Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 373 (2021) 113514
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

A non-cooperative meta-modeling game for automated third-party calibrating, validating and falsifying constitutive laws with parallelized adversarial attacks

Kun Wang[a,b], WaiChing Sun[b,*], Qiang Du[c]

[a] Fluid Dynamics and Solid Mechanics Group, Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA
[b] Department of Civil Engineering and Engineering Mechanics, Columbia University, New York, NY 10027, USA
[c] Department of Applied Physics and Applied Mathematics, and Data Science Institute, Columbia University, New York, NY 10027, USA

Available online xxxx

Abstract

The evaluation of constitutive models, especially for high-risk and high-regret engineering applications, requires efficient and rigorous third-party calibration, validation and falsification. While there are numerous efforts to develop paradigms and standard procedures to validate models, difficulties may arise due to the sequential, manual, and often biased nature of the commonly adopted calibration and validation processes, thus slowing down data collections, hampering the progress towards discovering new physics, increasing expenses and possibly leading to misinterpretations of the credibility and application ranges of proposed models. This work attempts to introduce concepts from game theory and machine learning techniques to overcome many of these existing difficulties. We introduce an automated meta-modeling game where two competing AI agents systematically generate experimental data to calibrate a given constitutive model and to explore its weakness such that the experiment design and model robustness can be improved through competitions. The two agents automatically search for the Nash equilibrium of the meta-modeling game in an adversarial reinforcement learning framework without human intervention. In particular, a protagonist agent seeks to find the more effective ways to generate data for model calibrations, while an adversary agent tries to find the most devastating test scenarios that expose the weaknesses of the constitutive model calibrated by the protagonist. By capturing all possible design options of the laboratory experiments into a single decision tree, we recast the design of experiments as a game of combinatorial moves that can be resolved through deep reinforcement learning by the two competing players. Our adversarial framework emulates idealized scientific collaborations and competitions among researchers to achieve a better understanding of the application range of the learned material laws and prevent misinterpretations caused by conventional AI-based third-party validation. Numerical examples are given to demonstrate the wide applicability of the proposed meta-modeling game with adversarial attacks on both human-crafted constitutive models and machine learning models.

Keywords: Data-driven computational mechanics; Multi-agent deep reinforcement learning; Adversarial learning; Directed graph

* Correspondence to: WaiChing Sun, Associate Professor, Department of Civil Engineering and Engineering Mechanics, Columbia University, 614 SW Mudd, Mail Code: 4709, New York, NY 10027, USA.
E-mail addresses: kunw@lanl.gov (K. Wang), wsun@columbia.edu (W. Sun), qd2125@columbia.edu (Q. Du).

https://doi.org/10.1016/j.cma.2020.113514

1. Introduction and background

As constitutive models that predict material responses become increasingly sophisticated and complex, the demands and difficulties for accurately calibrating and validating those constitutive laws also increase [1–12]. Engineering applications, particularly those involve high-risk and high-regret decision-makings, require models to maintain robustness and accuracy in unforeseen scenarios using as little amount of necessary calibration data as possible. Quantifying the reliability of a new constitutive law, however, is nontrivial. As many constitutive models are calibrated against a limited amount and variety of experimental data, identifying the reliable application range of these constitutive laws beyond the loading paths used in the calibration could be challenging. While an apparent good match between predictions and calibration data can be easily achieved by increasing the dimensionality of the parametric space of a given model, over-fitting may also jeopardize the application of the learned models in the case where the predicted loading path bears little resemblance to the calibration data [13–15]. Using such constitutive models therefore bears more risks for a third-party user who is unaware of the sensitivity of material parameters on the stress predictions. Furthermore, the culture and the ecosystem of the scientific communities often encourage researchers to place a more significant focus on reporting the success of the material models in limited cases. Yet, precise and thorough investigations on the weakness and shortcomings of material models are important and often necessary even though they are less likely to be reported or documented in the literature due to the lack of incentive [16,17].

Model calibration issues are critical not only for hand-crafted models but for many machine learning models and data-driven framework that either directly use experimental data to replace constitutive laws [18–20] or generate optimal response surfaces via optimization problems [21–23]. The recent trend of using black-box deep neural networks (DNN) to generate constitutive laws has made the reliability analysis even more crucial. Due to the lack of interpretability of predictions generated by neural networks, the reliability of DNN generated constitutive laws is often only assessed through uncertainty quantification (UQ) [22,23]. UQ can be conducted via different procedures, including Bayesian statistics, polynomial chaos expansion and Monte Carlo sampling where one seeks to understand how probability distributions of the input material parameters affect the outcomes of predictions, as often represented by some stress measures or performance metrics for solid mechanics applications. While UQ is a crucial step to ensure the readiness of constitutive laws for engineering applications, a common challenge is to detect rare events where a catastrophic loss of the prediction accuracy may occur in otherwise highly accurate constitutive laws.

The machine learning research community has been proposing methods to improve the interpolation and generalization capabilities, hence improving the predictive capability with exogenous data as well as reducing the epistemic uncertainties of trained neural network models. For instance, the active learning approach (e.g. [24]), which is sometimes referred as an “optimal experimental design” [25], introduces query strategies to choose what data to be generated to reduce generalization errors, balance exploration and exploitation and quantify uncertainties. These approaches have repeatedly outperformed traditional “passive learning” methods which involve randomly gathering a large amount of training data. Active learning is widely investigated using different deep learning algorithms like CNNs and LSTMs [26,27]. There is also research on implementing Generative Adversarial Networks (GANs) into the active learning framework [28]. With the increasing interest in deep reinforcement learning, researchers are trying to re-frame active learning as a reinforcement learning problem [29]. Another recent study focuses on the “semi-supervised learning” approaches [30–32], which take advantage of the structures of unlabeled input data to enhance the “interpolation consistency”, in addition to labeled training data. These recently developed techniques have shown some degrees of successes for image recognition, natural language processing, and therefore could potentially be helpful for mechanics problems.

The calibration, validation, and falsification of material models have issues similar to those discussed above. Moreover, experimental data are often expensive to get in both time and cost. Hence, experimentalists would like to generate the least amount of data that can calibrate a constitutive model with the highest reliability and can also identify its limitations. Traditionally the decisions on which experiments to conduct are based on human knowledge and experiences. We make an effort here to use AI to assist the decision-makings of experimentalists, which will be the first of its kind specifically targeting the automated design of data generation that can efficiently calibrate and falsify a constitutive model.

The major contribution of this paper is the introduction of a non-cooperative game that leads to new optimized experimental designs that both improve the accuracy and robustness of the predictions on unseen data, while at the same time exposing any potential weakness and shortcoming of a constitutive law.

We create a non-cooperative game in which a pair of agents are trained to emulate a form of artificial intelligence capable of improving their performance through repeated trial-and-error. The two agents play against each other in a turn-based strategy game, according to their own agendas and purposes respectively that serve to achieve opposite objectives. This setup constitutes a game in which each agent is rewarded by competing against the opponent. While the protagonist agent learns to validate models by designing the experiments than enhance the model predictions, the adversary agent learns how to undermine the protagonist agent by designing experiments that expose the weakness of the models. The optimal game strategies for both players are explored by searching for Nash equilibrium [33] of the games using deep reinforcement learning (DRL).

With recent rapid development, DRL techniques have found unprecedented success in the last decades on achieving superhuman intelligence and performance in playing increasingly complex games: Atari [34], board games [35,36], Starcraft [37]. AlphaZero [35] is also capable of learning the game strategies of our game without human knowledge. By emulating the learning process of human learners through trial-and-error and competition, the DRL process enables both AI agents to learn from their own successes and failures but also through their competitions to master the tasks of calibrating and falsifying a constitutive law. The knowledge gained from the competitions will help us understand the relative rewards of different experimental setup for validation and falsification mathematically represented by the decision trees corresponding to the protagonist and adversary agents.

The rest of the paper is organized as follows. We first describe the meta-modeling non-cooperative game, including the method to recast the design of experiments into decision trees (Section 2). Following this, we will introduce the detailed design of the calibration–falsification game for modeling the competition between the AI experimental agent and the AI adversarial agent (Section 3). In Section 4, we present the multi-agent reinforcement learning algorithms that enable us to find the optimal decision for calibrating and falsifying constitutive laws. In the numerical examples presented in Section 5, we report the performances of the agents playing three non-cooperative games, two for different classical elasto-plasticity models proposed by human experts for bulk granular materials, and one for a neural network traction–separation law of a granular interface.

As for notations and symbols, bold-faced letters denote tensors (including vectors which are rank-one tensors); the symbol '$\cdot$' denotes a single contraction of adjacent indices of two tensors (e.g. $\boldsymbol{a}\cdot\boldsymbol{b}=a_ib_i$ or $\boldsymbol{c}\cdot\boldsymbol{d}=c_{ij}d_{jk}$); the symbol ':' denotes a double contraction of adjacent indices of tensor of rank two or higher (e.g. $\boldsymbol{C}:\boldsymbol{\epsilon}^e=C_{ijkl}\epsilon^e_{kl}$); the symbol '$\otimes$' denotes a juxtaposition of two vectors (e.g. $\boldsymbol{a}\otimes\boldsymbol{b}=a_ib_j$) or two symmetric second-order tensors (e.g. $(\boldsymbol{\alpha}\otimes\boldsymbol{\beta})_{ijkl}=\alpha_{ij}\beta_{kl}$). Moreover, $(\boldsymbol{\alpha}\oplus\boldsymbol{\beta})_{ijkl}=\alpha_{jl}\beta_{ik}$ and $(\boldsymbol{\alpha}\ominus\boldsymbol{\beta})_{ijkl}=\alpha_{il}\beta_{jk}$. We also define identity tensors $(\boldsymbol{I})_{ij}=\delta_{ij}$, $(\boldsymbol{I}^4)_{ijkl}=\delta_{ik}\delta_{jl}$, and $(\boldsymbol{I}^4_{\text{sym}})_{ijkl}=\frac{1}{2}(\delta_{ik}\delta_{jl}+\delta_{il}\delta_{kj})$, where δ_{ij} is the Kronecker delta.

2. AI-designed experiments: selecting paths in arborescences of decisions

Traditionally the decisions on which experiments to conduct are based on a combination of intuition, knowledge and experience of the experimentalist. We make the first effort to use AI to assist the decision-makings of experimentalists on how to get data that can efficiently calibrate and falsify a constitutive model. Our method differs from the existing machine learning techniques that we formulate the experimentalist-model-critic environment as Markov games via decision-trees. In this game, the generation of calibration data is handled by a protagonist agent, once the model is calibrated, the testing data are generated by an adversary agent to evaluate the forward prediction accuracy. The goal of the adversary is to identify all application scenarios that the model will fail according to a user-defined objective function (falsification). Hence the validation will be simultaneously achieved: the model is valid within the calibration scenarios picked by the protagonist and the testing scenarios that the adversary has not picked. Practically, the model is safe to use unless the adversary "warns" that the model is at high risk. The formalization of decisions (or actions) as decision-trees, along with the communication mechanism designed in the game, enable AI agents to play this game competitively instead of human players.

Here we idealize the process of designing or planning an experiment as a sequence of decision making among different available options. All the available options and choices in the design process considered by the AI experimentalists (protagonist and adversary) are modeled by "arborescences" in graph theory with labeled vertices and edges. An arborescence is a rooted polytree in which, for a single root vertex u and any other vertex v, there exists one unique directed path from u to v. A polytree (or directed tree) is a directed graph whose underlying graph is a singly connected acyclic graph. A brief review of the essential terminologies are given in [38], and their detailed definitions can be found in, for instance, Graham et al. [39], West et al. [40] and Bang-Jensen and Gutin [41]. Mathematically, the arborescence for decision making (referred to as "decision tree" hereafter) can be

expressed as an 8-tuple $\mathbb{G} = (\mathbb{L}_\mathbb{V}, \mathbb{L}_\mathbb{E}, \mathbb{V}, \mathbb{E}, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{n_V}, \boldsymbol{n_E})$ where $\mathbb{V}$ and $\mathbb{E}$ are the sets of vertices and edges, $\mathbb{L}_\mathbb{V}$ and $\mathbb{L}_\mathbb{E}$ are the sets of labels for the vertices and edges, $\boldsymbol{s} : \mathbb{E} \to \mathbb{V}$ and $\boldsymbol{t} : \mathbb{E} \to \mathbb{V}$ are the mappings that map each edge to its source vertex and its target vertex, $\boldsymbol{n_V} : \mathbb{V} \to \mathbb{L}_\mathbb{V}$ and $\boldsymbol{n_E} : \mathbb{E} \to \mathbb{L}_\mathbb{E}$ are the mappings that give the vertices and edges their corresponding labels (names) in $\mathbb{L}_\mathbb{V}$ and $\mathbb{L}_\mathbb{E}$.

The decision trees are constructed based on a hierarchical series of test conditions (e.g., category of test, pressure level, target strain level) that an experimentalist must decide to complete the design of an experiment. Assuming that an experiment can be completely and uniquely defined by an ordered list of selected test conditions $tc = [tc_1, tc_2, tc_3, \ldots, tc_n]$, where $N_{\mathbb{TC}}$ is the total number of test conditions. Each tc_i is selected from a finite set of choices $\mathbb{TC}_i = \{tc_i^1, tc_i^2, tc_i^3, \ldots, tc_i^{m_i}\}$, where m_i is the number of choices for the ith test condition. For test conditions with inherently continuous design variables, $\mathbb{TC}_i$ can include preset discrete values. For example, the target strain for a loading can be chosen from discrete values of 1%, 2%, 3%, etc. All design choices available to experimentalists are represented by an ordered list of sets $\mathbb{TC} = [\mathbb{TC}_1, \mathbb{TC}_2, \mathbb{TC}_3, \ldots, \mathbb{TC}_n]$ with a hierarchical relationship such that, if $i < j$, $tc_i \in \mathbb{TC}_i$ must be selected prior to the selection of $tc_j \in \mathbb{TC}_j$.

After the construction of $\mathbb{TC}$ for experimentalists, a decision tree is built top-down from a root node representing the 'Null' state that no test condition is decided. The root node is split into m_1 subnodes according to the first level of decisions $\mathbb{TC}_1$. Each subnode is further split into m_2 subnodes according to the second level of decisions $\mathbb{TC}_2$. The splitting process on the subnodes is carried out recursively for all the $N_{\mathbb{TC}}$ levels of decisions in $\mathbb{TC}$. Finally, the down-most leaf nodes represent all possible combinations of test conditions. The maximum number of possible configurations of experiments is $N_{test}^{max} = \prod_{i=1}^{N_{\mathbb{TC}}} m_i$, when all decisions across $\mathbb{TC}_i$ are independent. The number of possible experiments is reduced ($N_{test} < N_{test}^{max}$) when restrictions are specified for the selections of test conditions. E.g., the selection of $tc_i \in \mathbb{TC}_i$ may prohibit the selections of certain choices tc_j in subsequent test conditions $\mathbb{TC}_j$, $j > i$. The experimentalists can choose multiple experiments by taking multiple paths in the decision tree from the root node to the leaf nodes. The total number of possible combination of paths, if the maximum allowed number of simultaneously chosen paths is N_{path}^{max}, is $\sum_{k=1}^{N_{path}^{max}} C_{N_{test}}^k$, where $C_{N_{test}}^k = \frac{N_{test}!}{k!(N_{test}-k)!}$ is the combination number.

Example for Hierarchical Test Conditions and Experimental Decision Tree. Consider a simple design of mechanical experiments for geomaterials, for which all choices are listed in

$$\mathbb{TC} = [\text{'Sample'},\ \text{'Type'},\ \text{'Target'}]. \tag{1}$$

The first decision is to pick the initial geomaterial sample to test. Assuming that a sample is fully characterized by its initial pressure p_0, a simple set of discrete sample choices is given as

$$\mathbb{TC}_1 = \text{'Sample'} = \{\text{'300 kPa'},\ \text{'400 kPa'}\}. \tag{2}$$

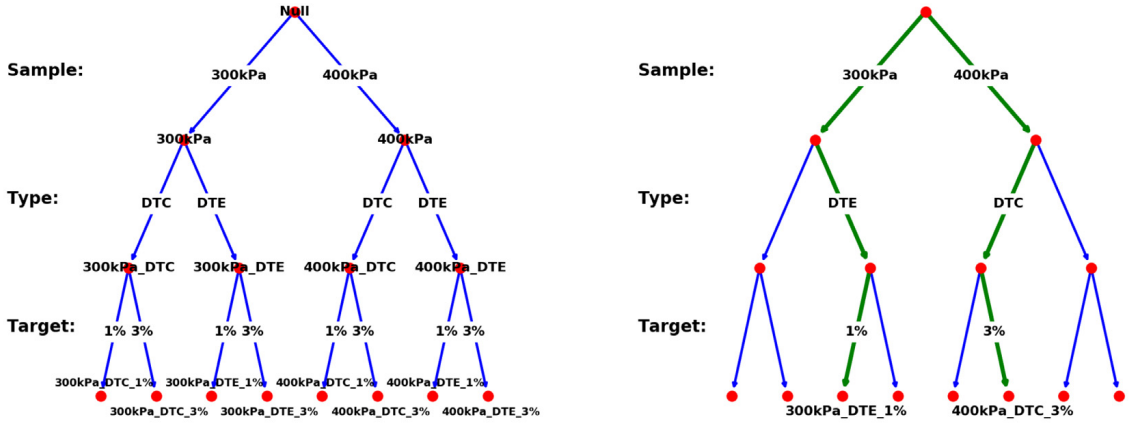
The second test condition is the type of the experiment. The experiment can be either drained triaxial compression test ('DTC') or drained triaxial extension test ('DTE'). Then

$$\mathbb{TC}_2 = \text{'Type'} = \{\text{'DTC'},\ \text{'DTE'}\}. \tag{3}$$

The third test condition to decide is the target strain magnitude for the loading. For example,

$$\mathbb{TC}_3 = \text{'Target'} = \{\text{'1\%'},\ \text{'3\%'}\}. \tag{4}$$

After all three decisions are sequentially made (taking a path in the decision tree), the experiment is completely determined by an ordered list, e.g., $tc = [\text{'300 kPa'},\ \text{'DTE'},\ \text{'3\%'}]$. It indicates that the AI experimentalist decides to perform a monotonic drained triaxial extension test on a sample with $p_0 = 300$ kPa until the axial strain reaches 3%.



(a) Decision tree with labels of vertices and edges

(b) Example of paths in the decision tree, the selected tests are '300kPa_DTE_1%' and '400kPa_DTC_3%'

Fig. 1. Decision tree for a simple experimental design for geomaterials (Eqs. (1), (2), (3), (4)).

The decision tree $\mathbb{G}$ for the hierarchical design of geomaterial experiments specified by Eqs. (1), (2), (3), (4) is shown in Fig. 1(a). The vertex sets and edge sets of the graph are

$$
\begin{aligned}
\mathbb{V} =& \{\text{'Null'},\ \text{'300 kPa'},\ \text{'400 kPa'},\ \text{'300 kPa\_DTC'},\ \text{'300 kPa\_DTE'},\ \text{'400 kPa\_DTC'},\ \text{'400 kPa\_DTE'},\\
& \text{'300 kPa\_DTC\_1\%'},\ \text{'300 kPa\_DTC\_3\%'},\ \text{'300 kPa\_DTE\_1\%'},\ \text{'300 kPa\_DTE\_3\%'},\\
& \text{'400 kPa\_DTC\_1\%'},\ \text{'400 kPa\_DTC\_3\%'},\ \text{'400 kPa\_DTE\_1\%'},\ \text{'400 kPa\_DTE\_3\%'}\},\\
\mathbb{E} =& \{\text{'Null'} \rightarrow \text{'300 kPa'},\ \text{'Null'} \rightarrow \text{'400 kPa'},\ \text{'300 kPa'} \rightarrow \text{'300 kPa\_DTC'},\\
& \text{'300 kPa'} \rightarrow \text{'300 kPa\_DTE'},\ \text{'400 kPa'} \rightarrow \text{'400 kPa\_DTC'},\ \text{'400 kPa'} \rightarrow \text{'400 kPa\_DTE'},\\
& \text{'300 kPa\_DTC'} \rightarrow \text{'300 kPa\_DTC\_1\%'},\ \text{'300 kPa\_DTC'} \rightarrow \text{'300 kPa\_DTC\_3\%'},\\
& \text{'300 kPa\_DTE'} \rightarrow \text{'300 kPa\_DTE\_1\%'},\ \text{'300 kPa\_DTE'} \rightarrow \text{'300 kPa\_DTE\_3\%'},\\
& \text{'400 kPa\_DTC'} \rightarrow \text{'400 kPa\_DTC\_1\%'},\ \text{'400 kPa\_DTC'} \rightarrow \text{'400 kPa\_DTC\_3\%'},\\
& \text{'400 kPa\_DTE'} \rightarrow \text{'400 kPa\_DTE\_1\%'},\ \text{'400 kPa\_DTE'} \rightarrow \text{'400 kPa\_DTE\_3\%'}\},\\
\mathbb{L}_{\mathbb{V}} =& \mathbb{V},\\
\mathbb{L}_{\mathbb{E}} =& \{\text{'300 kPa'},\ \text{'400 kPa'},\ \text{'DTC'},\ \text{'DTE'},\ \text{'1\%'},\ \text{'3\%'}\}.
\end{aligned}
\tag{5}
$$

In this example, $N_{test} = N_{test}^{max} = 2*2*2 = 8$. If an experimentalist only collects data from one or two experiments, i.e., $N_{path}^{max} = 2$, the total number of possible combinations is $C_8^1+C_8^2 = 36$. Fig. 1(b) presents two example paths with edge labels illustrating the hierarchical decisions on the test conditions in order to arrive at the final experimental designs '300 kPa_DTE_1%' and '400 kPa_DTC_3%'. □

In this section, we present two decision trees for the design of geomechanical experiments, one for the bulk mechanical behavior of granular materials, another for the traction–separation behavior of granular interfaces. We later study the intelligence of the reinforcement-learning-based experimentalists (protagonist and adversary) on these decision trees in Section 5.

2.1. Decision tree for AI-guided experimentation on bulk granular materials

This section defines a representative decision tree for the AI-guided experimentation on bulk geomaterials. The hierarchical series of test conditions includes six elements, $\mathbb{TC} = [\mathbb{TC}_1, \mathbb{TC}_2, \mathbb{TC}_3, \mathbb{TC}_4, \mathbb{TC}_5, \mathbb{TC}_6]$, such that the

Table 1
Choices of test conditions for AI-guided experimentation on bulk granular materials.

$\mathbb{TC}$ Test conditions	Choices
$\mathbb{TC}_1$ = 'Sample p_0'	{'300 kPa', '400 kPa', '500 kPa'}
$\mathbb{TC}_2$ = 'Sample e_0'	{'0.60', '0.55'}
$\mathbb{TC}_3$ = 'Type'	{'DTC', 'DTE', 'TTC'}
$\mathbb{TC}_4$ = 'Load target'	{'3%', '5%'}
$\mathbb{TC}_5$ = 'Unload target'	{'NaN', '0%', '3%'}
$\mathbb{TC}_6$ = 'Reload target'	{'NaN', '3%', '5%'}

AI experimentalists can choose isotropic granular samples of different initial pressure p_0 and initial void ratio e_0, perform different drained triaxial tests, and design different loading–unloading–reloading paths.

The choices for each test condition are shown in Table 1, represented by decision labels. The decision labels for the test types $\mathbb{TC}_3$ are defined as follows,

1. 'DTC': drained conventional triaxial compression test ($\dot{\epsilon}_{11} < 0$, $\dot{\sigma}_{22} = \dot{\sigma}_{33} = \dot{\sigma}_{12} = \dot{\sigma}_{23} = \dot{\sigma}_{13} = 0$),
2. 'DTE': drained conventional triaxial extension test ($\dot{\epsilon}_{11} > 0$, $\dot{\sigma}_{22} = \dot{\sigma}_{33} = \dot{\sigma}_{12} = \dot{\sigma}_{23} = \dot{\sigma}_{13} = 0$),
3. 'TTC': drained true triaxial test with $b = 0.5$ ($\dot{\epsilon}_{11} < 0$, $b = \frac{\sigma_{22}-\sigma_{33}}{\sigma_{11}-\sigma_{33}} = const$, $\dot{\sigma}_{33} = \dot{\sigma}_{12} = \dot{\sigma}_{23} = \dot{\sigma}_{13} = 0$),

with the loading conditions represented by constraints on the components of the stress rate and strain rate tensors

$$\dot{\boldsymbol{\epsilon}} = \begin{bmatrix} \dot{\epsilon}_{11} & \dot{\epsilon}_{12} & \dot{\epsilon}_{13} \\ & \dot{\epsilon}_{22} & \dot{\epsilon}_{23} \\ \text{sym} & & \dot{\epsilon}_{33} \end{bmatrix}, \quad \dot{\boldsymbol{\sigma}} = \begin{bmatrix} \dot{\sigma}_{11} & \dot{\sigma}_{12} & \dot{\sigma}_{13} \\ & \dot{\sigma}_{22} & \dot{\sigma}_{23} \\ \text{sym} & & \dot{\sigma}_{33} \end{bmatrix}. \tag{6}$$

Since 'DTC' and 'DTE' are special cases of true triaxial tests, the choices {'DTC', 'DTE', 'TTC'} for $\mathbb{TC}_3$ are equivalent to choosing the value of $b = \frac{\sigma_{22}-\sigma_{33}}{\sigma_{11}-\sigma_{33}}$ from {'0.0', '1.0', '0.5'}, respectively [42].

The decision labels 'NaN' in $\mathbb{TC}_5$ and $\mathbb{TC}_6$ indicate that the unloading or reloading is not activated. This design enables the freedom of generating monotonic loading paths (e.g., '5%_NaN_NaN'), loading–unloading paths (e.g., '5%_0%_NaN') and loading–unloading–reloading paths (e.g., '5%_0%_3%'). There are restrictions in choosing the strain targets. The experimentalist picks the loading target in $\mathbb{TC}_4$ first and the unloading target in $\mathbb{TC}_5$ must be, if not 'NaN' (stop the experiment), smaller than the loading strain. Then the reloading target in $\mathbb{TC}_6$ must be, if not 'NaN', larger than the unloading strain.

The corresponding decision tree is shown in Fig. 2. The subtree concerning the restricted decision-making in $\mathbb{TC}_4$, $\mathbb{TC}_5$, and $\mathbb{TC}_6$ is also detailed in the figure. The total number of experimental designs (which equals to the number of leaf nodes in the tree) is $N_{test} = 180$. Fig. 3 provides the experimental settings on DEM (discrete element methods) numerical specimens and data from one example of the experiments. The total number of experimental data combinations increases significantly when the maximum allowed simultaneous paths N_{path}^{max} increases. The combination number equals to $C_{180}^1 = 180$ when $N_{path}^{max} = 1$, equals to $C_{180}^1 + C_{180}^2 = 16\,290$ when $N_{path}^{max} = 2$, equals to $C_{180}^1 + C_{180}^2 + C_{180}^3 = 972\,150$ when $N_{path}^{max} = 3$, etc.

2.2. Decision tree for AI-guided experimentation on granular interfaces

This section defines a representative decision tree for the AI-guided experimentation on granular interfaces. The hierarchical series of test conditions includes six elements, $\mathbb{TC}$ = [$\mathbb{TC}_1$, $\mathbb{TC}_2$, $\mathbb{TC}_3$, $\mathbb{TC}_4$, $\mathbb{TC}_5$, $\mathbb{TC}_6$], such that the AI experimentalists can choose the direction of the prescribed displacement jump, the number of loading cycles, and different target displacement values to design complex loading paths.

The choices for each test condition are shown in Table 2, represented by decision labels. 'NormTangAngle' represents the angle between the displacement jump vector and the tangential direction vector, the corresponding values in the Choices column are in units of degree. 'NumCycle' represents the number of loading–unloading cycles. The conditions 'Target1', 'Target2', 'Target3', 'Target4' represent the target displacement jump magnitudes along the loading–unloading cycles, the corresponding values in the Choices column are in units of millimeters. Regardless of the loading–unloading cycles, the final displacement jump reaches the magnitude of 0.4 mm. The decision label

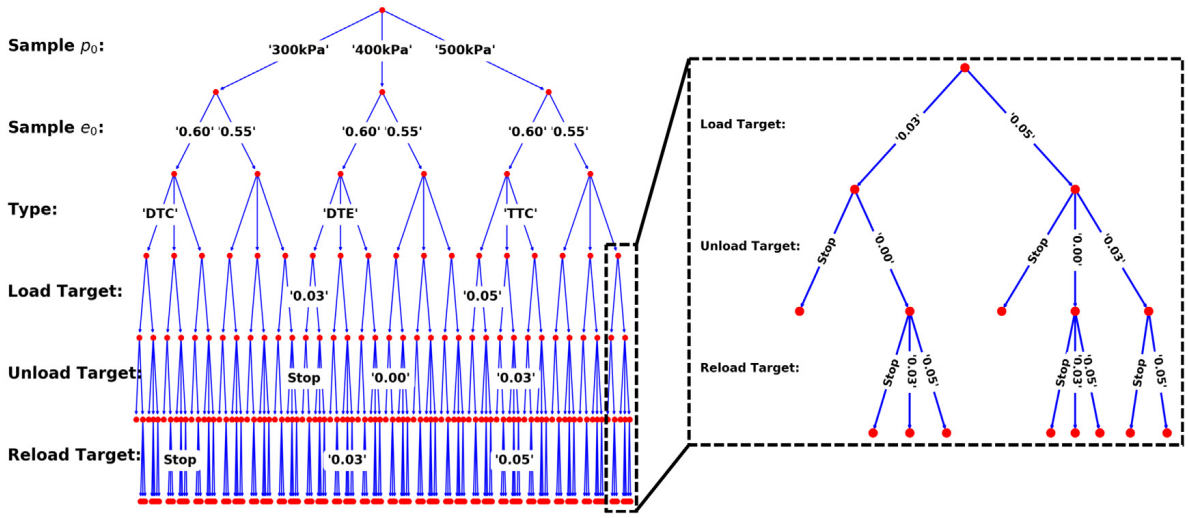


Fig. 2. Decision tree for AI-guided drained true triaxial tests on bulk granular materials. Due to the complexity of the graph, the vertex labels are omitted, and only a few edge labels are shown. See Fig. 1 for exhaustive vertex and edge labels in a simple decision tree example.

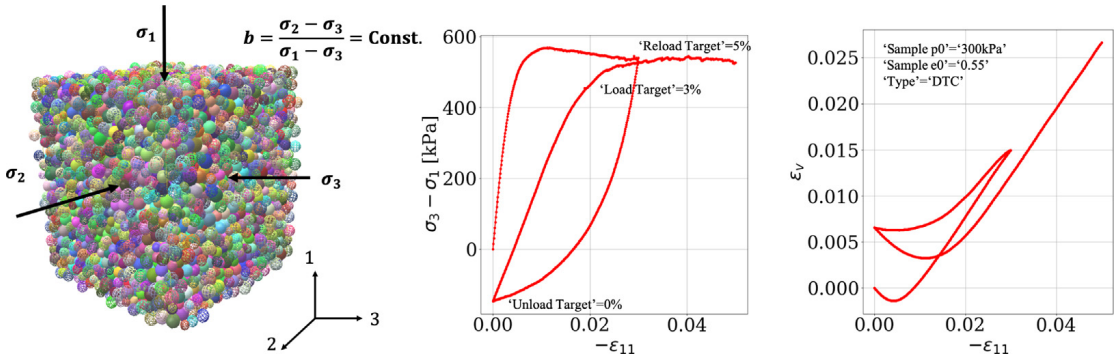


Fig. 3. Experimental settings of drained true triaxial tests on numerical specimens of bulk granular materials simulated via DEM (discrete element methods). The test conditions for the AI experimentalist are presented in Table 1. As an example, the differential stress data and volumetric strain data obtained from a test designed by the decision tree path '300 kPa' → '0.55' → 'DTC' → '3%' → '0%' → '5%' are presented.

Table 2
Choices of test conditions for AI-guided experimentation on granular interfaces.

$\mathbb{TC}$ test conditions	Choices
$\mathbb{TC}_1$ = 'NormTangAngle'	{'0', '15', '30', '45', '60', '75'}
$\mathbb{TC}_2$ = 'NumCycle'	{'0', '1', '2'}
$\mathbb{TC}_3$ = 'Target1'	{'NaN', '0.1', '0.2', '0.3'}
$\mathbb{TC}_4$ = 'Target2'	{'NaN', '0.0', '0.1', '0.2'}
$\mathbb{TC}_5$ = 'Target3'	{'NaN', '0.1', '0.2', '0.3'}
$\mathbb{TC}_6$ = 'Target4'	{'NaN', '0.0', '0.1', '0.2'}

'NaN' indicates that the unloading or reloading is not activated. For example, 'NumCycle'='0' means a monotonic loading to 0.4 mm, hence all the target conditions should adopt the values of 'NaN'; 'NumCycle'='1' means a loading–unloading–reloading path to 0.4 mm, hence 'Target1' (loading target) and 'Target2' (unloading target) can adopt values within '0.0', '0.1', '0.2', '0.3', while 'Target3' and 'Target4' should be 'NaN's.

The corresponding decision tree is shown in Fig. 4. The total number of experimental designs (which equals to the number of leaf nodes in the tree) is $N_{test} = 228$. Fig. 5 provides the experimental settings on DEM (discrete element methods) numerical specimens and data from one example of the experiments. The total number of experimental

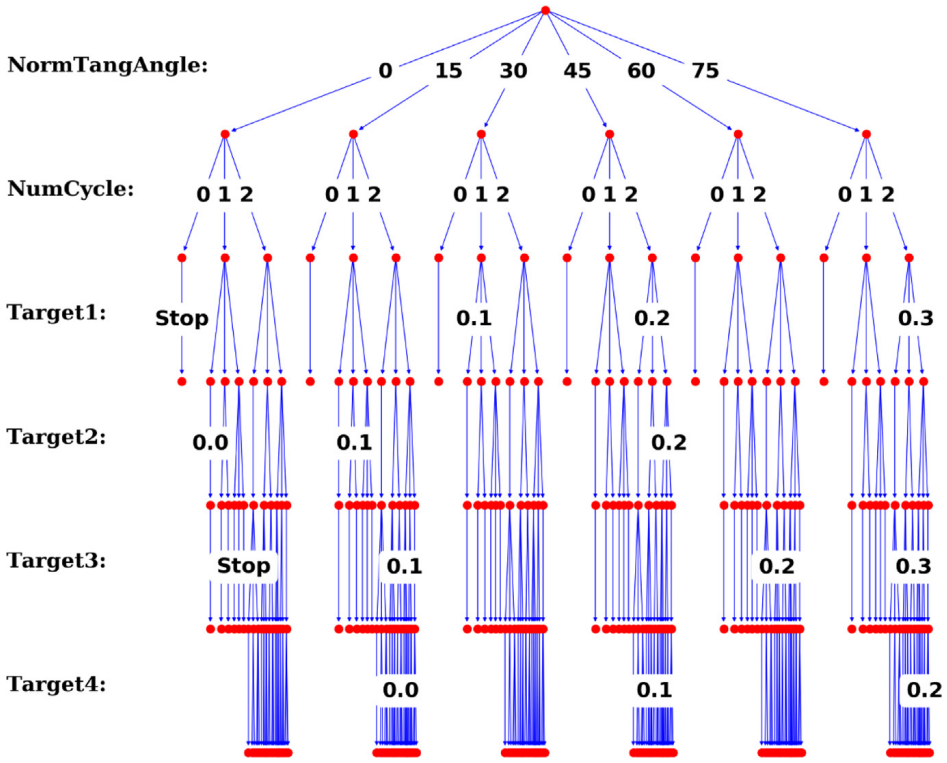


Fig. 4. Decision tree for AI-guided displacement-driven mixed-mode shear tests on granular interfaces. Due to the complexity of the graph, the vertex labels are omitted, and only a few edge labels are shown.

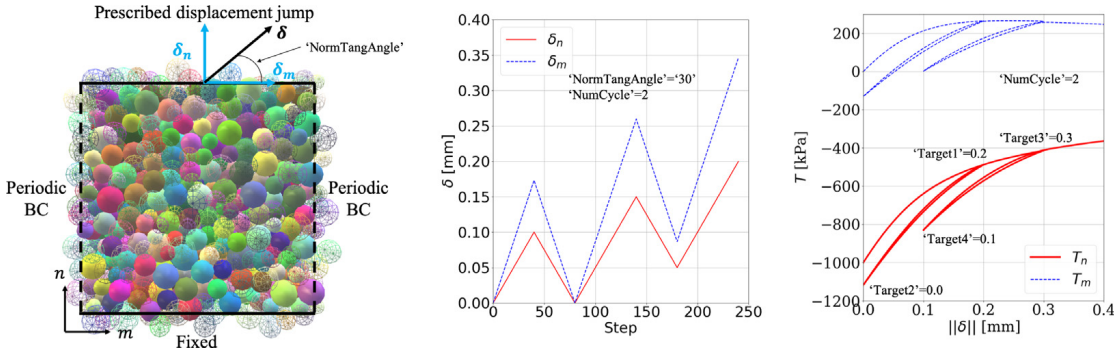


Fig. 5. Experimental settings of displacement-driven mixed-mode shear tests on numerical specimen of granular interfaces using DEM (discrete element methods). The test conditions for the AI experimentalist are presented in Table 2. As an example, the loading path and traction in normal and tangential directions obtained from a test designed by the decision tree path '30' → '2' → '0.2' → '0.0' → '0.3' → '0.1' are presented. Regardless of the designed loading–unloading cycles, the final displacement jump reaches the magnitude of 0.4 mm.

data combinations, for example, equals to $C_{228}^{1} + C_{228}^{2} + C_{228}^{3} \approx 1.97e6$ when $N_{path}^{max} = 3$. Such number is already impractical for a human to find the optimal data sets for calibration and falsification by trial and error. For high efficiency, the decisions in performing experiments should be guided by experienced experts or, in this paper, reinforcement-learning-based AI.

3. Multi-agent non-cooperative game for model calibration/falsification with adversarial attacks

This section presents the design of a data acquisition game for both AI experimentalists (protagonist and adversary) to play, based on the decision trees defined in Section 2 involving the common actions in testing the mechanical properties of geomaterials. The goal of this game is to enable the protagonist agent to find the optimal

design of experiments that best calibrate a constitutive law, while having the adversary agent designs a counterpart set of experiments that expose the weakness of the models in the same decision tree that represents the application range. For simplicity, we assume that all experiments conducted by both agents are fully reproducible and free of noise. We will introduce a more comprehensive treatment for more general situations in which the bias and sensitivity of the data as well as the possibility of erroneous and even fabricated data are considered in the future. Such a treatment is, nevertheless out of the scope of this work.

Multi-agent multi-objective Markov games [43] have been widely studied and applied in robotics [44], traffic control [45], social dilemmas [46], etc. In our previous work, Wang et al. [38], our focus was on designing agents that have different actions and states but share the same goal. In this work, our innovation is on designing a non-cooperative game in which the agents are competing against each other for a pair of opposite goals. While the reinforcement learning may lead to improved gameplay through repeated trial-and-error, the non-cooperative nature of this new game will force the protagonist to act differently in response to the weakness exposed by the adversary. This treatment therefore may lead to a more robust and regularized model. In this work, the protagonist and the adversary are given exactly the same set of possible actions mathematically represented in a decision tree. While a non-cooperative game with non-symmetric action spaces can enjoy the great performance as demonstrated in some of the OpenAI systems [44], such an extension is out of the scope of this study and will be considered in the future.

3.1. Non-cooperative calibration/falsification game involving protagonist and adversary

We follow the general setup in [44] to create a two-player Markov game with competing objectives to calibrate and falsify a constitutive model. Both calibration and falsification are idealized as procedures that involve sequences of actions taken to maximize (in the case of calibration) and minimize (in the case of the falsification) a metric that assesses the prediction accuracy and robustness.

Consider the Markov decision process (MDP) in this game expressed as a tuple $(\mathcal{S}, \mathcal{A}_p, \mathcal{A}_a, \mathcal{P}, r_p, r_a, s_0)$ where $\mathcal{S}$ is the set of game states and s_0 is the initial state distribution. $\mathcal{A}_p$ is the set of actions taken by the protagonist in charge of generating the experimental data to calibrate a given material model. $\mathcal{A}_a$ is the set of actions taken by the adversary in charge of falsifying the material model. $\mathcal{P} : \mathcal{S} \times \mathcal{A}_p \times \mathcal{A}_a \times \mathcal{S} \to \mathbb{R}$ is the transition probability density. $r_p : \mathcal{S} \times \mathcal{A}_p \times \mathcal{A}_a \to \mathbb{R}$ and $r_a : \mathcal{S} \times \mathcal{A}_p \times \mathcal{A}_a \to \mathbb{R}$ are the rewards of protagonist and adversary, respectively. If $r_p = r_a$, the game is fully cooperative. If $r_p = -r_a$, the game is zero-sum competitive. At the current state s of the game, if the protagonist is taking action a_p sampled from a stochastic policy μ_p and the adversary is taking action a_a sampled from a stochastic policy μ_a, the reward functions are $r_p^{\mu_p,\mu_a} = E_{a_p\sim\mu_p(\cdot|s),a_a\sim\mu_a(\cdot|s)}[r_p(s, a_p, a_a)]$ and $r_a^{\mu_p,\mu_a} = E_{a_p\sim\mu_p(\cdot|s),a_a\sim\mu_a(\cdot|s)}[r_a(s, a_p, a_a)]$.

In this work, all the possible actions of the protagonist and the adversary agent are mathematically represented by decision trees (Section 2). The protagonist first selects one or more paths in its own tree which provide the detailed experimental setups to generate calibration data for the material model, then the adversary selects one or more paths in its own tree (identical to the protagonist's tree) to generate test data for the calibrated model, aiming to find the worst prediction scenarios. The rewards are based on the prediction accuracy measures SCORE of the constitutive model against data. This measure of 'win' or 'lose' is only available when the game is terminated, similar to Chess and Go [35,36], thus the final rewards are back-propagated to inform all intermediate rewards $r_p(s, a_p, a_a)$ and $r_a(s, a_p, a_a)$. r_p is defined to encourage the increase of SCORE of model calibrations, while r_a is defined to favor the decrease of SCORE of forward predictions. In this setting, the game is non-cooperative, and generally not zero-sum.

3.2. Components of the game for the experimentalist agents

The agent–environment interactive system (game) for the experimentalist agents consists of the game environment, game states, game actions, game rules, and game rewards [47,48] (Fig. 6). These key ingredients are detailed as follows.

Game Environment consists of the geomaterial samples, the constitutive model for performance evaluation, and the experimental decision trees. The samples in this game are representative volume elements (RVEs) of virtual granular assemblies modeled by the discrete element method (DEM) (e.g., Figs. 3, 5). The preparation of such DEM RVEs are detailed in the numerical examples. The constitutive model can be given by the modeler agent

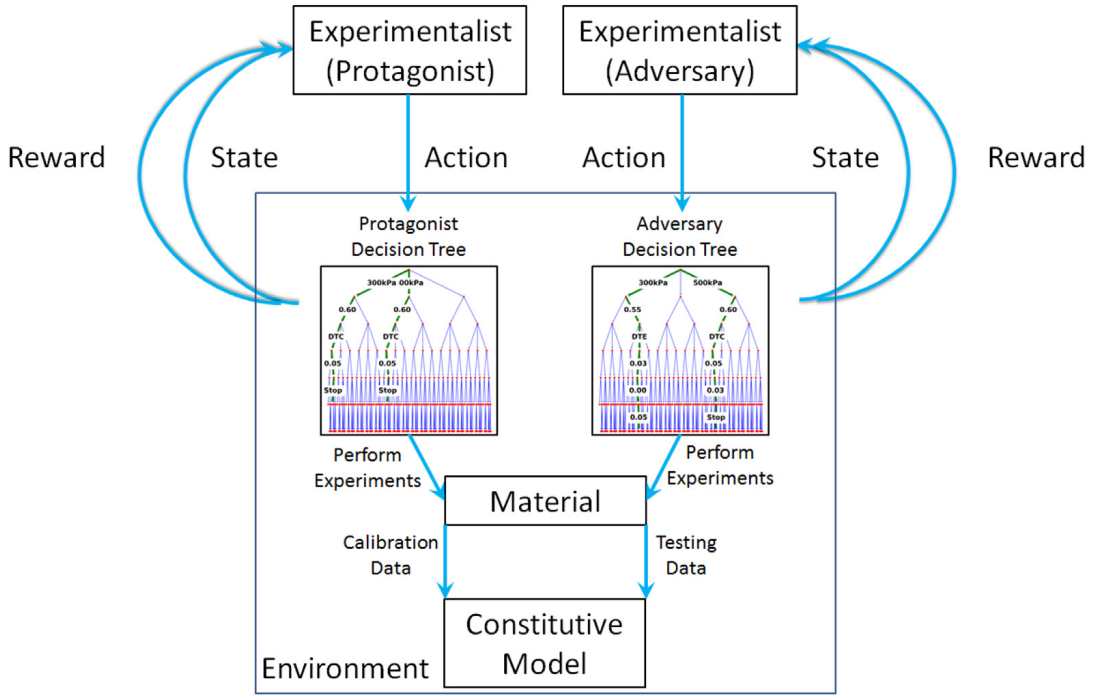


Fig. 6. Key ingredients (environment, agents, states, actions, rules, and rewards) of the two-player non-cooperative agent–environment interactive system (game) for the experimentalist agents.

in a meta-modeling game [38,48]. In this paper, we focus on the interactive learning of data acquisition strategies for a certain constitutive model of interest. A three-agent protagonist-modeler-adversary reinforcement learning game that incorporate modeling stratwgies is out of the scope of this study. The protagonist and adversary agents determine the experiments on the RVEs in order to collect data for model parameter identification and testing the forward prediction accuracy of the constitutive model, respectively, via taking paths in their own decision trees (e.g., Figs. 1, 2, 4).

Game State For the convenience of deep reinforcement learning using policy/value neural networks, we use a 2D array $s_{(2)}$ to concisely represent the paths that the protagonist or adversary has selected in the experimental decision tree. The mapping from the set of the 2D arrays to the set of path combinations in the decision tree is injective. The array has a row size of N_{path}^{max} and a column size of $N_{\mathbb{TC}}$. Each row represents one path in the decision tree from the root node to a leaf node, i.e., a complete design of one experiment. The number of allowed experiments is restricted by the row size N_{path}^{max}, which is defined by the user. Each array entry in the $N_{\mathbb{TC}}$ columns represents the selected decision label of each test condition in $\mathbb{TC}$. The entry a (integer) in the jth row and ith column indicates that the ath decision label in the set $\mathbb{TC}_i$ is selected for the jth experiment. Before the decision tree selections, the agent first decides a 1D array $s_{(1)}$ of size N_{path}^{max}, with its kth entry indicating whether the agent decides to take a new path in the decision tree (i.e., perform another experiment) after the current kth experiment is done. A value of 1 indicates continuation and 2 indicates stop. The total state s of the game combines $s_{(1)}$ and $s_{(2)}$, with $s_{(2)}$ flattened to a 1D array of size $N_{path}^{max} * N_{\mathbb{TC}}$ and then input into the policy/value neural networks for policy evaluations. Initially, all entries in the arrays are 0, indicating no decision has been made.

Game Action The AI agent works on the game state arrays by changing the initial zero entries into integers representing the decision labels. The agent firstly selects 1 for continuation or selects 2 for stop in $s_{(1)}$, in the left-to-right order. The agent then works on $s_{(2)}$ in the left-to-right then top-to-bottom order. Suppose that the first zero element of the current state array $s_{(2)}$ is in the jth row and ith column, the agent will select an integer $1 \leq a \leq m_i$ (number of choices) to choose a decision label in $\mathbb{TC}_i$. The size of the action space is $N_{action} = \max_{i \in [1, N_{\mathbb{TC}}]} m_i$.

Game Rule The AI agents are restricted to follow existing edges in the constructed decision tree, which has already incorporated decision limitations such as the choices of loading/unloading/reloading strain targets. The game rules are reflected by a list of N_{action} binaries $LegalActions(s) = [ii_1, ii_2, \ldots, ii_{N_{action}}]$ at the current state s. If the

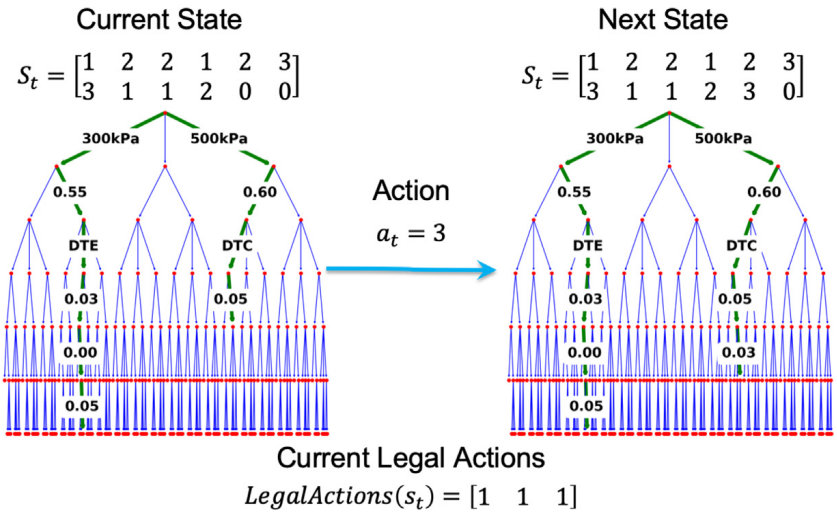


Fig. 7. Example of the current s_t and next s_{t+1} game states describing the selected edges in the decision tree, action by the agent a_t to "advance" in the decision tree, and the legal actions at the current state, with $N_{path}^{max} = 2$.

a-th decision is allowed, the ath entry is 1. Otherwise, the entry is 0. Fig. 7 provides an example of the mathematical representations of the game states, actions, and rules of the decision tree game.

Game Reward The rewards from the game environment to the experimentalists should consider the performance of a given constitutive model on calibration data and testing data. After the decision of experiments by the protagonist, these experiments are performed on material samples to collect data. Then the constitutive model is calibrated with these data, and the accuracy is evaluated by a model score $\text{SCORE}_{\text{protagonist}}$. After the decision of experiments by the adversary, the calibrated constitutive model gives forward predictions on these testing data. The accuracy is evaluated by a model score $\text{SCORE}_{\text{adversary}}$. $+\text{SCORE}_{\text{protagonist}}$ is returned to the protagonist to inform its game reward, while $-\text{SCORE}_{\text{adversary}}$ is returned to the adversary. This adversary attack reward system is the key to ensure that the protagonist generates calibration data to maximize the prediction strength of the constitutive model, while the adversary tries to explore the weakness of the model.

3.3. Evaluation of model scores and game rewards

The accuracy of model calibrations and forward predictions are quantified by calculating the discrepancy between the vector of data points $[\overline{Y}_i^{\text{data}}]_{i=1}^{N_{data}}$ and the vector of predicted values $[\overline{Y}_i^{\text{model}}]_{i=1}^{N_{data}}$ under the same experimental conditions. For both data points and predictions, $\overline{Y}_i = \mathcal{S}_j(Y_i^j)$, where Y_i^j is the data that falls into the jth category of output features (quantities of interest, such as deviatoric stress q and void ratio e). $\mathcal{S}_j$ is the scaling operator (standardization, min–max scaling, ...) for the jth output feature.

The predictions $[\overline{Y}_i^{\text{model}}]_{i=1}^{N_{data}}$ come from a given constitutive model that is calibrated with data generated by the protagonist. In this work, for elasto-plastic models, the nonlinear least-squares solver "NL2SOL" in Dakota software [49] is used to find the optimal parameter values. The initial guess, upper and lower bounds of each parameter are given by domain experts' knowledge and preliminary estimations. For models of artificial neural networks, parameters are tuned by backpropagation algorithms using Tensorflow [50]. In both cases, the optimal material parameters minimize the scaled mean squared error objective function

$$\text{scaled MSE} = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} (\overline{Y}_i^{\text{model}} - \overline{Y}_i^{\text{data}})^2 . \tag{7}$$

The predictions $\overline{Y}_i^{\text{model}}$ from a calibrated constitutive model are compared against the output data $\overline{Y}_i^{\text{data}}$ corresponding to the input loading paths $\overline{X}_i^{\text{data}}$ for both calibration data and testing data. The model scores measuring

the prediction accuracy are based on the modified Nash–Sutcliffe efficiency index [51–53],

$$E_{NS}^{j} = 1 - \frac{\sum_{i=1}^{N_{data}} |\overline{Y}_i^{\text{data}} - \overline{Y}_i^{\text{model}}|^j}{\sum_{i=1}^{N_{data}} |\overline{Y}_i^{\text{data}} - \text{mean}(\overline{Y}^{\text{data}})|^j} \in (-\infty, 1.0]. \tag{8}$$

When $j = 2$, it recovers the conventional Nash–Sutcliffe efficiency index. Both the original and modified Nash–Sutcliffe efficiency indexes are both commonly used to assess the predictive power of various hydrological models [54]. They both range from $-\inf$ to 1. An efficiency of 1 ($E_{NS}^{j} = 1$) corresponds to a perfect match of model predictions to the observed data. An efficiency of 0 ($E_{NS}^{j} = 0$) indicates that the model predictions are as accurate as the mean of the observed data, whereas an efficiency less than zero ($E_{NS}^{j} < 0$) occurs when the observed mean is a better predictor than the model itself. Here we adopt $j = 1$, and

$$\text{SCORE}_{\text{protagonist or adversary}} = 2 * \frac{\min(\max(E_{NS}^{1}, E_{NS}^{min}), E_{NS}^{max}) - 0.5 * (E_{NS}^{min} + E_{NS}^{max})}{E_{NS}^{max} - E_{NS}^{min}}, \tag{9}$$

where E_{NS}^{max} and E_{NS}^{min} are maximum and minimum cutoff values of the modified Nash–Sutcliffe efficiency index, $\text{SCORE} \in [-1.0, 1.0]$.

The game reward returned to the protagonist can consider both the calibration accuracy and the forward prediction accuracy, by including an exponential decay term:

$$\text{Reward}_{\text{protagonist}} = -1 + (\text{SCORE}_{\text{protagonist}} + 1) * \exp\left[-\alpha_{\text{SCORE}} * \max(E_{NS}^{min} - \min(\{E_{NS}^{1}\}), 0)\right]. \tag{10}$$

where $\min(\{E_{NS}^{1}\})$ is the minimum N–S index observed in the gameplay history, α_{SCORE} is a user-defined decay coefficient. When $\min(\{E_{NS}^{1}\}) < E_{NS}^{min}$, the decay term starts to drop the reward of the protagonist, otherwise $\text{Reward}_{\text{protagonist}} = +\text{SCORE}_{\text{protagonist}}$. On the other hand, the game reward returned to the adversary is

$$\text{Reward}_{\text{adversary}} = -\text{SCORE}_{\text{adversary}}. \tag{11}$$

Since the adversary is rewarded at the expense of the protagonist's failure, it is progressively learning to create increasingly devastating experimental designs to falsify the model, thus forcing the protagonist to calibrate material models that are robust to any disturbances created by the adversary. In this work, we refer to the move of the protagonist as calibration or defense, while the move of the adversary as falsification or attack.

4. Parallel reinforcement learning algorithm for the non-cooperative experimental/adversarial game

In the language of game theory, the meta-modeling game defined in the previous section is categorized as non-cooperative, asymmetric (the payoff of a particular strategy depends on whether protagonist or adversary is playing), non-zero-sum, sequential (the adversary is aware of the protagonist's strategy in order to attack accordingly), imperfect information (the protagonist does not know how the adversary will attack). Let $(\mathcal{M}, \mathcal{R})$ be a representation of this two-player (denoted by subscripts p and a) non-cooperative game, with $\mathcal{M} = \mathcal{M}_p \times \mathcal{M}_a$ the set of strategy profiles. $\mathcal{R}(\mu) = (\mathcal{R}_p(\mu), \mathcal{R}_a(\mu))$ is the payoff (final reward) function evaluated at a strategy profile $\mu = (\mu_p, \mu_a) \in \mathcal{M}$. A strategy profile μ^* is a Nash equilibrium if no unilateral change in μ^* by any player is more profitable for that player, i.e.,

$$\begin{cases} \forall \mu_p \in \mathcal{M}_p, \ \mathcal{R}_p((\mu_p^*, \mu_a^*)) \geq \mathcal{R}_p((\mu_p, \mu_a^*)) \\ \forall \mu_a \in \mathcal{M}_a, \ \mathcal{R}_a((\mu_p^*, \mu_a^*)) \geq \mathcal{R}_a((\mu_p^*, \mu_a)) \end{cases}. \tag{12}$$

The existence of at least one such equilibrium point is proven by Nash et al. [33].

Solving the optimization problem directly to find the Nash equilibria strategies for this complex game is prohibitive [55]. Instead, deep reinforcement learning (DRL) algorithm is employed. In this technique, the strategy of each player (μ_p or μ_a) is parameterized by an artificial neural network f_θ that takes in the description of the current state s of the game and outputs a policy vector $\boldsymbol{p}$ with each component representing the probability of taking actions from state s, as well as a scalar v for estimating the expected reward of the game from state s, i.e.,

$$(\boldsymbol{p}, v) = f_\theta(s). \tag{13}$$

These policy/value networks guide the learning optimal strategies of both protagonist and adversary such that their actions may maximize their corresponding final game rewards. The learning is completely free of human

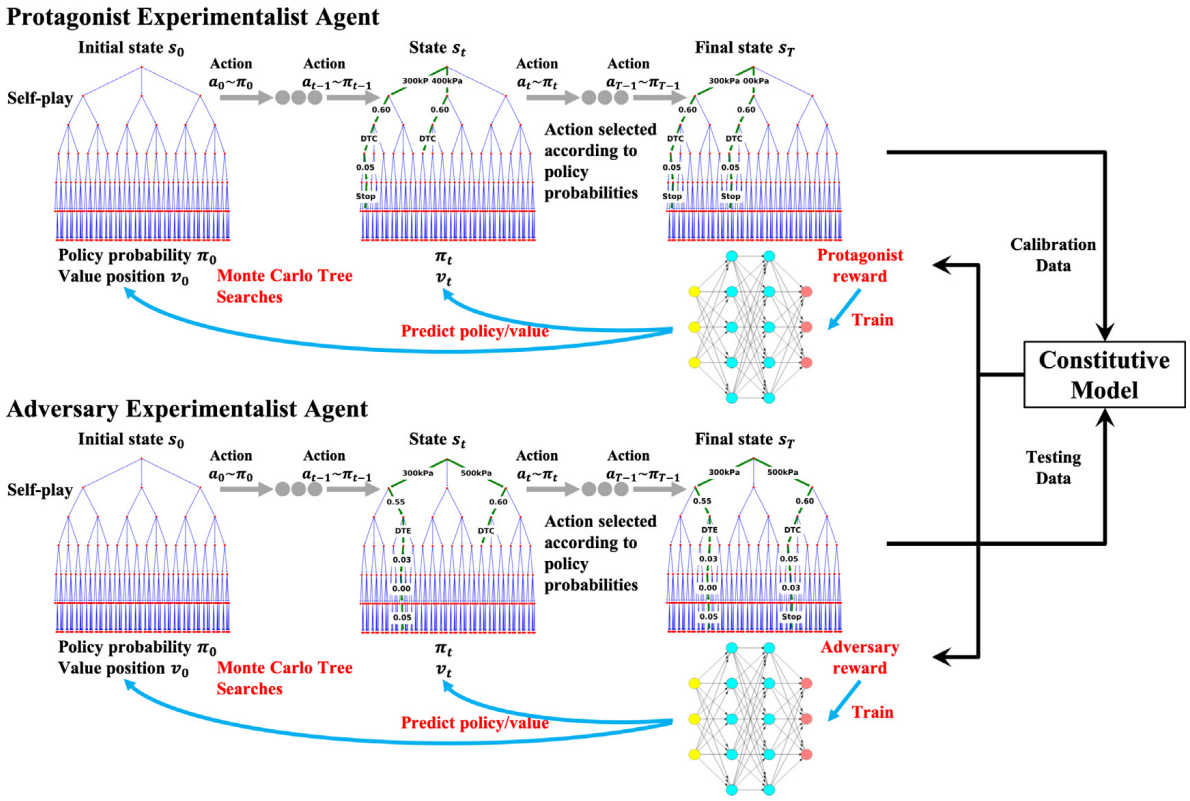


Fig. 8. Two-player adversarial reinforcement learning for generating optimal strategies to automate the calibration and falsification of a constitutive model.

interventions after the complete game settings. This tactic is considered one of the key ideas leading to the major breakthrough in AI playing the game of Go (AlphaGo Zero) [35], Chess and shogi (Alpha Zero) [36] and many other games. In [48], the key ingredients (policy/value network, upper confidence bound for Q-value, Monte Carlo Tree Search) of the DRL technique are detailed and applied to a meta-modeling game for modeler agent only, focusing on finding the optimal topology of physical relations from fixed training/testing datasets. Since DRL needs to figure out the optimal strategies for both agents, the algorithm is extended to multi-agent multi-objective DRL [56–58]. The AI for protagonist and adversary are improved simultaneously during the self-plays of the entire meta-modeling game, according to the individual rewards they receive from the game environment and the communications between themselves (see Fig. 8).

The pseudocode of the reinforcement learning algorithm to play the non-cooperative game is presented in Algorithm 1. This is an extension of the algorithm in [48]. As demonstrated in Algorithm 1, each complete DRL procedure involves $numIters$ number of training iterations and one final iteration for generating the converged selected paths in decision trees. Each iteration involves $numEpisodes$ number of game episodes that construct the training example set $trainExamples$ for the training of the policy/value networks $f_\theta^{\text{Protagonist}}$ and $f_\theta^{\text{Adversary}}$. For decision makings in each game episode, the action probabilities are estimated from $numMCTSSims$ runs of Monte Carlo Tree Search (MCTS) simulations.

The Monte Carlo Tree Search is an algorithm that estimates the optimal action out of a set of possible actions through four sequential processes (selection, expansion, simulation and back-propagation) [36,48,59]. In the selection step, the agents traverse the tree produced thus far. Each agent then selects an unexpanded node of the tree. Then, the successors of the states chosen during the selection phase are added to the tree in the expansion step. The agents then simulate the game starting at the node chosen during the selection phase until the terminal states are reached. Finally, the terminal states of both agents generate new data for the constitutive model. The constitutive model is then calibrated and falsified against the new data generated by the protagonist and adversary agents. The rewards assigned based on the performance of both agents evaluated at their corresponding terminal states are then propagated back along the path to the root nodes, as illustrated in Fig. 8.

The state values v can be equal to the continuous reward functions Eqs. (10) and (11) to train the policy/value neural networks. To further improve the convergence rate of the DRL algorithm, we propose an empirical method to train the networks with binary state values (1 or -1) post-processed from the reward values, which is similar to the concept of "win" (1) and "lose" (-1) in the game of Chess.

Consider the set of rewards $\{\text{Reward}_{\text{protagonist}}\}_i$ of the $numEpisodes$ games played in the ith DRL iteration. The maximum reward encountered from Iteration 0 to the current Iteration k, is,

$$R_p^{max} = \max_{i\in[0,k]}(\max(\{\text{Reward}_{\text{protagonist}}\}_i)). \tag{14}$$

Meanwhile, the minimum reward is chosen as,

$$R_p^{min} = \max_{i\in[0,k]}(\min(\{\text{Reward}_{\text{protagonist}}\}_i)). \tag{15}$$

A reward range in Iteration k is $R_p^{range} = R_p^{max} - R_p^{min}$. A strategy μ_p is considered as a "win" ($v = 1$) when its reward $\text{Reward}_{\text{protagonist}} \geq R_p^{max} - R_p^{range} * \alpha_{range}$, while it is a "lose" ($v = -1$) when $\text{Reward}_{\text{protagonist}} < R_p^{max} - R_p^{range} * \alpha_{range}$. α_{range} is a user-defined coefficient which influences the degree of "exploration and exploitation" of the AI agents. Similarly, for the adversary agent, the maximum and minimum rewards are

$$R_{a_{\mu_p}}^{max} = \min_{i\in[0,k]}(\max(\{-\text{Reward}_{\text{adversary}}^{\mu_p}\}_i)) \tag{16}$$

and

$$R_{a_{\mu_p}}^{min} = \min_{i\in[0,k]}(\min(\{-\text{Reward}_{\text{adversary}}^{\mu_p}\}_i)), \tag{17}$$

and $R_{a_{\mu_p}}^{range} = R_{a_{\mu_p}}^{max} - R_{a_{\mu_p}}^{min}$ are collected for each protagonist strategy μ_p. Then an attack strategy μ_a corresponding to μ_p is considered as a "win" ($v = 1$) when its reward $-\text{Reward}_{\text{adversary}}^{\mu_p} \leq R_{a_{\mu_p}}^{min} + R_{a_{\mu_p}}^{range} * \alpha_{range}$, while it is a "lose" ($v = -1$) when $-\text{Reward}_{\text{adversary}}^{\mu_p} > R_{a_{\mu_p}}^{min} + R_{a_{\mu_p}}^{range} * \alpha_{range}$. The training examples for the policy/value neural networks are limited to the gameplays in the DRL iterations $i \in [\max(k - i_{lookback}, 0), k]$, where $i_{lookback}$ is a user-defined hyperparameter controlling the degree of "forget" of the AI agents.

Another new contribution to this DRL framework is that we improve the computational efficiency of DRL by executing the mutually independent gameplays and reward evaluations in a parallel manner, instead of serial executions as in previous works [38,48,60]. We use the parallel python library "Ray" [61] for its simplicity and speed in building and running distributed applications. The new workflow of parallel playing of game episodes in each training iteration for DRL is illustrated in Fig. 9.

5. Automated calibration and falsification experiments

We demonstrate the applications of the non-cooperative game for automated calibration and falsification on three types of constitutive models. The material samples are representative volume elements (RVEs) of densely-packed spherical DEM particles. The decision-tree-based experiments are performed via numerical simulations on these samples. The preparation and experiments of the samples are detailed in Appendix A. The three constitutive models studied in this paper are the Drucker–Prager model [62], the SANISAND model [63], and a data-driven traction–separation recurrent neural network model [60]. Their formulations are detailed in Appendix B. The results shown in this section are representatives of the AI agents' performances, since the policy/value networks are randomly initialized and the MCTS simulations involve samplings from action probabilities. The gameplays during DRL iterations may vary, but similar convergence performances are expected for different executions of the algorithm. Furthermore, the material calibration procedures, such as the initial guesses in Dakota and the hyperparameters in training of the neural networks, may affect the game scores and the converged Nash equilibrium points. Finally, since simplifications and assumptions are involved in the DEM samples, their mechanical properties differ from real-world geo-materials. The conclusions of the three investigated constitutive models are only on these artificial and numerical samples. However, the same DRL algorithm is also applicable for real materials, when the actions of the AI experimentalists can be programmed in laboratory instruments.

The policy/value networks f_θ are deep neural network in charge of updating the Q table that determines the optimal strategies. The design of the policy/value networks is identical for both agents in this paper. Both of them

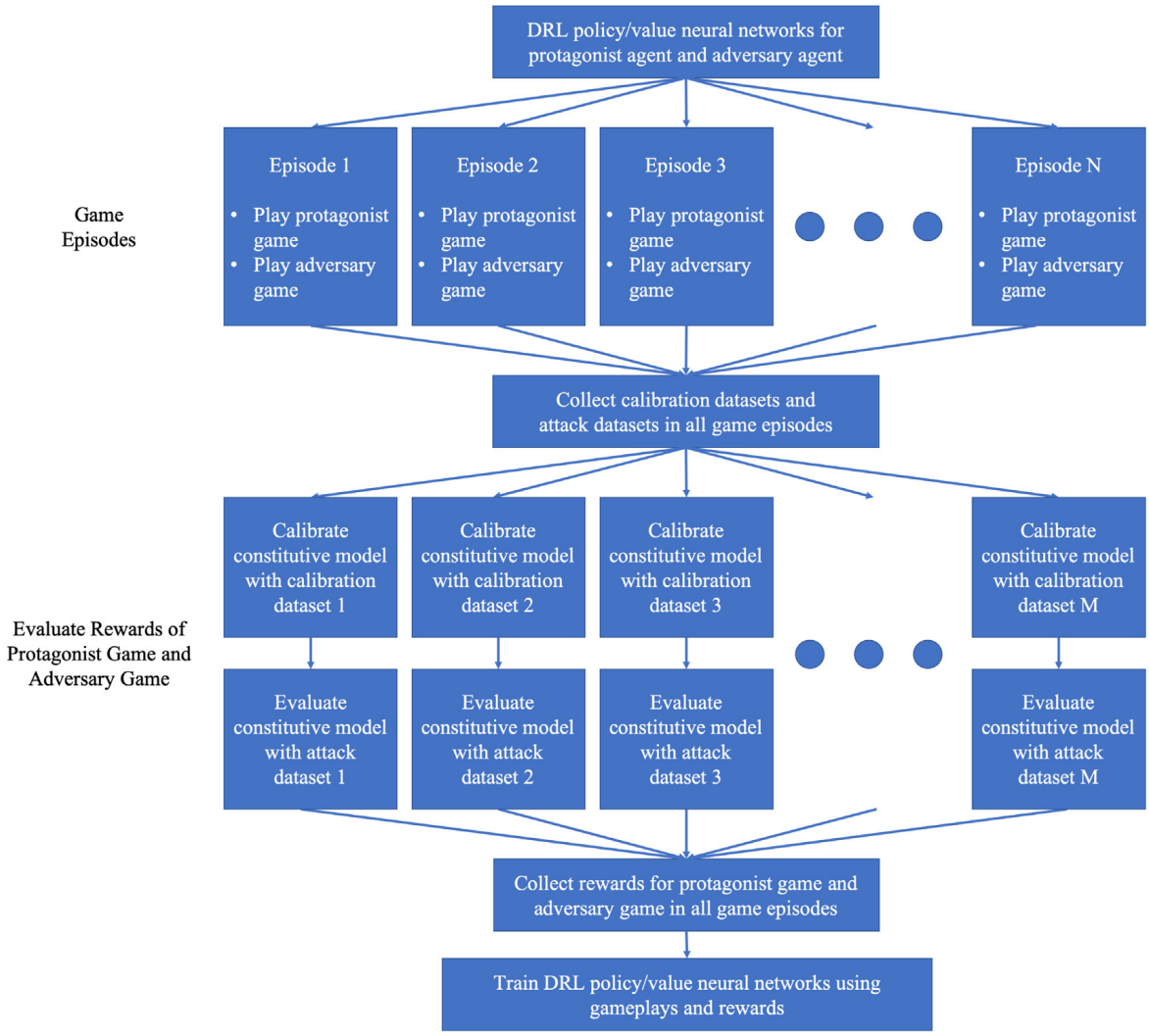


Fig. 9. Workflow of parallel gameplays and reward evaluations in DRL.

consist of one input layer of the game state s, two densely connected hidden layers, and two output layers for the action probabilities $\boldsymbol{p}$ and the state value v, respectively. Each hidden layer contains 256 artificial neurons, followed by Batch Normalization, ReLU activation and Dropout. The dropout layer is a popular regularization mechanism designed to reduce overfitting and improve generalization errors in deep neural network (cf. [64]). The dropout rate is 0.5 for the protagonist and 0.25 for the adversary. These different dropout rates are used such that the higher dropout rate for the protagonist will motivate the protagonist to calibrate the Drucker–Prager model with less generalization errors, while the smaller dropout rate will help the adversary to find the hidden catastrophic failures in response to a large amount of the protagonist's strategies. In addition, the non-cooperative game requires the hyperparameters listed in Table 3 to configure the game.

Remarks on the Computational Cost. The three numerical experiments are run on an Intel(R) Core(TM) i9-9980HK CPU @ 2.40 GHz using 8 cores and 32 GB memory. The typical computation costs (expressed in physical hours) for essential tasks in the proposed parallel reinforcement learning algorithm are listed in Table 4. These computational costs depend on the user-defined hyperparameters for the non-cooperative game (See Table 3) and the training speed of the parameters in the constitutive models. □

Algorithm 1 Self-play reinforcement learning of the non-cooperative meta-modeling game

Require: The definitions of the non-cooperative meta-modeling game: game environment, game states, game actions, game rules, game rewards (Section 3).

Initialize the policy/value networks $f_\theta^{\text{Protagonist}}$ and $f_\theta^{\text{Adversary}}$. For fresh learning, the networks are randomly initialized. For transfer learning, load pre-trained networks instead.
Initialize empty sets of the training examples for both protagonist and adversary $trainExamples^{\text{Protagonist}} \leftarrow []$, $trainExamples^{\text{adversary}} \leftarrow []$.
for i in [0,..., $numIters - 1$] **do**
    for j in [0,..., $numEpisodes - 1$] **do**
        Initialize the starting game state s.
        for $player$ in [Protagonist, Adversary] **do**
            Initialize empty tree of the Monte Carlo Tree search (MCTS), set the temperature parameter τ_{train} for "exploration and exploitation".
            while True **do**
                Check for all legal actions at current state s according to the game rules.
                Get the action probabilities $\pi(s, \cdot)$ for all legal actions by performing $numMCTSSims$ times of MCTS simulations.
                Sample action a from the probabilities $\pi(s, \cdot)$
                Modify the current game state to a new state s by taking the action a.
                if s is the end state of the game of $player$ **then**
                    Evaluate the score of the selected paths in the decision tree.
                    Evaluate the reward r of this gameplay according to the score.
                    Break.
            Append the gameplay history $[s, a, \pi(s, \cdot), r]$ to $trainExamples^{player}$.
    Train the policy/value networks $f_\theta^{\text{Protagonist}}$ and $f_\theta^{\text{Adversary}}$ with $trainExamples^{\text{Protagonist}}$ and $trainExamples^{\text{Adversary}}$.
Use the final trained networks $f_\theta^{\text{Protagonist}}$ and $f_\theta^{\text{Adversary}}$ in MCTS with temperature parameter τ_{test} for one more iteration of "competitive gameplays" to generate the final converged selected experiments.
Exit

5.1. Experiment 1: Drucker–Prager model

The two-player non-cooperative game is played by DRL-based AI experimentalists for Drucker–Prager model. The formulations of the model are detailed by Eqs. (18), (19), (20). The initial guesses, upper and lower bounds of the material parameters for Dakota calibration are presented in Table 6. The game settings are $N_{path}^{max} = 5$ for both the protagonist and the adversary, $E_{NS}^{max} = 1.0$, $E_{NS}^{min} = -1.0$. Hence the combination number of the selected experimental decision tree paths in this example is $180!/(5!(180-5)!) \approx 1.5e9$ where 180 is the total number of leaves in the decision tree and 5 is the maximum number of paths chosen by either agent. The hyperparameters for the DRL algorithm used in this game are $numIters = 10$, $numEpisodes = 50$, $numMCTSSims = 50$, $\alpha_{\text{SCORE}} = 0.0$, $\alpha_{range} = 0.2$, $i_{lookback} = 4$, $\tau_{train} = 1.0$, $\tau_{test} = 0.1$.

The statistics of the game scores played for the "Calibration/Defense" by the protagonist and the "Falsification/Attack" by the adversary during the DRL iterations are shown in Fig. 10. The AI agents only know the experimental decision tree and the rules of the two-player game without any prior knowledge on the strengths and weaknesses of the Drucker–Prager model. At the first DRL iteration, the agents play the game through trial and error guided by randomly initialized policy/value networks and MCTS. This lack of knowledge on proper gameplay strategies can be seen from the widely spread density distribution of game scores and the large inter-quantile range between 25% and 75% in both "Calibration/Defense" and "Falsification/Attack". In the subsequent iterations, the agents progressively understand the "winning strategies" via reinforcement learning on the gameplay histories and the associated game rewards, hence is capable of play games with better outcomes. This trend is evidenced by the increase of the median of game scores by the protagonist, the decrease of the median scores by the adversary,

Table 3
Hyperparameters required to setup the non-cooperative game.

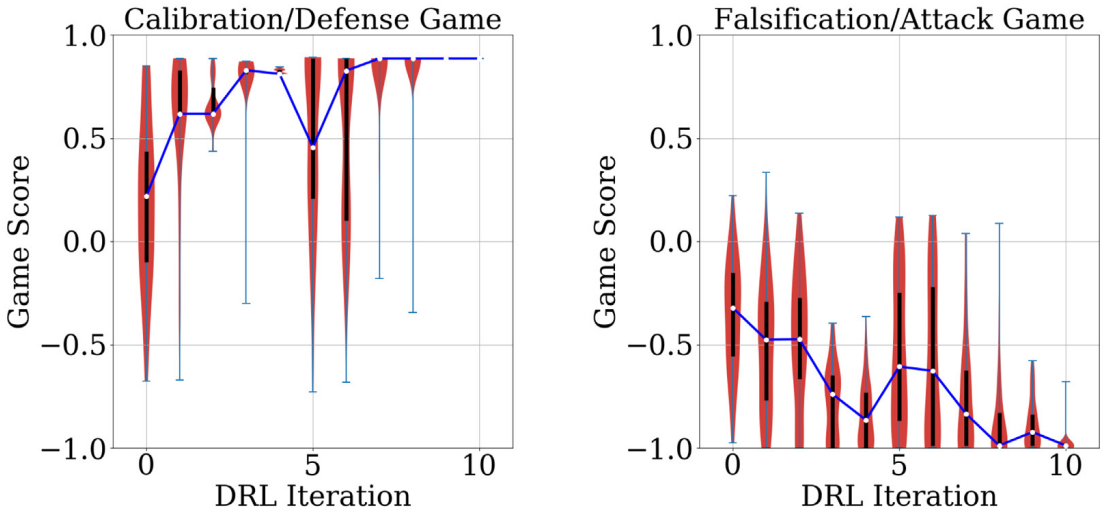
Hyperparameters	Definition	Usage
N_{path}^{max}	Maximum number of decision tree paths chosen by the agents	Define the dimension of the game states
E_{NS}^{max}	Maximum cutoff value of the modified Nash–Sutcliffe efficiency index	See Eq. (9)
E_{NS}^{min}	Minimum cutoff value of the modified Nash–Sutcliffe efficiency index	See Eq. (9)
$numIters$	Number of training iterations	Define DRL iterations for training policy/value networks
$numEpisodes$	Number of gameplay episodes in each training iterations	Define the amount of collected gameplay evidences
$numMCTSSims$	Number of Monte Carlo Tree Search simulations in each gameplay step	Control the agents' estimations of action probabilities
α_{SCORE}	Decay coefficient for the protagonist's reward	See Eq. (10)
α_{range}	Coefficient for determining "win" or "lose" of a game episode	Set the agents' balance between "exploration and exploitation"
$i_{lookback}$	Number of gameplay iterations for training of the policy/value networks	Control the agents' "memory depth"
τ_{train}	Temperature parameter for training iterations	Set the agents' balance between "exploration and exploitation"
τ_{test}	Temperature parameter for competitive gameplays	Set the agents' balance between "exploration and exploitation"

Table 4
Computation costs for the DRL games.

Task	Approximate CPU time (hours)
DEM Test data generation using YADE	0.1
Play one full game episode using MCTS	0.01–0.1 (depending on the complexity of the decision tree)
Training of policy/Value networks	0.1–0.2 (depending on the size of the game examples)
One complete model training and testing	0.2–1.0 (depending on the complexity of the model and the training convergence of the parameters)
Experiment 1 total	10
Experiment 2 total	30
Experiment 3 total	18

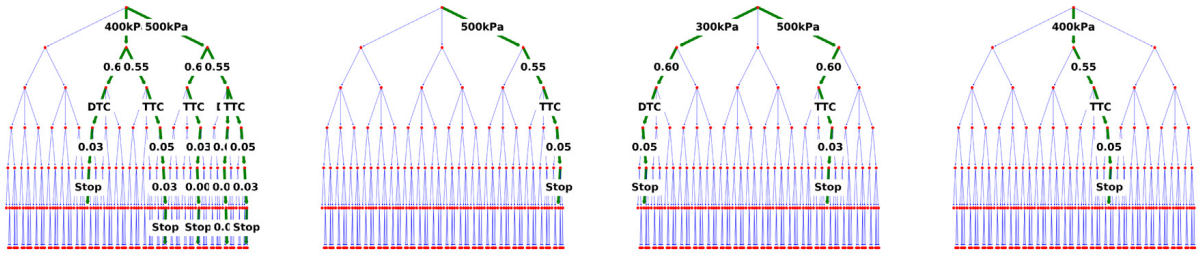
and also the narrowing of inter-quantile ranges. In these intermediate training iterations, games can sometimes be played badly, since the agents are allowed to explore various game policies such that they can avoid the convergence to a local extremum. The strengths of the AI agents after the 0th to 9th training iterations are eventually tested by suppressing the "exploration plays" and focusing on exploitation. The ultimate game scores show outstanding performances.

Examples of paths (experiments) selected by the protagonist during the DRL iterations are shown in Fig. 11. Based on all the evidence presented in these examples, the agent realizes that the Drucker–Prager model is not designed to simultaneously replicate data from samples with different initial confinement, initial void ratio, test



(a) Protagonist **(b)** Adversary

Fig. 10. Violin plots of the density distributions of game scores in each DRL iteration in Drucker–Prager model. The shaded area represents the density distribution of scores. The white point represents the median. The thick black bar represents the inter-quantile range between 25% quantile and 75% quantile. The maximum and minimum scores played in each iteration are marked.



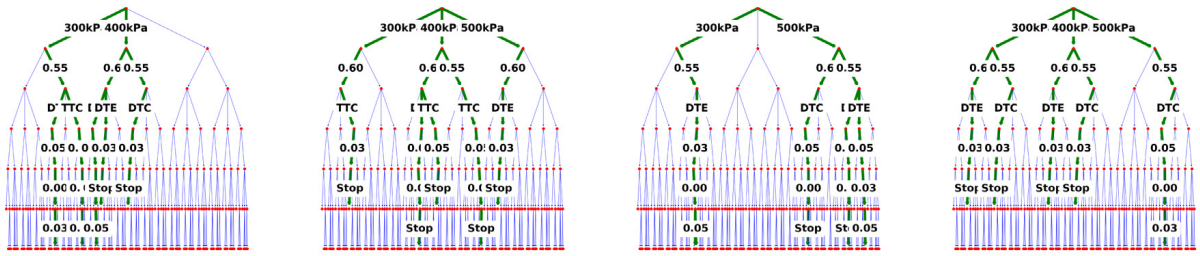
(a) Iteration 0, Episode 10, Defense Game Score: 0.262 **(b)** Iteration 3, Episode 0, Defense Game Score: 0.811 **(c)** Iteration 6, Episode 40, Defense Game Score: 0.699 **(d)** Iteration 10, Episode 0, Defense Game Score: 0.886

Fig. 11. Examples of paths (experiments) in the decision trees selected by the protagonist during the DRL training iterations for the Drucker–Prager model.

types, and unloading–reloading paths. In the end, the agent concludes that the model is only accurate in modeling the mechanical behavior of a single sample in TTC test with monotonic loading. Meanwhile, the adversary tries to attack the models calibrated by the protagonist using the experiments as shown in Fig. 12. The agent progressively comes to the conclusion that, when calibrated with monotonic TTC data, the model fails to predict DTC or DTE experiments on other samples with unloading–reloading. Figs. 13 and 14 shows predicted constitutive responses and the corresponding benchmarks associated to the example decision tree paths shown in Figs. 11 and 12, respectively. They illustrate the strength of the model in replicating the hardening–softening and contraction–dilation behavior of a densely compressed granular material. They also expose the model's weakness in predicting the unloading–reloading behavior, regardless of the calibration data. These conclusions on the Drucker–Prager model by the AI agents are consistent with the judgments from human experts, but they are drawn from the reinforcement learning on the two-player game without human knowledge.

5.2. Experiment 2: SANISAND model

The two-player non-cooperative game is played by DRL-based AI experimentalists for SANISAND model. The formulations of the model are detailed by Eqs. (21), (22), (23), (24), (25). The initial guesses, upper and lower



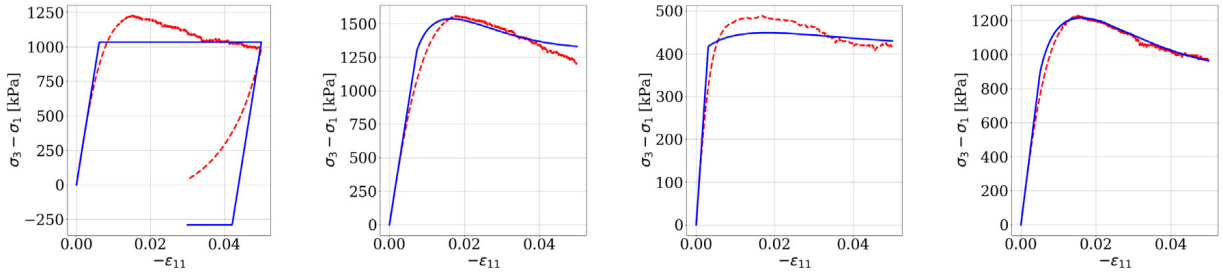
(a) Iteration 0, Episode 10, Attack Game Score: -0.151

(b) Iteration 3, Episode 0, Attack Game Score: -0.725

(c) Iteration 6, Episode 40, Attack Game Score: -0.570

(d) Iteration 10, Episode 0, Attack Game Score: -0.989

Fig. 12. Examples of paths (experiments) in the decision trees selected by the adversary during the DRL training iterations for the Drucker–Prager model.

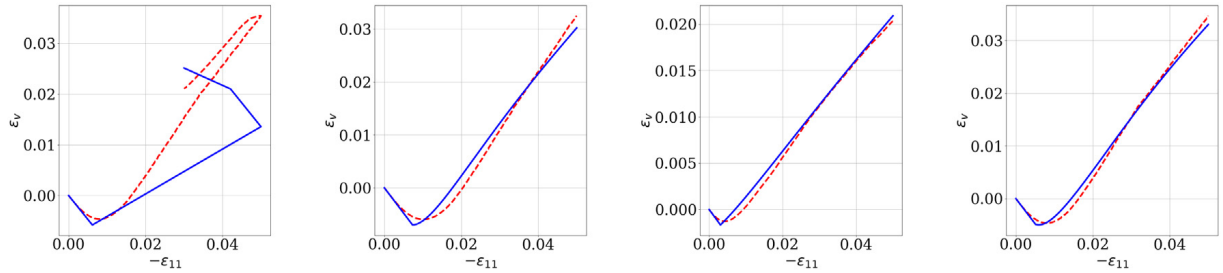


(a) Iteration 0, Episode 10, Defense Game Score: 0.262

(b) Iteration 3, Episode 0, Defense Game Score: 0.811

(c) Iteration 6, Episode 40, Defense Game Score: 0.699

(d) Iteration 10, Episode 0, Defense Game Score: 0.886



(e) Iteration 0, Episode 10, Defense Game Score: 0.262

(f) Iteration 3, Episode 0, Defense Game Score: 0.811

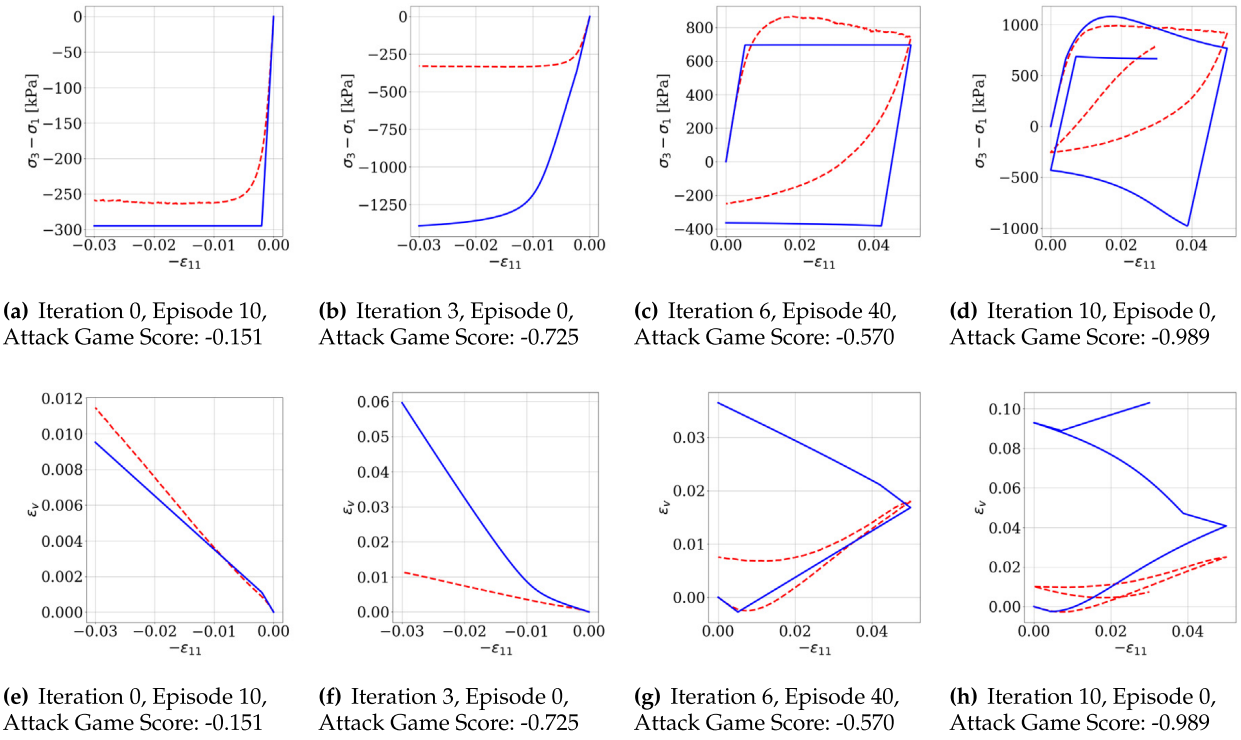
(g) Iteration 6, Episode 40, Defense Game Score: 0.699

(h) Iteration 10, Episode 0, Defense Game Score: 0.886

Fig. 13. Examples of response curves of the games played by the protagonist during the DRL training iterations for the Drucker–Prager model. Experimental data are plotted in red dashed curves, model predictions are plotted in blue solid curves.

bounds of the material parameters for Dakota calibration are presented in Table 7. The game settings are $N_{path}^{max} = 5$ for both the protagonist and the adversary, $E_{NS}^{max} = 1.0$, $E_{NS}^{min} = -1.0$. The hyperparameters for the DRL algorithm are $numIters = 10$, $numEpisodes = 40$, $numMCTSSims = 50$, $\alpha_{\text{SCORE}} = 1.0$, $\alpha_{range} = 0.2$, $i_{lookback} = 4$, $\tau_{train} = 1.0$, $\tau_{test} = 0.1$. The policy/value networks for both AI agents are identical to the ones used in the previous example. In order to help explore $\min(\{E_{NS}^{1}\})$ in Eq. (10), we also manually pre-select 5 experiments that have unloading–reloading paths which need to be predicted by all calibrated SANISAND models, along with the test data selected by the adversary.

The statistics of the game scores played for the "Calibration/Defense" by the protagonist and the "Falsification/Attack" by the adversary during the DRL iterations are shown in Fig. 15. The AI agents only know the experimental decision tree and the rules of the two-player game without any prior knowledge on the strengths and weaknesses of the SANISAND model. The improvement of the protagonist's policy is shown by the increase of the median of game scores, and also the narrowing of inter-quantile ranges. Some lower scores encountered during



(a) Iteration 0, Episode 10, Attack Game Score: -0.151

(b) Iteration 3, Episode 0, Attack Game Score: -0.725

(c) Iteration 6, Episode 40, Attack Game Score: -0.570

(d) Iteration 10, Episode 0, Attack Game Score: -0.989

(e) Iteration 0, Episode 10, Attack Game Score: -0.151

(f) Iteration 3, Episode 0, Attack Game Score: -0.725

(g) Iteration 6, Episode 40, Attack Game Score: -0.570

(h) Iteration 10, Episode 0, Attack Game Score: -0.989

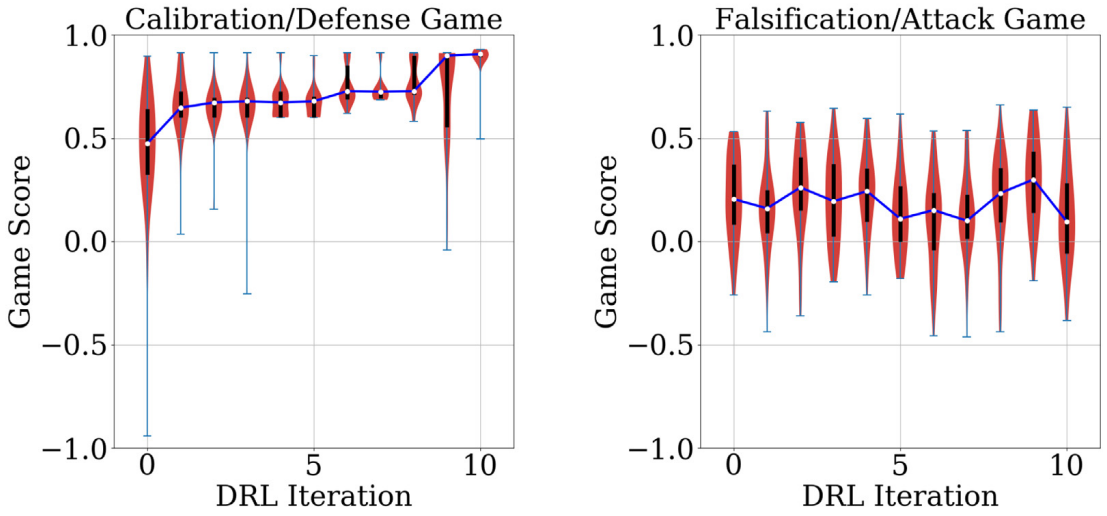
Fig. 14. Examples of response curves of the games played by the adversary during the DRL training iterations for the Drucker–Prager model. Experimental data are plotted in red dashed curves, model predictions are plotted in blue solid curves.

the later iterations of the DRL are due to random explorations of game strategies by the agent. Fig. 16 provides some example experiments selected by the protagonist for calibration data and Fig. 18 provides some example response curves associated to these experiments. Meanwhile, the adversary tries to attack the models calibrated by the protagonist using some experiments as shown in Fig. 17. Fig. 19 gives example response curves associated to these adversarial decision tree paths. These attacks inform and drive the protagonist to find more adequate calibration data via the score systems in this game.

In the end, the protagonist concludes that the model is accurate in modeling the mechanical behavior of a single sample in TTC test with monotonic loading. In this case, the final value explored for $\min(\{E^1_{NS}\})$ is -0.933, which is slightly above the lower bound $E^{min}_{NS} = -1.0$ in the game score setting. Hence the decay coefficient in Eq. (10) is not activated and the protagonist score is equal to the calibration score. The adversary concludes that, when calibrated with this monotonic TTC data, the model is not accurate in predicting DTC, DTE, TTC experiments on other samples with unloading–reloading. Nevertheless, based on all the game episodes played during the DRL, the agents learn that the SANISAND model is capable of replicating the hardening–softening and contraction–dilation behavior of a densely compressed granular material. They also learn that SANISAND is more powerful than Drucker–Prager in replicating data from samples with different initial confinement, initial void ratio, test types, and unloading–reloading paths. Note that he relative weak performance on calibrations and forward predictions shown in Figs. 18 and 19 may be due to the limited choices of calibration procedures. Furthermore, as the SANISAND model is designed to replicate constitutive responses of real sand assembles as continua, the difference between the discrete element simulaltions and real experiments the different morphology of real sand and spherical particles, and the resultant different topology of grain connectivity may all contribute to the discrepancy.

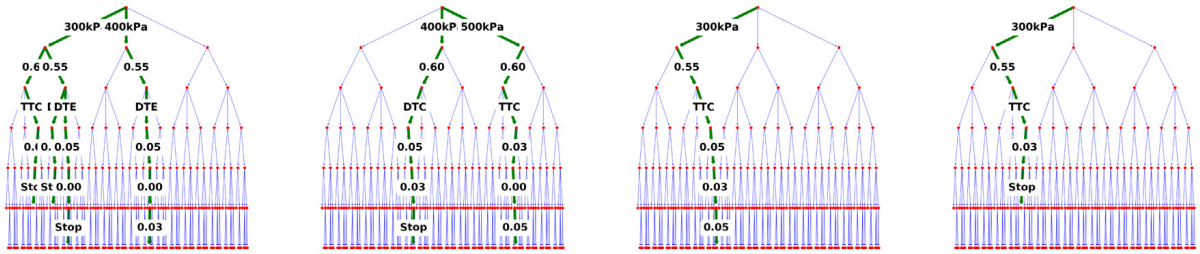
5.3. Experiment 3: Deep learning graph-based traction–separation model

The two-player non-cooperative game is played by DRL-based AI experimentalists for the data-driven traction–separation model. The traction–separation law is trained via a recurrent neural network called gated recurrent unit.



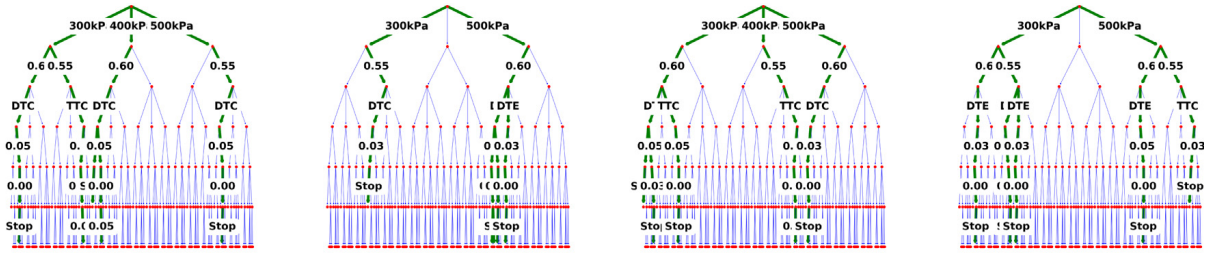
(a) Protagonist **(b)** Adversary

Fig. 15. Violin plots of the density distributions of game scores in each DRL iteration in the SANISAND model. The shaded area represents the density distribution of scores. The white point represents the median. The thick black bar represents the inter-quantile range between 25% quantile and 75% quantile. The maximum and minimum scores played in each iteration are marked.



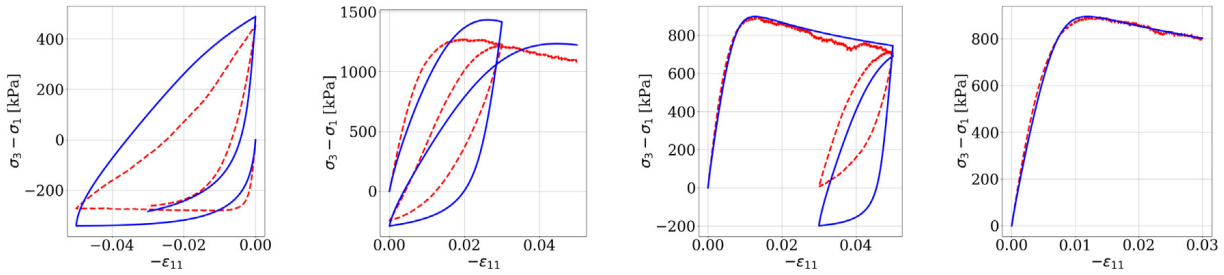
(a) Iteration 0, Episode 24, Defense Game Score: 0.162 **(b)** Iteration 3, Episode 33, Defense Game Score: 0.626 **(c)** Iteration 6, Episode 10, Defense Game Score: 0.687 **(d)** Iteration 10, Episode 30, Defense Game Score: 0.912

Fig. 16. Examples of paths (experiments) in the decision trees selected by the protagonist during the DRL training iterations for the SANISAND model.



(a) Iteration 0, Episode 24, Attack Game Score: 0.206 **(b)** Iteration 3, Episode 33, Attack Game Score: 0.323 **(c)** Iteration 6, Episode 10, Attack Game Score: 0.295 **(d)** Iteration 10, Episode 30, Attack Game Score: -0.290

Fig. 17. Examples of paths (experiments) in the decision trees selected by the adversary during the DRL training iterations for the SANISAND model.

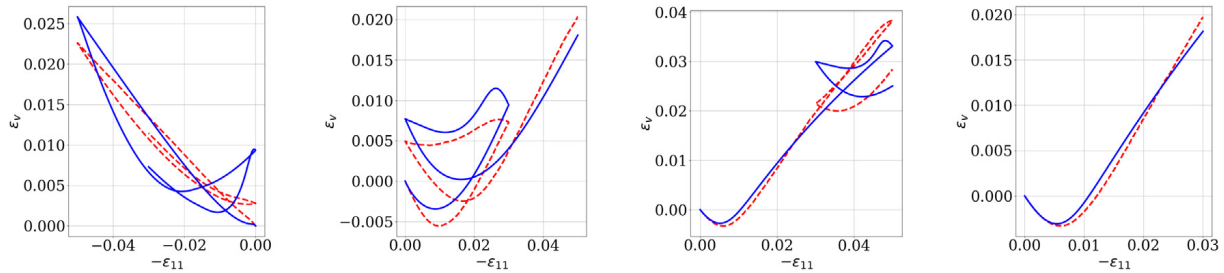


(a) Iteration 0, Episode 24, Defense Game Score: 0.162

(b) Iteration 3, Episode 33, Defense Game Score: 0.626

(c) Iteration 6, Episode 10, Defense Game Score: 0.687

(d) Iteration 10, Episode 30, Defense Game Score: 0.912



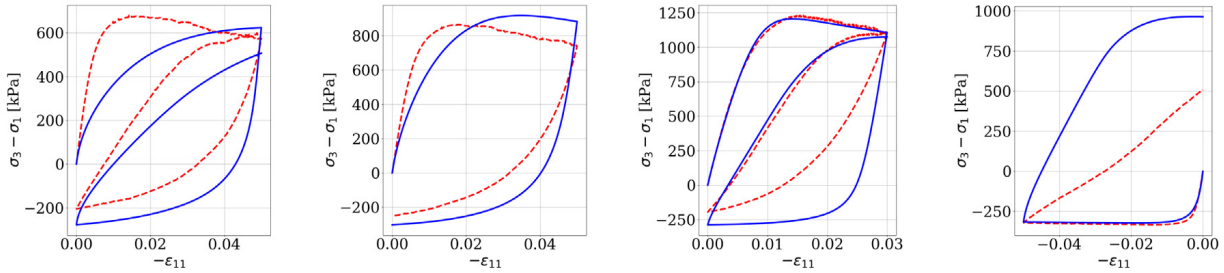
(e) Iteration 0, Episode 24, Defense Game Score: 0.162

(f) Iteration 3, Episode 33, Defense Game Score: 0.626

(g) Iteration 6, Episode 10, Defense Game Score: 0.687

(h) Iteration 10, Episode 30, Defense Game Score: 0.912

Fig. 18. Examples of response curves of the games played by the protagonist during the DRL training iterations for the SANISAND model. Experimental data are plotted in red dashed curves, model predictions are plotted in blue solid curves.

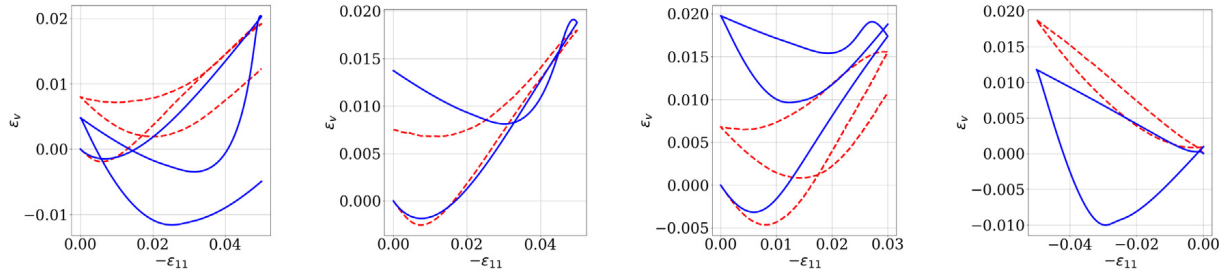


(a) Iteration 0, Episode 24, Attack Game Score: 0.206

(b) Iteration 3, Episode 33, Attack Game Score: 0.323

(c) Iteration 6, Episode 10, Attack Game Score: 0.295

(d) Iteration 10, Episode 30, Attack Game Score: -0.290



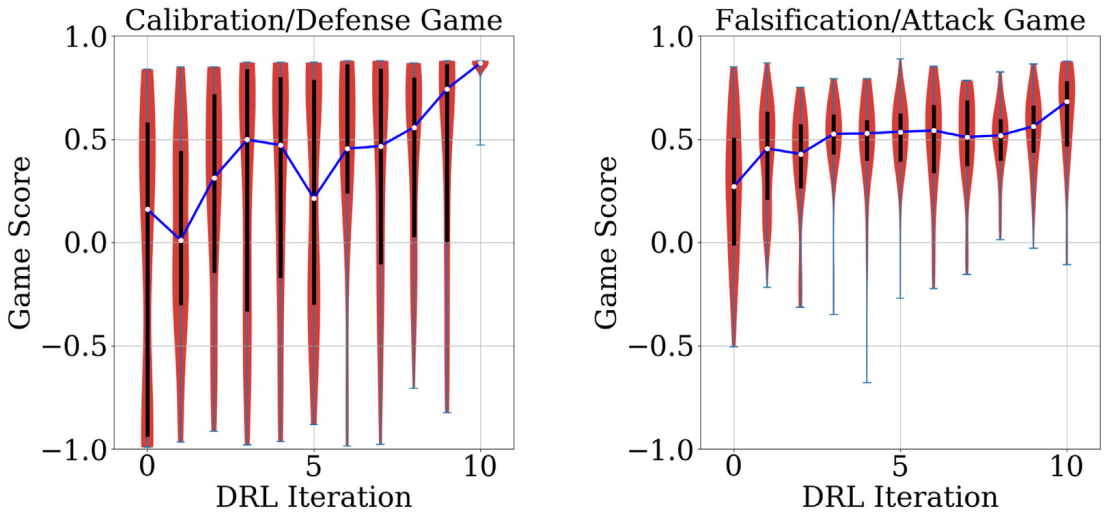
(e) Iteration 0, Episode 24, Attack Game Score: 0.206

(f) Iteration 3, Episode 33, Attack Game Score: 0.323

(g) Iteration 6, Episode 10, Attack Game Score: 0.295

(h) Iteration 10, Episode 30, Attack Game Score: -0.290

Fig. 19. Examples of response curves of the games played by the adversary during the DRL training iterations for the SANISAND model. Experimental data are plotted in red dashed curves, model predictions are plotted in blue solid curves.



(a) Protagonist **(b)** Adversary

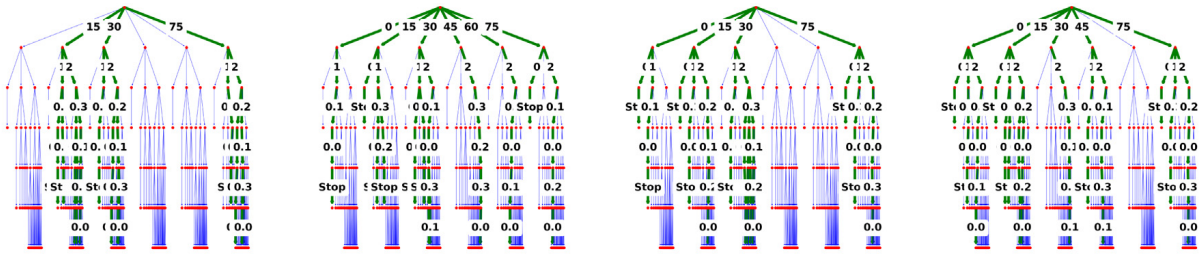
Fig. 20. Violin plots of the density distributions of game scores in each DRL iteration in data-driven traction–separation model. The shaded area represents the density distribution of scores. The white point represents the median. The thick black bar represents the inter-quantile range between 25% quantile and 75% quantile. The maximum and minimum scores played in each iteration are marked.

Similar approaches have been used to train plasticity models in [15,65] and traction–separation laws in [48,66] For brevity, the neural network architectures and the calibration of the model are outlined in Appendix B.3.

The game settings are $N_{path}^{max} = 15$ for the protagonist and $N_{path}^{max} = 10$ for the adversary, $E_{NS}^{max} = 1.0$, $E_{NS}^{min} = 0.8$. Hence the combination number of the selected experimental decision tree paths in this example is about $1e23$. The hyperparameters for the DRL algorithm are $numIters = 10$, $numEpisodes = 40$, $numMCTSSims = 50$, $\alpha_{\text{SCORE}} = 2.5$, $\alpha_{range} = 0.1$, $i_{lookback} = 4$, $\tau_{train} = 1.0$, $\tau_{test} = 0.1$. The policy/value networks are identical to the ones used in the previous examples. In this example, since the number of the possible game configurations is enormous, we constrain the policy of the protagonist to play only the winning games (those who have got their rewards $\text{Reward}_{\text{protagonist}} \geq R_p^{max} - 0.1 * R_p^{range} * \alpha_{range}$) in 1/5 of the 40 game episodes in each training iteration and in the last iteration of "competitive gameplays". Also, we manually pre-select 10 experiments that have two loading cycles and regard them as the "shared" test data. They need to be predicted by all calibrated traction–separation models, along with the test data selected by the adversary, in order to help explore $\min(\{E_{NS}^1\})$ in Eq. (10). These methods are applied in addition to the general reinforcement learning framework in Section 4 to enhance the convergence of the agents' gameplay strategies.

The statistics of the game scores played for the "Calibration/Defense" by the protagonist and the "Falsification/Attack" by the adversary during the DRL iterations are shown in Fig. 20. The improvement of the protagonist's policy is shown by the increase of the median of game scores. Fig. 21 provides some examples of experiments selected by the protagonist for calibration data. The protagonist progressively develops the intelligence to select experiments from multiple displacement jump angles and multiple loading cycles, instead of concentrating on selections only cover very few angles and monotonic loading. This is consistent with intuitions from human experts, but is automatically discovered by the AI. We further include some examples of estimated Q-values of the experimental decision tree (Fig. 22) by the protagonist to illustrate how the agent is learning during the DRL. We record the agent's policy/value network $(\boldsymbol{p}, v) = f_\theta(s)$ trained after each iteration of the DRL. Each checkpoint is used to predict the Q-value of each possible state in the experimental decision tree. The figure presents the expectations from the protagonist, before choosing any experiments, on how beneficial if an experiment is included in the calibration data. The evolution of the colors illustrates the progressively improved Q-value estimations learned from the game episodes and their rewards collected during the DRL.

Meanwhile, the adversary's scores also tend to increase as opposed to the previous examples. This could be attributed to the fact that the increasingly well-trained model by the protagonist using more effective calibration data, hence the corresponding prediction accuracy on unseen testing data also increases. Nevertheless, the adversarial



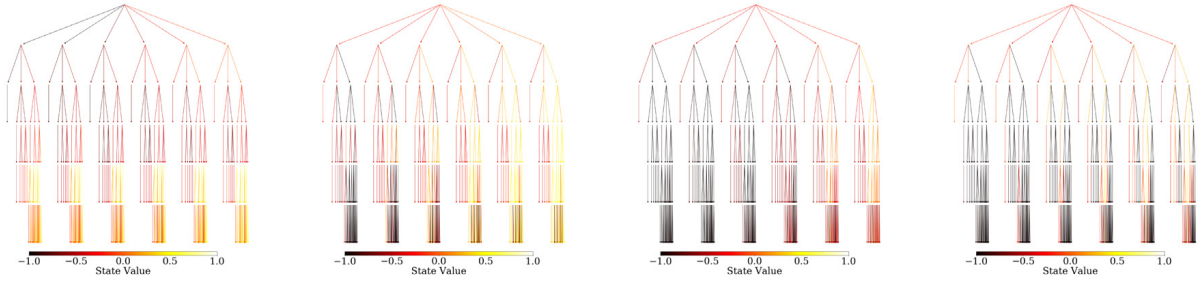
(a) Iteration 0, Episode 11, Defense Game Score: -0.992

(b) Iteration 3, Episode 7, Defense Game Score: -0.331

(c) Iteration 6, Episode 31, Defense Game Score: 0.269

(d) Iteration 10, Episode 20, Defense Game Score: 0.879

Fig. 21. Examples of paths (experiments) in the decision trees selected by the protagonist during the DRL training iterations for the traction–separation model.



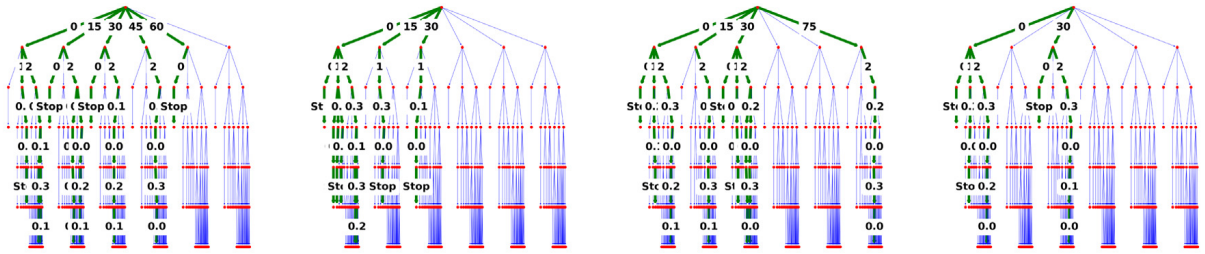
(a) Iteration 0

(b) Iteration 3

(c) Iteration 6

(d) Iteration 10

Fig. 22. Examples of Q-values of all possible states in the experimental decision tree estimated by the protagonist's policy/value network f_θ during the DRL training iterations for the traction–separation model.



(a) Iteration 0, Episode 11, Attack Game Score: -0.085

(b) Iteration 3, Episode 7, Attack Game Score: -0.127

(c) Iteration 6, Episode 31, Attack Game Score: 0.540

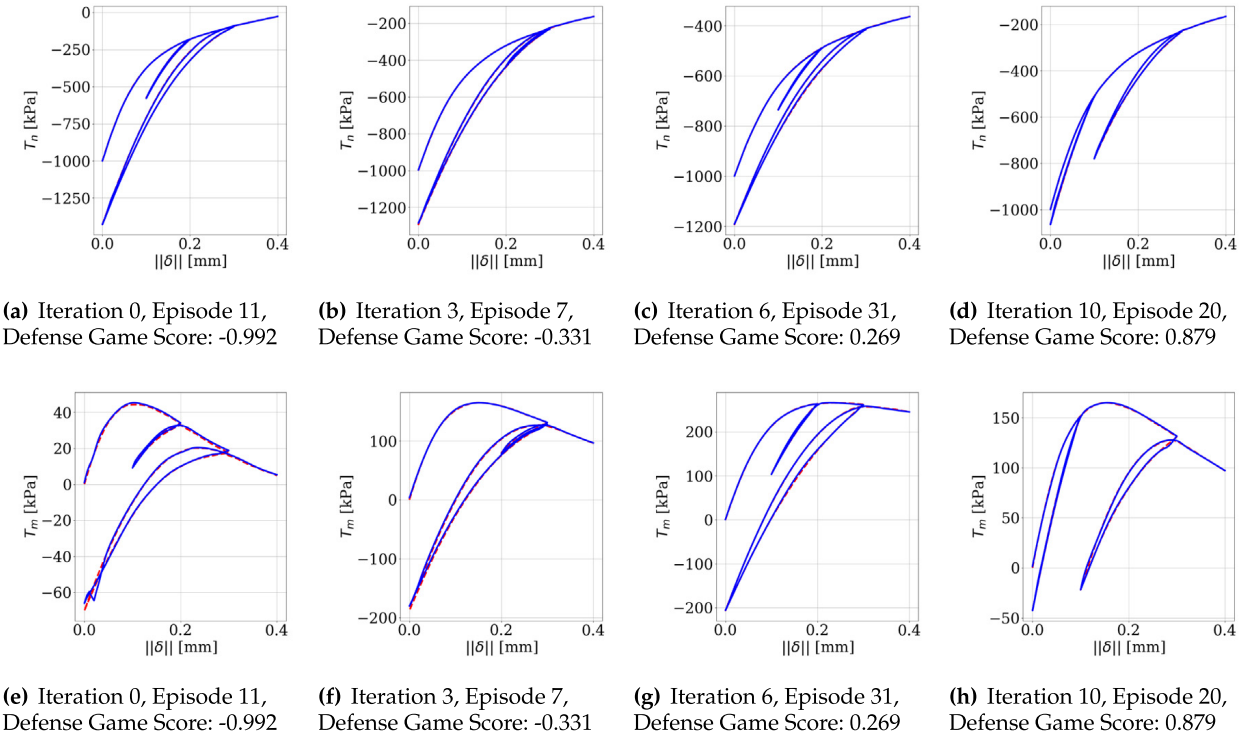
(d) Iteration 10, Episode 20, Attack Game Score: 0.443

Fig. 23. Examples of paths (experiments) in the decision trees selected by the adversary during the DRL training iterations for the traction–separation model.

game objective keeps driving the adversary to explore the model's weakest performance. *This feedback loop plays an important role in forcing the protagonist to find more adequate calibration data to make the model more resilient to attacks orchestrated by the adversary agent, based on improved skills learned from previous walks on the decision tree.* Some example experiments selected by the adversary for testing data along the DRL are provided in Fig. 23.

In this example, we observe a slow convergence of the game policies as shown by the score distributions in Fig. 20. We attribute these difficulties to the following factors.

1. The game dimension of this example is $228!/(10!(228-10)!) \approx 1e23$ where 228 is the total number of leaves in the decision tree (cf. Section 2.2) and 10 is the maximum number of paths chosen by the agent (N_{path}^{max}), even when the agents try to follow the best policy learned from previous gameplay experiences, a



(a) Iteration 0, Episode 11, Defense Game Score: -0.992

(b) Iteration 3, Episode 7, Defense Game Score: -0.331

(c) Iteration 6, Episode 31, Defense Game Score: 0.269

(d) Iteration 10, Episode 20, Defense Game Score: 0.879

(e) Iteration 0, Episode 11, Defense Game Score: -0.992

(f) Iteration 3, Episode 7, Defense Game Score: -0.331

(g) Iteration 6, Episode 31, Defense Game Score: 0.269

(h) Iteration 10, Episode 20, Defense Game Score: 0.879

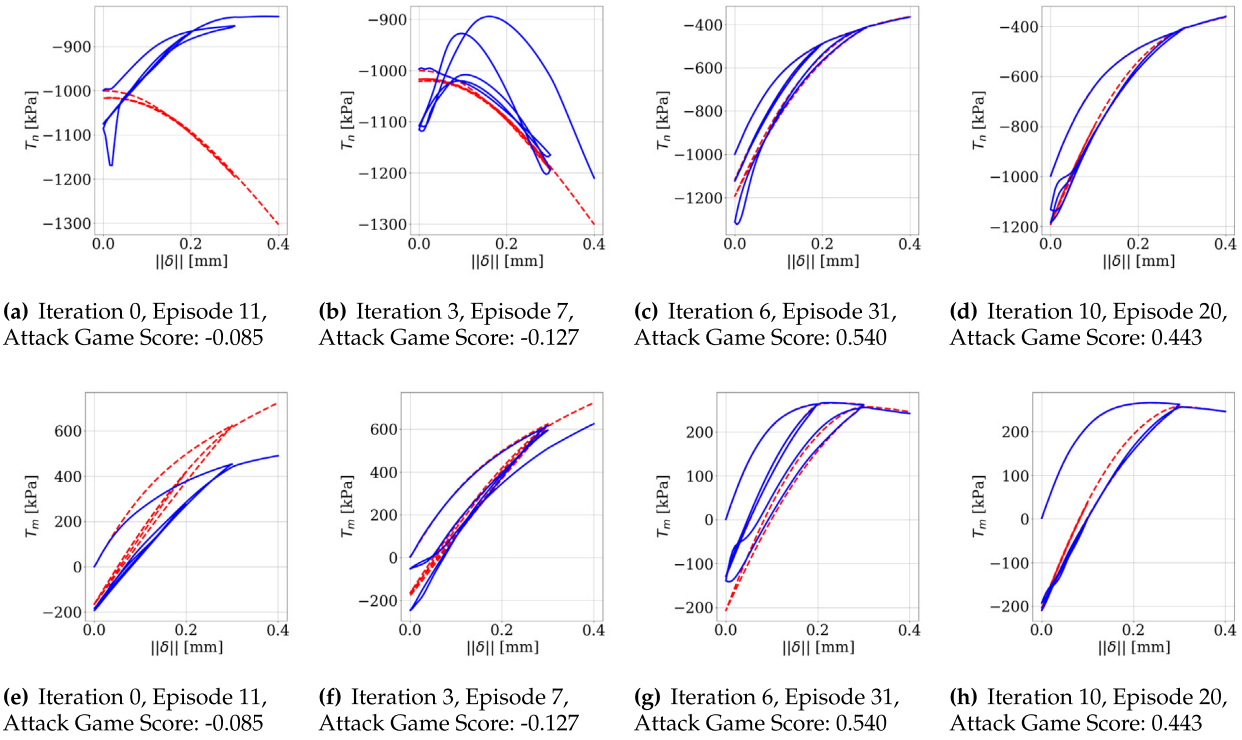
Fig. 24. Examples of response curves of the games played by the protagonist during the DRL training iterations for the traction–separation model. Experimental data are plotted in red dashed curves, model predictions are plotted in blue solid curves.

slight deviation from this policy due to the freedom of "exploration" may lead to significant deterioration on the game performance.

2. The ANN-based traction–separation model is highly adaptive to the calibration data. Neural networks can be trained very accurately to a broad range of calibration data, without changing its architecture. The handcrafted models from experts, however, are developed from fixed theory and have fixed mathematical expressions. They are not flexible to significant changes in calibration data. This can be seen from the example response curves in Fig. 24. Because the calibration scores are uniformly high, the performance of the protagonist cannot be judged solely based on the calibration accuracy.
3. The performance of the protagonist mainly depends on the prediction accuracy on unseen testing data. To ensure fairness, the assessment on the accuracy (the score the protagonist received) nevertheless also depends on the data generated from experiments designed by the adversary and is not soely controlled by the action of the protagonist.

At the early stage of the game, the adversary is learning from scratch without any human knowledge provided. As a result, the Q table of the adversary may not be accurate enough to find the best way to attack the trained model properly in the first DRL iterations. Some early gameplays from the protagonist may yield an apparently high reward, because the adversary may design experiments very similar to that of the protagonist used for calibration. However, such a high score should be interpreted with caution due to the lack of sufficient expolartion in the early gameplays. For example, the protagonist may only train the model with monotonic loading and few loading angles, whereas the adversary also test the model via similiar monotonic loadings. In this case, the gameplays have not yet provided sufficient knowledge to the adversary to expose the potential weakness of the constitutive models at other stress pathes. As a result, the Q tables for both agents are not sufficiently accurate to yield a score that carries the credibility to predict the performance of the unexplored loading paths in the decision tree.

At the late stage of the game (Iteration 10), the DRL is capable of improving the blind prediction capability of the data-driven models, as shown in Fig. 25. This result is attributed to the fact that the Q table of the protagonist has been sufficiently improved such that it is able to design experiments that calibrate the model much better. While adversary agent that launch the adversarial attacks are given the same opportunities to improve its own Q table and



(a) Iteration 0, Episode 11, Attack Game Score: -0.085

(b) Iteration 3, Episode 7, Attack Game Score: -0.127

(c) Iteration 6, Episode 31, Attack Game Score: 0.540

(d) Iteration 10, Episode 20, Attack Game Score: 0.443

(e) Iteration 0, Episode 11, Attack Game Score: -0.085

(f) Iteration 3, Episode 7, Attack Game Score: -0.127

(g) Iteration 6, Episode 31, Attack Game Score: 0.540

(h) Iteration 10, Episode 20, Attack Game Score: 0.443

Fig. 25. Examples of response curves of the games played by the adversary during the DRL training iterations for the traction–separation model. Experimental data are plotted in red dashed curves, model predictions are plotted in blue solid curves.

therefore the policy decision skills, its action no longer exposes any particularly severe weakness of the model. This assessment is due to the non-negative Attack Game Score in this game (0.443) which is significantly higher than the Attack Game Score in the previous two games for the Drucker–Prager model (-0.989) and the SANISAND model (-0.29). This result indicates the machine learning traction–separation model may provide efficiently robust predictions on the unseen cases within the decision tree.

More importantly, these three numerical experiments show that the competition between the two agents is helpful to improve the robustness of the collective performance of both agents. This finding is consistent with the previous efforts on the validation and blind predictions of material models such as the Sandia Fracture Challenges [17,67] and the VELAS project [68,69]. Similar conclusions can also be found in AI for video game (cf. [70]) and simulated biological evolution (cf. [71]) where the co-evolutionary algorithm is shown to improve the robustness of the agents.

6. Conclusion and future perspectives

We introduce a multi-agent non-cooperative meta-modeling game in which the generation of calibration/validation data and the adversary testing data aided to falsify the model are handled by two competing artificial intelligence experimentalist agents. Mimicking the competition between a pair of protagonist and adversary in order to calibrate/validate and falsify a constitutive model for a path-dependent process, these two AI agents interact with each other sequentially and exchange information until both agents reach their own objectives and further actions do not gain better individual rewards. The winning strategies of the non-cooperative game are efficiently searched by the deep reinforcement learning (DRL) technique. The wide-applicability and efficiency of our approach have been shown through two elasto-plasticity models and a data-driven traction–separation model, with the number of possible game configurations as enormous as $1.5e9$ and $1e23$, respectively. To the best knowledge of the authors, this is the first time the strategies of experimentalists who provide data to validate/falsify a history-dependent model are formulated in a non-cooperative decision-making game. Both agents are able to continuously improve their knowledge of the constitutive law using experimental data generated by each other in a competitive DRL framework [35,36] and establish the Nash equilibria strategies that inform us the quality of the models in the applications represented in the decision trees. Such innovations are keys for developing powerful AI assistants that

take over large amounts of trial-and-error burdens from human researchers for material modeling and knowledge discoveries with decision trees that are too deep to explore manually. More importantly, the competitive nature enables us to not only find the optimal setup of constitutive laws, but also find out the weakness of the models in an unbiased third-party manner.

Further improvements and extensions can be made regarding the following aspects of our current framework. (1) Experimental data from real-world granular materials, instead of DEM samples, can be used if the AI agents' decision trees are connected with laboratory instruments. (2) The rewards of the game strategies and hence the conclusions drawn from the game are sensitive to the game settings and the score systems (objective functions), e.g., the hyperparameters N_{path}^{max}, E_{NS}^{max}, E_{NS}^{min}, α_{SCORE}. These game designs need to be tuned by human experts in order to appropriately investigate the strengths and weaknesses of the model. (3) They also depend on the calibration procedures, especially the elasto-plasticity models from human experts in which the material parameters have specific meanings and require initial guesses and bounds. The neural network models, on the other hand, can adapt to a wide range of material behavior and calibrate well, but with material parameters difficult to interpret.

Declaration of competing interest

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

Acknowledgments

The work of KW and WCS is supported by National Science Foundation, United States of America under grants contracts CMMI-1846875 and CMMI-1940203, the Earth Materials and Processes program from the US Army Research Office under grant contract W911NF-18-2-0306, the Dynamic Materials and Interactions Program from the Air Force Office of Scientific Research, United States of America under grant contract FA9550-17-1-0169 and FA9550-19-1-0318, as well as the Columbia SEAS Interdisciplinary Research Seed, United States of America Grant. The work of QD is supported in part by NSF CCF-1704833, DMS-1719699, DMR-1534910, and ARO MURI, United States of America W911NF-15-1-0562. These supports are gratefully acknowledged. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the sponsors, including the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Appendix A. Preparation and experiments of DEM samples

The data for calibration and evaluation of the prediction accuracy of the constitutive models are generated by numerical simulations on representative volume elements (RVEs) of densely-packed spherical DEM particles. The open-source discrete element simulation software YADE for DEM is used by the AI experimentalist agents to generate data, including the homogenized stress and strain measures and the geometrical attributes such as porosity, coordination number and fabric tensor [72].

The RVEs for AI-guided experimentation on bulk granular materials (Section 2.1) consist of discrete element particles having radii between 1 ± 0.3 mm with a uniform distribution. The Cundall's elastic-frictional contact model [73] is used for the inter-particle constitutive law. The material parameters are: interparticle elastic modulus $E_{eq} = 0.5$ GPa, ratio between shear and normal stiffness $k_s/k_n = 0.3$, frictional angle $\varphi = 30°$, density $\rho = 2600$ kg/m^3, Cundall damping coefficient $\alpha_{damp} = 0.4$. Firstly, a random loose packing enclosed in a parallelepiped of edge size 50 mm is generated. The sample is then isotropically compressed to $p_0 = -300$ kPa. $e_0 = 0.60$ is approximated using a fictitious frictional angle of 0.05 rad, whereas $e_0 = 0.55$ is approximated using a frictional angle of 0.01 rad. These samples are then isotropically compressed to higher confinements $p_0 = -400$ kPa and $p_0 = -500$ kPa. The generated RVE samples for the AI experimentalists to choose are presented in Table 5. The 'DTC', 'DTE', 'TTC' tests are conducted quasi-statically using the YADE engine "PeriTriaxController", and data are recorded at every strain increment of 1e−4.

The RVEs for AI-guided experimentation on granular interfaces (Section 2.2) consist of discrete element particles having radii between 1 ± 0.3 mm with a uniform distribution. The Cundall's elastic-frictional contact model is used. The material parameters are identical to those of the bulk RVEs. The sample with initially random loose

Table 5
Initial DEM samples for AI-guided experimentation on bulk granular materials.

Sample no.	'Sample p_0'	'Sample e_0'	p_0	e_0
1	'300 kPa'	'0.60'	−300 kPa	0.5955
2	'300 kPa'	'0.55'	−300 kPa	0.5554
3	'400 kPa'	'0.60'	−400 kPa	0.5936
4	'400 kPa'	'0.55'	−400 kPa	0.5538
5	'500 kPa'	'0.60'	−500 kPa	0.5917
6	'500 kPa'	'0.55'	−500 kPa	0.5521

Table 6
Initial guesses, upper and lower bounds of the material parameters for Drucker–Prager model.

Parameter	Initial guess	Lower bound	Upper bound
G_0	6e4 kPa	4e4 kPa	8e4 kPa
ν	0.25	0.1	0.4
a_0	1.0	0.5	1.5
a_1	2e4	1e2	6e4
a_2	1e−5 1/Pa	5e−6 1/Pa	5e−5 1/Pa
a_3	60.0	20.0	200.0
β_0	0.5	0.2	0.8

packing is isotropically compressed to $p_0 = -1$ MPa using a fictitious frictional angle of 0.01 rad. Hence the initial traction is -1 MPa in the normal direction and 0 MPa in the tangential direction. The width between the upper and lower surfaces of the sample is 20 mm. The mixed-mode shear tests with different loading paths are conducted quasi-statically, and data are recorded at every displacement jump increment of 0.005 mm.

Appendix B. Material models in numerical examples

B.1. Drucker–Prager elasto-plasticity model (Section 5.1)

The model adopts a linear elasticity law with the elastic stiffness tensor

$$\boldsymbol{C}^e = K\boldsymbol{I}\otimes\boldsymbol{I} + 2G(\boldsymbol{I}^4_{sym} - \frac{\boldsymbol{I}\otimes\boldsymbol{I}}{3}), \tag{18}$$

where K is the elastic bulk modulus and G is the elastic shear modulus.

$$\begin{cases} K = K_0 = \dfrac{2(1+\nu)}{3(1-2\nu)}G_0 \\ G = G_0 \end{cases}, \tag{19}$$

where G_0 is the reference shear modulus and ν is the Poisson ratio.

The yield surface has the form $f = q + \alpha p$, where $p = \frac{1}{3}\mathrm{tr}(\boldsymbol{\sigma})$, $\boldsymbol{s} = \boldsymbol{\sigma} - p\boldsymbol{I}$, $q = \sqrt{3J_2} = \sqrt{\frac{3}{2}}\|\boldsymbol{s}\|$. The potential surface has the form $g = q + \beta p - c_g$. α and β evolve according to

$$\begin{cases} \alpha = a_0 + a_1\bar{\epsilon}^p \exp(a_2 p - a_3\bar{\epsilon}^p) \\ \beta = \alpha - \beta_0 \end{cases}, \tag{20}$$

where a_0, a_1, a_2, a_3 and β_0 are material parameters to calibrate. $\bar{\epsilon}^p$ is the accumulated plastic strain.

The calibration using the nonlinear least-squares solver "NL2SOL" in Dakota software [49] requires initial guesses, upper and lower bounds of each parameter. They are given in Table 6. The elasticity parameters G_0 and ν are calibrated using the first three data points of each calibration experiment. They are fixed in the later calibration of plasticity parameters a_0, a_1, a_2, a_3, β_0. The target features for calibration objectives and accuracy evaluations include data of the pressure p, the deviatoric stress q, and the volumetric strain $\varepsilon_v = \mathrm{tr}(\boldsymbol{\varepsilon})$.

B.2. SANISAND elasto-plasticity model (Section 5.2)

The model is expressed in geomechanics sign convention as in the original paper. The model adopts the nonlinear elasticity with dependence on the mean pressure p and the void ratio e,

$$\begin{cases} K = \dfrac{2(1+\nu)}{3(1-2\nu)}G \\ G = G_0 p_{at}\dfrac{(2.97-e)^2}{1+e}(\dfrac{p}{p_{at}})^{1/2} \end{cases}, \tag{21}$$

where G_0 and ν are material parameters, $p_{at} = 100$ kPa is the atmospheric pressure.

The yield surface has the shape of a small cone

$$f = \|\boldsymbol{s} - p\boldsymbol{\alpha}\| - \sqrt{2/3}pm, \tag{22}$$

where we fix m to be 0.01.

The back stress-ratio tensor $\boldsymbol{\alpha}$ evolves according to

$$\dot{\boldsymbol{\alpha}} = \dot{\lambda}(2/3)h(\boldsymbol{\alpha}_\theta^b - \boldsymbol{\alpha}), \tag{23}$$

where $\dot{\lambda}$ is the rate of the plastic multiplier, and

$$\begin{cases} h = \dfrac{b_0}{(\boldsymbol{\alpha}-\boldsymbol{\alpha}_{in}):\boldsymbol{n}} \\ b_0 = G_0 h_0(1-c_h e)(p/p_{at})^{-1/2} \\ \boldsymbol{\alpha}_\theta^b = \sqrt{2/3}[g(\theta,c)M\exp(-n^b\psi) - m]\boldsymbol{n} \\ g(\theta,c) = \dfrac{2c}{(1+c)+(1-c)\cos 3\theta} \\ \cos 3\theta = \sqrt{6}\mathrm{tr}(\boldsymbol{n}^3) \\ \boldsymbol{n} = \dfrac{\frac{\boldsymbol{s}}{p}-\boldsymbol{\alpha}}{\sqrt{2/3}m} \\ \psi = e - e_0 + \lambda_c(p/p_{at})^{\xi} \end{cases}, \tag{24}$$

where h_0, c_h, M, c, n^b, e_0, λ_c, ξ are material parameters.

The plastic flow direction is defined as

$$\begin{cases} \boldsymbol{m}^{flow} = B\boldsymbol{n} - C(\boldsymbol{n}^2 - \dfrac{1}{3}\boldsymbol{I}) + \dfrac{1}{3}D\boldsymbol{I} \\ B = 1 + \dfrac{3}{2}\dfrac{1-c}{c}g(\theta,c)\cos 3\theta \\ C = 3\sqrt{\dfrac{3}{2}}\dfrac{1-c}{c}g(\theta,c) \\ D = A_d(\boldsymbol{\alpha}_\theta^d - \boldsymbol{\alpha}):\boldsymbol{n} \\ A_d = A_0(1+ < \boldsymbol{z}:\boldsymbol{n} >) \\ \boldsymbol{\alpha}_\theta^d = \sqrt{2/3}[g(\theta,c)M\exp(n^d\psi) - m]\boldsymbol{n} \\ \dot{\boldsymbol{z}} = -c_z < -\dot{\lambda}D > (z_{max}\boldsymbol{n} + \boldsymbol{z}) \end{cases}, \tag{25}$$

where A_0, n^d, c_z, z_{max} are additional material parameters.

The initial guesses, upper and lower bounds of the above parameters for Dakota's "NL2SOL" calibration are given in Table 7. The calibration procedure is identical to that of Drucker–Prager model.

B.3. Data-driven traction–separation model (Section 5.3)

Data-driven traction–separation models use artificial neural networks (ANNs) as universal function approximators to continuous functions of various complexity on compact subsets of R^n (Universal approximation theorem, [74]).

Table 7
Initial guesses, upper and lower bounds of the material parameters for SANISAND model.

Parameter	Initial guess	Lower bound	Upper bound
G_0	1e4 kPa	5e3 kPa	2e4 kPa
ν	0.25	0.1	0.4
M	0.75	0.5	1.0
c	0.9	0.7	1.0
e_0	0.8	0.7	0.9
λ_c	0.0025	0.0001	0.005
ξ	1.0	0.8	1.2
n^b	3.0	1.0	5.0
n^d	0.5	0.01	1.0
A_0	1.0	0.5	1.5
h_0	30.0	10.0	50.0
c_h	1.0	0.5	1.5
c_z	600.0	400.0	800.0
z_{max}	2.5	1.0	5.0

Moreover, a special type of ANNs, recurrent neural networks (RNN, e.g., long short-term memory (LSTM) [75], gated recurrent units (GRU) [76,77]), can capture the functions of a time series of inputs, which is appropriate for replicating the path-dependent material behaviors in granular interfaces. The data-driven model in the example firstly uses the histories of normal, tangential, accumulated norm, and maximum experienced norm of the displacement jumps through a RNN to predict the current normal and tangential fabrics of the interface. Then these displacement jump and fabric features are input together into a second RNN to predict the current normal and tangential traction across the interface. The parameters in each RNN are calibrated with training data of the corresponding input and output features using the backpropagation. Each RNN consists of two hidden layers with 32 GRU neurons in each layer, and the output layer is a dense layer with linear activation function. All input and output data are pre-processed by standard scaling using mean values and standard deviations [78]. Each input feature contains its current value and 4 history values prior to the current loading step. Each RNN is trained for 1000 epochs using the Adam optimization algorithm [79], with a batch size of 128.

References

[1] Yannis F. Dafalias, Modelling cyclic plasticity: simplicity versus sophistication, Mech. Eng. Mater. 153178 (1984).
[2] Ben H Thacker, Scott W Doebling, Francois M Hemez, Mark C Anderson, Jason E Pepin, Edward A Rodriguez, Concepts of Model Verification and Validation, Technical report, Los Alamos National Lab., 2004.
[3] Ronaldo I. Borja, Plasticity: Modeling & Computation, Springer Science & Business Media, 2013.
[4] Yang Liu, WaiChing Sun, Jacob Fish, Determining material parameters for critical state plasticity models based on multilevel extended digital database, J. Appl. Mech. 83 (1) (2016) 011003.
[5] Maria Laura De Bellis, Gabriele Della Vecchia, Michael Ortiz, Anna Pandolfi, A multiscale model of distributed fracture and permeability in solids in all-round compression, J. Mech. Phys. Solids 104 (2017) 12–31.
[6] Eric C. Bryant, WaiChing Sun, A mixed-mode phase field fracture model in anisotropic rocks with consistent kinematics, Comput. Methods Appl. Mech. Engrg. 342 (2018) 561–584.
[7] Eric C. Bryant, WaiChing Sun, A micromorphically regularized cam-clay model for capturing size-dependent anisotropy of geomaterials, Comput. Methods Appl. Mech. Engrg. 354 (2019) 56–95.
[8] SeonHong Na, Eric C. Bryant, WaiChing Sun, A configurational force for adaptive re-meshing of gradient-enhanced poromechanics problems with history-dependent variables, Comput. Methods Appl. Mech. Engrg. 357 (2019) 112572.
[9] S.H. Na, W.C. Sun, et al., A multi-phase-field/polycrystal plasticity for rock salt: micromorphic regularized grain-boundary slip, in: 53rd US Rock Mechanics/Geomechanics Symposium, American Rock Mechanics Association, 2019.
[10] Hyoung Suk Suh, WaiChing Sun, Devin O'Connor, A phase field model for cohesive fracture in micropolar continua, Comput. Methods Appl. Mech. Engrg. 369 (2020).
[11] Ran Ma, WaiChing Sun, Computational thermomechanics for crystalline rock. part II: Chemo-damage-plasticity and healing in strongly anisotropic polycrystals, Comput. Methods Appl. Mech. Engrg. 369 (2020) 113184.
[12] Ran Ma, WaiChing Sun, FFT-based solver for higher-order and multi-phase-field fracture models applied to strongly anisotropic brittle materials, Comput. Methods Appl. Mech. Engrg. 362 (2020) 112781.
[13] Kun Wang, Waiching Sun, Simon Salager, S Na, Ghonwa Khaddour, Identifying material parameters for a micro-polar plasticity model via X-ray micro-CT images: lessons learned from the curve-fitting exercises, Int. J. Multiscale Comput. Eng. 14 (4) (2016) 389–413.

[14] Ritesh Gupta, Simon Salager, Kun Wang, WaiChing Sun, Open-source support toward validating and falsifying discrete mechanics models using synthetic granular materials—Part I: Experimental tests with particles manufactured by a 3D printer, Acta Geotech. 14 (4) (2019) 923–937.
[15] Yousef Heider, Kun Wang, WaiChing Sun, So (3)-invariance of informed-graph-based deep neural network for anisotropic elastoplastic materials, Comput. Methods Appl. Mech. Engrg. 363 (2020) 112875.
[16] Keunhwan Pack, Meng Luo, Tomasz Wierzbicki, Sandia fracture challenge: blind prediction and full calibration to enhance fracture predictability, Int. J. Fract. 186 (1–2) (2014) 155–175.
[17] BL Boyce, SLB Kramer, TR Bosiljevac, E Corona, JA Moore, K Elkhodary, CHM Simha, BW Williams, AR Cerrone, A Nonn, et al., The second sandia fracture challenge: predictions of ductile failure under quasi-static and moderate-rate dynamic loading, Int. J. Fract. 198 (1–2) (2016) 5–100.
[18] Trenton Kirchdoerfer, Michael Ortiz, Data-driven computational mechanics, Comput. Methods Appl. Mech. Engrg. 304 (2016) 81–101.
[19] Trenton Kirchdoerfer, Michael Ortiz, Data driven computing with noisy material data sets, Comput. Methods Appl. Mech. Engrg. 326 (2017) 622–641.
[20] Qizhi He, Jiun-Shyan Chen, A physics-constrained data-driven approach based on locally convex reconstruction for noisy database, 2019, arXiv preprint arXiv:1907.12651.
[21] MA Bessa, R Bostanabad, Zeliang Liu, A Hu, Daniel W Apley, C Brinson, Wei Chen, Wing Kam Liu, A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality, Comput. Methods Appl. Mech. Engrg. 320 (2017) 633–667.
[22] Yibo Yang, Paris Perdikaris, Adversarial uncertainty quantification in physics-informed neural networks, J. Comput. Phys. 394 (2019) 136–152.
[23] Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, Paris Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data, J. Comput. Phys. 394 (2019) 56–81.
[24] Burr Settles, Active Learning Literature Survey, Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.
[25] Fredrik Olsson, A literature survey of active machine learning in the context of natural language processing, 2009.
[26] Ozan Sener, Silvio Savarese, Active learning for convolutional neural networks: A core-set approach, 2017, arXiv preprint arXiv:1708.00489.
[27] Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, Animashree Anandkumar, Deep active learning for named entity recognition, 2017, arXiv preprint arXiv:1707.05928.
[28] Jia-Jie Zhu, José Bento, Generative adversarial active learning, 2017, arXiv preprint arXiv:1702.07956.
[29] Meng Fang, Yuan Li, Trevor Cohn, Learning how to active learn: A deep reinforcement learning approach, 2017, arXiv preprint arXiv:1708.02383.
[30] Xiaojin Jerry Zhu, Semi-Supervised Learning Literature Survey, Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005.
[31] Vikas Verma, Alex Lamb, Juho Kannala, Yoshua Bengio, David Lopez-Paz, Interpolation consistency training for semi-supervised learning, 2019, arXiv preprint arXiv:1903.03825.
[32] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, Colin A Raffel, Mixmatch: A holistic approach to semi-supervised learning, in: Advances in Neural Information Processing Systems, 2019, pp. 5050–5060.
[33] John F. Nash, et al., Equilibrium points in n-person games, Proc. Nat. Acad. Sci. 36 (1) (1950) 48–49.
[34] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller, Playing atari with deep reinforcement learning, 2013, arXiv preprint arXiv:1312.5602.
[35] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al., Mastering the game of go without human knowledge, Nature 550 (7676) (2017) 354–359.
[36] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al., Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017, arXiv preprint arXiv:1712.01815.
[37] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al., Grandmaster level in starcraft II using multi-agent reinforcement learning, Nature 575 (7782) (2019) 350–354.
[38] Kun Wang, WaiChing Sun, Qiang Du, A cooperative game for automated learning of elasto-plasticity knowledge graphs and models with AI-guided experimentation, Comput. Mech. (2019) 1–33.
[39] Ronald L Graham, Donald E Knuth, Oren Patashnik, Stanley Liu, Concrete mathematics: a foundation for computer science, Comput. Phys. 3 (5) (1989) 106–107.
[40] Douglas Brent West, et al., Introduction to Graph Theory, Vol. 2, Prentice hall Upper Saddle River, 2001.
[41] Jørgen Bang-Jensen, Gregory Z. Gutin, Digraphs: Theory, Algorithms and Applications, Springer Science & Business Media, 2008.
[42] Nina M. Rodriguez, Poul V. Lade, True triaxial tests on cross-anisotropic deposits of fine nevada sand, Int. J. Geomech. 13 (6) (2013) 779–793.
[43] Michael L. Littman, Markov games as a framework for multi-agent reinforcement learning, in: Machine Learning Proceedings 1994, Elsevier, 1994, pp. 157–163.
[44] Lerrel Pinto, James Davidson, Rahul Sukthankar, Abhinav Gupta, Robust adversarial reinforcement learning, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 2817–2826.
[45] M.A. Wiering, Multi-agent reinforcement learning for traffic light control, in: Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000), 2000, pp. 1151–1158.

[46] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, Thore Graepel, Multi-agent reinforcement learning in sequential social dilemmas, in: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2017, pp. 464–473.
[47] Eric Bonabeau, Agent-based modeling: Methods and techniques for simulating human systems, Proc. Natl. Acad. Sci. 99 (suppl 3) (2002) 7280–7287.
[48] Kun Wang, WaiChing Sun, Meta-modeling game for deriving theory-consistent, microstructure-based traction–separation laws via deep reinforcement learning, Comput. Methods Appl. Mech. Engrg. 346 (2019) 216–241.
[49] Brian M Adams, WJ Bohnhoff, KR Dalbey, JP Eddy, MS Eldred, DM Gay, K Haskell, Patricia D Hough, Laura P Swiler, Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.10 User's Manual, Tech. Rep. SAND2014-4633, Sandia National Laboratories, 2014.
[50] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, Xiaoqiang Zheng, Tensorflow: Large-scale machine learning on heterogeneous systems, 2015, URL https://www.tensorflow.org/. Software available from tensorflow.org.
[51] J. Eamonn Nash, Jonh V. Sutcliffe, River flow forecasting through conceptual models part I—A discussion of principles, J. Hydrol. 10 (3) (1970) 282–290.
[52] Peter Krause, D.P. Boyle, Frank Bäse, Comparison of different efficiency criteria for hydrological model assessment, Adv. Geosci. 5 (2005) 89–97.
[53] Richard H. McCuen, Zachary Knight, A. Gillian Cutter, Evaluation of the Nash–sutcliffe efficiency index, J. Hydrol. Eng. 11 (6) (2006) 597–602.
[54] Irrigation ASCE Task Committee on Definition of Criteria for Evaluation of Watershed Models of the Watershed Management Committee Drainage Division, Criteria for evaluation of watershed models, J. Irrig. Drain. Eng. 119 (3) (1993) 429–442.
[55] Julien Perolat, Bruno Scherrer, Bilal Piot, Olivier Pietquin, Approximate dynamic programming for two-player zero-sum Markov games, in: Proceedings of the 32nd International Conference on Machine Learning, 2015, pp. 1321–1329.
[56] Ming Tan, Multi-agent reinforcement learning: Independent vs. cooperative agents, in: Proceedings of the Tenth International Conference on Machine Learning, 1993, 330–337.
[57] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, Shimon Whiteson, Learning to communicate with deep multi-agent reinforcement learning, in: Advances in Neural Information Processing Systems, 2016, pp. 2137–2145.
[58] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, Raul Vicente, Multiagent cooperation and competition with deep reinforcement learning, PLoS One 12 (4) (2017) e0172395.
[59] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, Simon Colton, A survey of monte carlo tree search methods, IEEE Trans. Comput. Intell. AI Games 4 (1) (2012) 1–43.
[60] Kun Wang, WaiChing Sun, A multiscale multi-permeability poroplasticity model linked by recursive homogenizations and deep learning, Comput. Methods Appl. Mech. Engrg. 334 (2018) 337–380.
[61] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al., Ray: A distributed framework for emerging {AI} applications, in: 13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18), 2018, pp. 561–577.
[62] Xuxin Tu, José E. Andrade, Qiushi Chen, Return mapping for nonsmooth and multiscale elastoplasticity, Comput. Methods Appl. Mech. Engrg. 198 (30–32) (2009) 2286–2296.
[63] Yannis F. Dafalias, Majid T. Manzari, Simple plasticity sand model accounting for fabric change effects, J. Eng. Mech. 130 (6) (2004) 622–634.
[64] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[65] M Mozaffar, R Bostanabad, W Chen, K Ehmann, Jian Cao, MA Bessa, Deep learning predicts path-dependent plasticity, Proc. Natl. Acad. Sci. 116 (52) (2019) 26414–26420.
[66] Kun Wang, WaiChing Sun, An updated Lagrangian LBM–DEM–FEM coupling model for dual-permeability fissured porous media with embedded discontinuities, Comput. Methods Appl. Mech. Engrg. 344 (2019) 276–305.
[67] Brad L Boyce, Sharlotte LB Kramer, H Eliot Fang, Theresa E Cordova, Michael K Neilsen, K Dion, Amy K Kaczmarowski, Erin Karasz, Liang Xue, Andrew J Gross, et al., The sandia fracture challenge: blind round robin predictions of ductile tearing, Int. J. Fract. 186 (1–2) (2014) 5–68.
[68] K. Arulanandan, R.F. Scott, Project VELACS—Control test results, J. Geotech. Eng. 119 (8) (1993) 1276–1292.
[69] Radu Popescu, Jean H. Prevost, Comparison between VELACS numerical 'class A'predictions and centrifuge experimental soil test results, Soil Dyn. Earthq. Eng. 14 (2) (1995) 79–92.
[70] Kai Arulkumaran, Antoine Cully, Julian Togelius, Alphastar: An evolutionary computation perspective, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2019, pp. 314–315.
[71] W. Daniel Hillis, Co-evolving parasites improve simulated evolution as an optimization procedure, Physica D 42 (1–3) (1990) 228–234.
[72] Václav Šmilauer, Emanuele Catalano, Bruno Chareyre, Sergei Dorofeenko, Jerome Duriez, Anton Gladky, Janek Kozicki, Chiara Modenese, Luc Scholtès, Luc Sibille, et al., Yade reference documentation, Yade Doc. 474 (1) (2010).
[73] Peter A. Cundall, Otto D.L. Strack, A discrete numerical model for granular assemblies, Geotechnique 29 (1) (1979) 47–65.

[74] Kurt Hornik, Maxwell Stinchcombe, Halbert White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[75] Sepp Hochreiter, Jürgen Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[76] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, 2014, arXiv preprint arXiv:1406.1078.
[77] François Chollet, et al., Keras, 2015, https://keras.io.
[78] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[79] Diederik P. Kingma, Jimmy Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.