionary periodic boundary for hierarchical multiscale analysis. Int J Numer Methods Eng 122:2239–2253
130. Qu T, Guan S, Feng Y, Ma G, Zhou W, Zhao J (2023) Deep active learning for constitutive modelling of granular materials: From representative volume elements to implicit finite element modelling. Int J Plastic 164:103576
131. Qu T, Zhao J, Guan S, Feng Y (2023) Data-driven multiscale modelling of granular materials via knowledge transfer and sharing. Int J Plastic 171:103786
132. Raissi M, Perdikaris P, Karniadakis GE (2019) Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput phys 378:686–707
133. Rangel RL, Franci A, Oñate E, Gimenez JM (2024) Multi-scale data-driven modeling of the thermomechanical behavior of granular media with thermal expansion effects. Comput Geotech 176:106789
134. Rashidian V, Hassanlourad M (2014) Application of an artificial neural network for modeling the mechanical behavior of carbonate soils. Int J Geomech 14:142–150
135. Roberts N, Khodak M, Dao T, Li L, Ré C, Talwalkar A (2021) Learning operations for neural pde solvers. In: Proc. ICLR SimDL Workshop
136. Romo MP, García SR, Mendoza MJ, Taboada-Urtuzuástegui V (2001) Recurrent and constructive-algorithm networks for sand behavior modeling. Int J Geomech 1:371–387
137. Roscoe K, Burland JB (1968) On the Generalized Stress-Strain Behavior of Wet Clay. In: Heyman, J. andLeckie, F., Eds., Engineering Plasticity, Cambridge University Press, Cambridge, 535–609
138. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323:533–536
139. Sanchez-Gonzalez A, Godwin J, Pfaff T, Ying R, Leskovec J, Battaglia P (2020) Learning to simulate complex physics with graph networks. In: International conference on machine learning, PMLR. pp. 8459–8468
140. Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G (2008) The graph neural network model. IEEE Trans Neural Netw 20:61–80
141. Sezer A (2011) Prediction of shear development in clean sands by use of particle shape information and artificial neural networks. Expert Syst Appl 38:5603–5613
142. Shahin MA, Indraratna B (2006) Modeling the mechanical behavior of railway ballast using artificial neural networks. Can Geotech J 43:1144–1152
143. Sidarta D, Ghaboussi J (1998) Constitutive modeling of geomaterials from non-uniform material tests. Comput Geotech 22:53–71
144. Sołowski W, Sloan S (2015) Evaluation of material point method for use in geotechnics. Int J Numer Anal Methods Geomech 39:685–701
145. Spengler M (1999) Fast neural network emulation and control of physics-based models. Proceedings of the 25th annual conference on Computer graphics and interactive techniques, in Orlando, Florida, pp: 9–20
146. Stefanos D, Gyan P (2015) On neural network constitutive models for geomaterials. J Civil Eng Res 5:106–113
147. Strack O, Cundall PA (1978) The distinct element method as a tool for research in granular media. University of Minnesota, Minnesota
148. Sulsky D, Zhou SJ, Schreyer HL (1995) Application of a particle-in-cell method to solid mechanics. Comput Phys Commun 87:236–252
149. Sutskever I, Martens J, Hinton GE (2011) Generating text with recurrent neural networks. In: Proceedings of the 28th international conference on machine learning (ICML-11), pp. 1017–1024

150. Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. Adv Neural Inf Process Syst 27:1409
151. Tavarez FA, Plesha ME (2007) Discrete element method for modelling solid and particulate materials. Int J Numer Methods Eng 70:379–404
152. Thakur MM, Penumadu D (2020) Triaxial compression in sands using fdem and micro-x-ray computed tomography. Comput Geotech 124:103638
153. Ti KS, Huat B, Noorzaei J, Jaafar MS, Sew GS (2009) A review of basic soil constitutive models for geotechnical application. Electron J Geotech Eng 14:1–18
154. Tian Y, Yao YP (2017) Modelling the non-coaxiality of soils from the view of cross-anisotropy. Comput Geotech 86:219–229
155. Ueda K, Iai S (2019) Constitutive modeling of inherent anisotropy in a strain space multiple mechanism model for granular materials. Int J Numer Anal Methods Geomech 43:708–737
156. Ummenhofer B, Prantl L, Thuerey N, Koltun V (2019) Lagrangian fluid simulation with continuous convolutions. In: International conference on learning representations
157. Vlassis NN, Sun W (2021) Sobolev training of thermodynamic-informed neural networks for interpretable elasto-plasticity models with level set hardening. Comput Methods Appl Mech Eng 377:113695
158. Voyiadjis GZ, Alsaleh MI, Alshibli KA (2005) Evolving internal length scales in plastic strain localization for granular materials. Int J Plastic 21:2000–2024
159. Wang J, Chan D (2014) Frictional contact algorithms in sph for the simulation of soil-structure interaction. Int J Numer Anal Methods Geomech 38:747–770
160. Wang K, Sun W (2019) Meta-modeling game for deriving theory-consistent, microstructure-based traction-separation laws via deep reinforcement learning. Comput Methods Appl Mech Eng 346:216–241
161. Wang L, Cai Y, Liu D (2018) Multiscale reliability-based topology optimization methodology for truss-like microstructures with unknown-but-bounded uncertainties. Comput Methods Appl Mech Eng 339:358–388
162. Wang M, Feng Y, Guan S, Qu T (2024) Multi-layer perceptron-based data-driven multiscale modelling of granular materials with a novel frobenius norm-based internal variable. J Rock Mech Geotech Eng. https://doi.org/10.1016/j.jrmge.2024.02.003
163. Wang M, Qu T, Guan S, Zhao T, Liu B, Feng Y (2022) Data-driven strain-stress modelling of granular materials via temporal convolution neural network. Comput Geotech 152:105049
164. Wang M, Zhang DZ (2021) Deformation accommodating periodic computational domain for a uniform velocity gradient. Comput Methods Appl Mech Eng 374:113607
165. Wang X, Yin ZY, Su D, Xiong H, Feng Y (2021) A novel arcs-based discrete element modeling of arbitrary convex and concave 2d particles. Comput Methods Appl Mech Eng 386:114071
166. Wang Z, Liu K, Li J, Zhu Y, Zhang Y (2019) Various frameworks and libraries of machine learning and deep learning: a survey. Arch Comput Methods Eng 1:1–24
167. Weng JJ, Ahuja N, Huang TS (1993) Learning recognition and segmentation of 3-d objects from 2-d images. In: 1993 (4th) International Conference on Computer Vision, IEEE. pp. 121–128
168. Werbos PJ (1990) Backpropagation through time: what it does and how to do it. Proceed IEEE 78:1550–1560
169. Wiewel S, Becher M, Thuerey N (2019) Latent space physics: towards learning the temporal evolution of fluid flow. Comput Gr forum. Wiley Online Library, Hoboken, pp 71–82
170. Wikeckowski Z (2004) The material point method in large strain engineering problems. Comput Methods Appl Mech Eng 193:4417–4438
171. Williams JR, O'Connor R (1999) Discrete element simulation and the contact problem. Arch Comput Methods Eng 6:279–304
172. Williams JR, Pentland AP (1992) Superquadrics and modal dynamics for discrete elements in interactive design. Eng Comput 9:115–127
173. Wood DM (2017) Geotech Model. CRC Press, Boca Raton
174. Wriggers P (2008) Nonlinear finite element methods. Springer science & business media, Cham
175. Wu J, Yildirim I, Lim JJ, Freeman B, Tenenbaum J (2015) Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. Adv Neural Inf Process Syst 28:1
176. Wu L, Cui P, Pei J, Zhao L, Guo X (2022) Graph neural networks: foundation, frontiers and applications. In: Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining, pp. 4840–4841
177. Wu W, Bauer E, Kolymbas D (1996) Hypoplastic constitutive model with critical state for granular materials. Mech Mater 23:45–69
178. Xu K, Huang DZ, Darve E (2021) Learning constitutive relations using symmetric positive definite neural networks. J Comput Phys 428:110072
179. Yao Y, Sun D, Luo T (2004) A critical state model for sands dependent on stress and density. Int J Numer Anal Methods Geomech 28:323–337
180. Yao YP, Hou W, Zhou AN (2009) Uh model: three-dimensional unified hardening model for overconsolidated clays. Geotechnique 59:451–469
181. Yimsiri S, Soga K (2000) Micromechanics-based stress-strain behaviour of soils at small strains. Géotechnique 50:559–571
182. Yin ZY, Jin YF (2019) Practice of optimisation theory in geotechnical engineering. Springer, Cham
183. Yin ZY, Karstunen M, Chang CS, Koskinen M, Lojander M (2011) Modeling time-dependent behavior of soft sensitive clay. J Geotech Geoenviron Eng 137:1103–1113
184. Yin ZY, Wang P, Zhang F (2020) Effect of particle shape on the progressive failure of shield tunnel face in granular soils by coupled fdm-dem method. Tunnel Undergr Space Technol 100:103394
185. You Z (2003) Development of a micromechanical modeling approach to predict asphalt mixture stiffness using the discrete element method. University of Illinois at Urbana-Champaign, Champaign
186. Zhang DZ, Ma X, Giguere PT (2011) Material point method enhanced by modified gradient of shape function. J Comput Phys 230:6379–6398
187. Zhang DZ, Rauenzahn RM (2000) Stress relaxation in dense and slow granular flows. J Rheol 44:1019–1041
188. Zhang N, Shen SL, Zhou A, Xu YS (2019) Investigation on performance of neural networks using quadratic relative error cost function. IEEE Access 7:106642–106652
189. Zhang P, Yang Y, Yin ZY (2021) Bilstm-based soil-structure interface modeling. Int J Geomech 21:04021096
190. Zhang P, Yin ZY, Jin YF (2021) State-of-the-art review of machine learning applications in constitutive modeling of soils. Arch Comput Methods Eng 28:3661–3686
191. Zhang P, Yin ZY, Jin YF, Liu XF (2021) Modelling the mechanical behaviour of soils using machine learning algorithms with explicit formulations. Acta Geotech 1:1–20
192. Zhang P, Yin ZY, Jin YF, Ye GL (2020) An ai-based model for describing cyclic characteristics of granular materials. Int J Numer Anal Methods Geomech 44:1315–1335
193. Zhang S, Lan P, Li HC, Tong CX, Sheng D (2022) Physics-informed neural networks for consolidation of soils. Eng Comput 39:2845–2865

194. Zhou W, Huang Y, Ng TT, Ma G (2018) A geometric potential-based contact detection algorithm for egg-shaped particles in discrete element modeling. Powder Technol 327:152–162
195. Zhu JH, Zaman MM, Anderson SA (1998) Modelling of shearing behaviour of a residual soil with recurrent neural network. Int J Numer Anal Methods Geomech 22:671–687
196. Zienkiewicz OC, Taylor RL, Zhu JZ (2005) The finite element method: its basis and fundamentals. Elsevier, Amsterdam

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.