Physics-Informed Recurrent Neural Networks for Multiscale Damage Modeling

Shiguang Deng[1], Shirin Hosseinmardi[2], Diran Apelian[1], and Ramin Bostanabad*[2]

[1]ACRC, Materials Science and Engineering, University of California, Irvine
[2]Department of Mechanical and Aerospace Engineering, University of California, Irvine

Abstract

Computational modeling of heterogeneous materials is increasingly relying on multiscale simulations which typically leverage the homogenization theory for scale coupling. Such simulations are prohibitively expensive and memory-intensive especially when modeling damage and fracture in large 3D components such as cast alloys. To address these challenges, we develop a physics-informed deep learning model that surrogates the microscale simulations. We build this model within a mechanistic data-driven framework such that it accurately predicts the effective microstructural responses under irreversible elasto-plastic hardening and softening deformations. To achieve high accuracy while reducing the reliance on labeled data, we design the architecture of our deep learning model based on damage mechanics and introduce a new loss component that increases the thermodynamic consistency of the model. We use physics-based reduced-order models to generate the training data of our deep learning model and demonstrate that, in addition to achieving high accuracy on unseen deformation paths that include severe softening, the model can be embedded in 3D multiscale simulations with fracture. With this embedding we also demonstrate that state-of-the-art techniques such as teacher forcing result in deep learning models that cause divergence in multiscale simulations. Our numerical experiments indicate that our model is more accurate than pure data-driven models and is much more efficient than physics-based reduced-order models.

Keywords: Deep learning; multiscale modeling; damage; physics-informed surrogate; softening.

1 Introduction

Heterogeneous materials are increasing used in many engineering applications. Analyzing the behavior of such materials often relies on multiscale simulations such as the FE^2 method [1] which is a popular homogenization-based concurrent multiscale model that uses the finite element method (FEM) at two spatial scales. Despite the recent advancements in software/hardware and mechanics theory [2], the simulation of hierarchical materials via FE^2 is still prohibitively costly. Consider the multiscale model in Figure 1(a) where each integration point (IP) of the macroscale component represents a microstructure with complex local morphologies. In this model, the two-scale

*Corresponding Author: Raminb@uci.edu

spatial discretization requires a large memory storage and also results in long runtimes since the solver repeatedly iterates between the scales. These challenges are exacerbated in the presence of microstructural deformations that are path-dependent and involve damage. That is, evaluation of the microstructural responses are the primary computational bottleneck. Our goal in this paper is to address such bottlenecks by developing a deep learning (DL) model that surrogates the microstructural analyses in 3D multiscale simulations that involve plasticity and damage.

Mechanistic reduced-order models (ROMs) are attractive alternatives to expensive methods such as the FE^2 and direct numerical simulations (DNS). The main idea behind ROMs is to reduce the number of unknown variables (e.g., stresses, strains, or internal variables such as the damage parameters) while striking a balance between accuracy and efficiency. For example, the transformed field analysis method [3] and its non-uniform variant [4] employ proper orthogonal decomposition to reduce material state variables by expressing arbitrary strain fields as a subspace representation of pre-computed eigenstrains. Clustering-based ROMs reduce unknown variables by agglomerating a large number of IPs into a few clusters. For instance, the self-consistent clustering [5] and its variant the virtual clustering analysis [6] methods assume IPs with similar elastic responses behave similarly during inelastic deformations and solve incremental Lippmann-Schwinger equations to approximate the evolution of cluster-wise material responses. Deflated clustering analysis (DCA) [7] utilizes clusters to decompose both macroscale and microscale domains where macro analysis is faithfully accelerated in a deflation space while the effective microstructural responses are approximated in a coarse-graining manner where close-by IPs are presumed to share the same

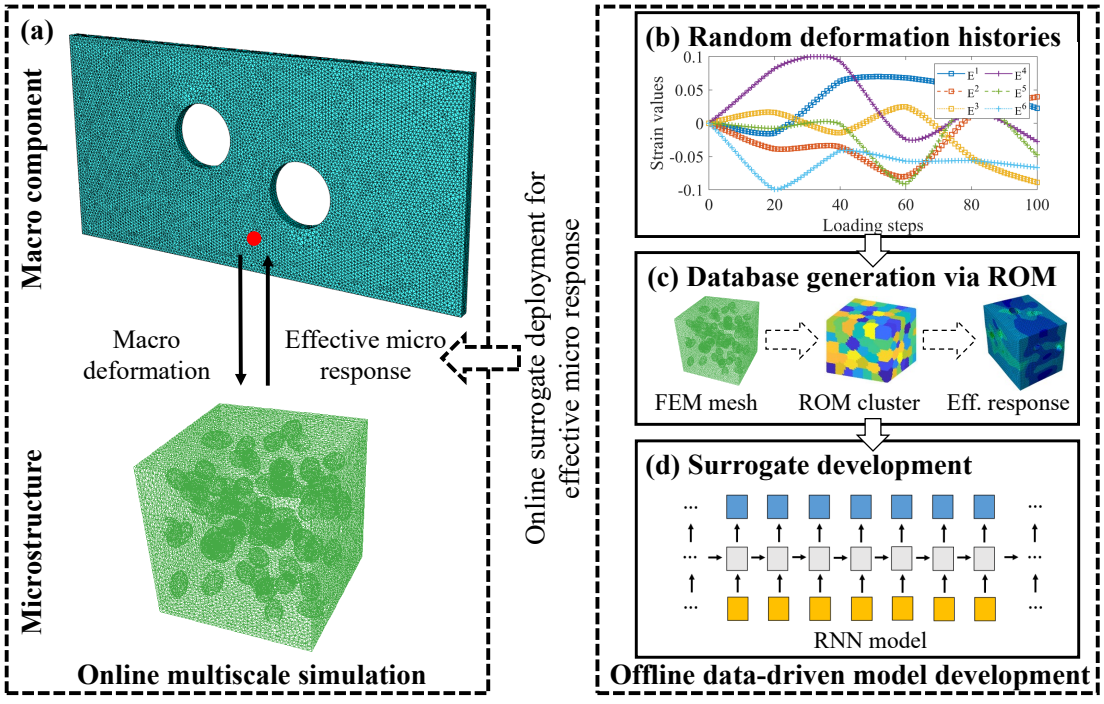


Figure 1 Multiscale simulations: (a) Each integration point (IP) of the macroscale component represents a microstructure whose effective response is needed as the solver iterates between the two scales. We build a physics-informed data-driven material model to surrogate the expensive microscale analyses amid online multiscale simulations. Our data-driven framework has three major offline components: **(b)** deformation representation, **(c)** response database generation, and **(d)** physics-informed deep learning.

behaviors. DCA's robustness and efficiency are further improved in [8] where both spatial and temporal dimensions are adaptively reduced for elasto-plastic deformations with softening. While ROMs dramatically accelerate multiscale simulations, their runtimes are still quite high (especially in the presence of softening). Additionally, ROMs lack solution transferability in that the expensive data of one instance of the model is not reused (e.g., the full strain-stress history obtained for a microstructure corresponding to a macroscopic IP is not reused in another multiscale simulation).

Machine learning (ML) provides a feasible avenue for building transferable and extremely fast material models. Among the various ML techniques such as Gaussian processes (GPs) [9–11] and polynomial chaos expansion [12], neural networks (NNs) are increasingly employed in computational solid mechanics to build data-driven material models. For instance, NNs are used to learn visco-plasticity [13], cyclic plasticity [14], interface mechanism [15], and anisotropic electrical behaviors [16]. More recently, Mianroodi et al. [17] build an NN to calculate local stress distributions in non-homogeneous microstructures with elasto-plastic behaviors. Haghighat et al. [18] incorporate the momentum balance and constitutive relations into a feed-forward NN and demonstrate the improved extrapolation capability for single scale elasto-plastic simulations. Peivaste et al. [19] develop a convolutional NN to surrogate costly phase-field simulations to learn microstructural grain evolution.

Data-driven material models are increasingly built via recurrent neural networks (RNNs) to learn path-dependent constitutive laws that govern elasto-plastic deformations. For example, Mozaffar et al. [20] successfully use an RNN to learn plasticity with distortional hardening on 2D fiber composite microstructures. Wang et al. [21] develop an RNN to link information from different scales via recursive homogenization to capture the multiscale hydro-mechanical coupling effects of heterogeneous media with various pore sizes. In these works, various methods are proposed for data generation. For instance, Wu et al. [22] design a random walk algorithm to generate a database of effective elasto-plastic hardening behaviors under cyclic and non-proportional loading paths. An on-demand sampling strategy is adopted by Ghavamian et al. [23] that reduces sampling space by running prior macro models to collect the strain-stress sequences for the subsequent RNN's learning process. This strategy reduces sampling efforts and improves prediction accuracy but reduces the generalization power since the trained model can be only applied to the macro component that is used to collect the training sequences. In a recent work [24], Logarzo et al. use an RNN to learn the hardening behavior of a 2D composite microstructure under a wide range of deformation histories that are sampled from the space of principal strains.

All aforementioned RNN surrogates are black-box or pure data-driven models whose accuracy relies on large training datasets. Building such datasets is very challenging for 3D microstructural analyses that involve softening. Since infusing physical laws into the training process can improve the reliance on data and energy consistency, we rigorously explore this direction to derive the physical constraints that must be met when surrogating elasto-plastic deformations with softening. That is, our main contribution is to develop a physics-informed RNN that surrogates the micro analyses amid online multiscale simulations that involve softening. Compared to the reviewed ROMs and pure data-driven models, our surrogate is computationally efficient, memory light, physics consistent, and transferable.

The rest of the paper is organized as follows. In Section 2, we briefly review the homogenization-based concurrent multiscale damage analysis along with the numerical techniques that improve the

convergence issues associated with simulating softening. We also develop two constraints that a generic microstructural response (under arbitrary deformations) should always satisfy to ensure thermodynamic consistency. In Section 3, we propose our physics-informed data-driven model to surrogate effective elasto-plastic microstructural responses that may involve damage and fracture. In Section 4, we illustrate the efficiency and accuracy of our data-driven model by (1) evaluating its predictions of microstructural effective responses subject to random deformation paths, and (2) embedding it in a number of multiscale structures subject to complex cyclic loading conditions with hardening and softening material behaviors. We conclude our paper with some notes on the contributions and future work in Section 5.

2 Homogenization-based multiscale damage analysis

Our multiscale damage analysis is based on the the first-order homogenization model which we review in Section 2.1. We discuss instability issues associated with strain softening in simulating damage in Section 2.2 and then present a hybrid time integration scheme in Section 2.3 to address them. To define the physical constraints that a material model must satisfy during plastic deformations, in Section 2.4 we perform an energy analysis to derive thermodynamic consistency conditions that a generic microstructure satisfies in arbitrary iso-thermal elasto-plastic deformations. We apply these conditions in Section 3 to reduce the reliance of our RNN to data while increasing its prediction accuracy.

2.1 Multiscale modeling

We use first-order computational homogenization which assumes scale separation between a macro-component and its micro-features. In solving multiscale systems, the solutions at the macroscale and microscale are coupled via the Hill-Mandel condition [25]. This condition equates the density of virtual internal work of a macroscale IP to the volume average of the virtual work in the associated microstructure subject to any kinematically admissible displacement field:

$$\boldsymbol{S}_M : \delta \boldsymbol{E}_M = \frac{1}{|\Omega_{0m}|} \int_{\Omega_{0m}} \boldsymbol{S}_m : \delta \boldsymbol{E}_m \, d\Omega \tag{1}$$

where $\boldsymbol{S}_M$, $\delta \boldsymbol{E}_M$, $\boldsymbol{S}_m$ and $\delta \boldsymbol{E}_m$ represent the macroscopic and microscopic stress and virtual strain tensors, respectively. The subscripts M and m indicate the macroscale and microscale, respectively. The : operator represents the double dot product contracting a pair of repeated indices. In addition, Ω_{0m} and $|\Omega_{0m}|$ indicate the reference microstructural domain and its volume, respectively. Following the virtual energy condition in Equation (1), the macroscopic effective stress and virtual strain can be expressed as the volume average of their micro counterparts as:

$$\boldsymbol{S}_M = \frac{1}{|\Omega_{0m}|} \int_{\Omega_{0m}} \boldsymbol{S}_m \, d\Omega; \qquad \delta \boldsymbol{E}_M = \frac{1}{|\Omega_{0m}|} \int_{\Omega_{0m}} \delta \boldsymbol{E}_m \, d\Omega \tag{2}$$

The stress and strain fields at both the macro- and micro- scales need to satisfy equilibrium equations at the corresponding length-scales. For instance, under the infinitesimal deformation assumption, the macro-solutions at the arbitrary macroscopic IP $\boldsymbol{P}$ can be computed by solving the

following boundary value problem (BVP):

$$\boldsymbol{S}_M(\boldsymbol{P}) \cdot \nabla_0 + \boldsymbol{b}_M = \boldsymbol{0} \qquad \forall \boldsymbol{P} \in \Omega_{0M} \tag{3a}$$

$$\boldsymbol{u}_M(\boldsymbol{P}) = \bar{\boldsymbol{u}}_M \qquad \forall \boldsymbol{P} \in \Gamma^D_{0M} \tag{3b}$$

$$\boldsymbol{S}_M(\boldsymbol{P}) \cdot \boldsymbol{n}_M = \bar{\boldsymbol{t}}_M \qquad \forall \boldsymbol{P} \in \Gamma^N_{0M} \tag{3c}$$

where $\boldsymbol{u}_M$ is the unknown macroscopic displacement in Ω_{0M} and $\bar{\boldsymbol{u}}_M$ is the prescribed displacement on the Dirichlet boundary Γ^D_{0M} over the undeformed macroscopic domain Ω_{0M} with an outward unit vector $\boldsymbol{n}_M$. Also, ∇_0 indicates the gradient operator with respect to the original configuration. $\boldsymbol{b}_M$ and $\bar{\boldsymbol{t}}_M$ represent the body force and prescribed surface traction on the Neumann boundary Γ^N_{0M}, respectively.

In a similar manner, the strong form of the microscale equilibrium equations can be written as a BVP for the microstructure or representative volume element (RVE) composed of micro IPs $\boldsymbol{p}$ as:

$$\boldsymbol{S}_m(\boldsymbol{p}) \cdot \nabla_0 = \boldsymbol{0} \qquad \forall \boldsymbol{p} \in \Omega_{0m} \tag{4a}$$

$$\boldsymbol{S}_m(\boldsymbol{p}) \cdot \boldsymbol{n}_m = \bar{\boldsymbol{t}}_m \qquad \forall \boldsymbol{p} \in \Gamma_{0m} \tag{4b}$$

where $\bar{\boldsymbol{t}}_m$ indicates the surface traction per unit area over the reference microstructural boundary Γ_{0m} with an outward unit normal vector $\boldsymbol{n}_m$.

2.2 Strain softening

In this work, we adopt isotropic continuum damage model to simulate strain softening in ductile metals whose load-carrying capacity drops due to the degradation of yield stress and stiffness. To simulate the onset of softening, we choose ductile damage initiation criteria which models the effective strain at damage initiation, i.e., $\bar{E}^{pl}_d$, as a function of stress and strain states. We presume $\bar{E}^{pl}_d$ is a constant and that damage begins when the equivalent plastic strain is equal or greater than it, i.e., $\bar{E}^{pl} \geqslant \bar{E}^{pl}_d$. Under progressing damage, we formulate a softening response of the ductile metal with an elasto-plastic behavior as:

$$\boldsymbol{S} = (1 - D)\boldsymbol{S}^0; \qquad \boldsymbol{S}^0 = \mathbb{C}^{el} : \boldsymbol{E}^{el} = \mathbb{C}^{el} : (\boldsymbol{E} - \boldsymbol{E}^{pl}) \tag{5}$$

where $\boldsymbol{S}$ and $\boldsymbol{S}^0$ are, respectively, the damaged stress and the reference stress that undergoes the same deformation path but in the absence of damage. $\mathbb{C}^{el}$ represents the fourth-order elasticity tensor. $\boldsymbol{E}$, $\boldsymbol{E}^{el}$ and $\boldsymbol{E}^{pl}$ are the total strain, elastic strain, and plastic strain, respectively. D represents the damage parameter that monotonically increases within $[0.0, 1.0]$. We note that in our context of isotropic continuum damage, D is a scalar and it becomes a tensor in anisotropic damage models.

A major challenge in using the isotropic continuum damage model in Equation (5) is the softening-induced non-positive definite stiffness matrix that results in slow solution convergence and negative wave speeds [26]. Specifically, the ill-posed problem causes equilibrium equations to lose objectivity with respect to mesh sizes by exhibiting spurious mesh sensitivity. We therefore adopt two different damage models to mitigate the mesh dependency at macroscale and microscale, respectively.

For macroscopic softening, we define the evolution of damage parameter D_M as a function of

$\bar{E}^{pl}$, $\bar{E}_d^{pl}$, and a user-defined non-negative damage evolution rate parameter α [27] as:

$$D_M = \begin{cases} 0; & \bar{E}^{pl} \leq \bar{E}_d^{pl} \\ 1 - \frac{\bar{E}_d^{pl}}{\bar{E}^{pl}} exp(-\alpha(\bar{E}^{pl} - \bar{E}_d^{pl})); & \bar{E}^{pl} > \bar{E}_d^{pl} \end{cases} \tag{6}$$

where no damage occurs when the plastic strain is smaller than or equal to $\bar{E}_d^{pl}$. Per Equation (6), damage initiates if $\bar{E}^{pl} > \bar{E}_d^{pl}$ and the value of D_M monotonically increases from 0.0 to 1.0 during the irreversible damage process. We constrain the progression of D_M by an integral-type non-local damage model to mitigate the spurious mesh dependency as:

$$\hat{D}_M(\boldsymbol{P}, \boldsymbol{P}') = \int_B \omega(\|\boldsymbol{P} - \boldsymbol{P}'\|) D_M(\boldsymbol{P}')\, d\boldsymbol{P}' \tag{7}$$

where $\hat{D}_M(\boldsymbol{P}, \boldsymbol{P}')$ is the non-local damage parameter at the macroscopic point $\boldsymbol{P}$ surrounded by points $\boldsymbol{P}'$ in the compact neighborhood B. $D_M(\boldsymbol{P}')$ represents the local damage parameter at $\boldsymbol{P}'$ and ω indicates the non-local weighting function which depends on the distance $\|\boldsymbol{P} - \boldsymbol{P}'\|$ between the studied point and its supporting points. In this work, we define ω by a polynomial bell-shape function as:

$$\omega(\|\boldsymbol{P} - \boldsymbol{P}'\|) = \frac{\omega_\infty(\|\boldsymbol{P} - \boldsymbol{P}'\|)}{\int_B \omega_\infty(\|\boldsymbol{P} - \boldsymbol{P}'\|)\, d\boldsymbol{P}'}; \qquad \omega_\infty(\|\boldsymbol{P} - \boldsymbol{P}'\|) = \left\langle 1 - \frac{4(\|\boldsymbol{P} - \boldsymbol{P}'\|)^2}{l_d^2} \right\rangle^2 \tag{8}$$

where $\langle \dots \rangle$ is the Macauley bracket defined as $\langle x \rangle = max(0, x)$, l_d denotes the strain localization bandwidth whose value represents the non-local interacting radius, and the support domain B is a sphere with a radius of $l_d/2$ in 3D models.

To address the lack of objectivity to mesh choices in microstructural damage simulations, we do not re-define the microscopic strain localization bandwidth since it would counteract with the physical meaning of its macroscopic counterpart l_d. Instead, we convert the stress-strain relation in the constitutive equation to a stress-displacement relation to drive the micro-damage evolution after initiation as:

$$G_f = \int_{\bar{E}_0^{pl}}^{\bar{E}_f^{pl}} l_e S_y\, d\bar{E}^{pl} = \int_0^{\bar{u}_f^{pl}} S_y\, d\bar{u}^{pl} \tag{9}$$

where l_e indicates the element characteristic length in an arbitrary RVE, and G_f represents the dissipated energy that opens a unit area of crack after damage initiation. The equivalent plastic displacement $\bar{u}^{pl}$ is the fracture work conjugate to the yield stress S_y from the onset of damage (with the effective plastic strain $\bar{E}_0^{pl}$ and zero plastic displacement $\bar{u}^{pl}$) until the final failure (with the effective fracture strain $\bar{E}_f^{pl}$ and the fracture displacement $\bar{u}_f^{pl}$). Using Equation (9) we define the damage evolution rule based on an exponential form of the released energy [28], that is:

$$D_m = 1 - exp(-\frac{1}{G_f} \int_0^{\bar{u}_f^{pl}} S_y\, d\bar{u}^{pl}) \tag{10}$$

We note that in Equation (10) D_m approaches 1.0 asymptotically with infinitely large $\bar{u}^{pl}$. In practice, we set D_m as 1.0 when the dissipated energy exceeds $0.99G_f$.

2.3 Hybrid temporal integration

The non-positive definiteness of the stiffness matrix is the primarily reason for the slow convergence of classic implicit time integration schemes that are used in continuum damage simulations. For illustration, consider the constitutive equation of an isotropic damage model integrated by an implicit backward-Euler integration scheme. Its algorithmic tangent operator at an arbitrary macroscopic IP can be written as:

$$\mathbb{C}^{alg}_{n+1} = \frac{\partial \boldsymbol{S}_{n+1}}{\partial \boldsymbol{E}_{n+1}} = (1 - D_{n+1})\mathbb{C}^{el} - \frac{S_{n+1} - H_n \bar{E}^{pl}_{n+1}}{(\bar{E}^{pl}_{n+1})^3} \boldsymbol{S}^0_{n+1} \otimes \boldsymbol{S}^0_{n+1} \tag{11}$$

where $\mathbb{C}^{alg}_{n+1}$, $\bar{E}^{pl}_{n+1}$, S_{n+1}, $\boldsymbol{S}^0_{n+1}$ and H_n represent the fourth-order algorithmic tangent operator, equivalent plastic strain, equivalent stress, referenced stress tensor, and softening modulus, respectively. The subscripts denote time steps and the symbol $\otimes$ represents the cross product between tensors. Softening causes negative values for H_n which can render $\mathbb{C}^{alg}_{n+1}$ indefinite. A non-positive $\mathbb{C}^{alg}_{n+1}$ leads to ill-conditioned elemental stiffness matrix with near-zero or negative eigenvalues, and further deteriorates the global stiffness matrix in the element assembly process. Such ill-posed matrices dramatically reduce the efficiency of iterative solvers (e.g., Newton-Raphson methods) and often cause job abortion before final convergence.

To fundamentally resolve the convergence issue, we adopt a hybrid time integration scheme [8, 29] to integrate the governing equations of elasto-plastic and softening equations explicitly-implicitly. The basic idea of the hybrid integration is to maintain the positive-definiteness of the system's algebraic tangent operator by separately integrating constitutive equations in two consecutive stages via explicit and implicit schemes. At the first stage, we explicitly extrapolate internal material state variables at time step $n+1$ from step n to compute the explicit stress state $\tilde{\boldsymbol{S}}_{n+1}$ that balances the equilibrium equation between internal and external forces. At the second stage, we compute the implicit stress state $\boldsymbol{S}_{n+1}$ based on the current strain state $\boldsymbol{E}_{n+1}$ using the classic backward Euler method to update the trial stress tensor and yield functions for the next time step where the tangent operator between $\tilde{\boldsymbol{S}}_{n+1}$ and $\boldsymbol{E}_{n+1}$ is kept positive definite.

For the elasto-plastic model, we choose the material state variable as the incremental plastic strain tensor $\triangle\tilde{\boldsymbol{E}}^{pl}_{n+1}$ such that $\tilde{\boldsymbol{S}}_{n+1}$ can be computed as:

$$\begin{gathered}\tilde{\boldsymbol{S}}_{n+1}(\triangle\tilde{\boldsymbol{E}}^{pl}_{n+1}) = \tilde{\boldsymbol{S}}^{trial}_{n+1} - \mathbb{C}^{el} : \triangle\tilde{\boldsymbol{E}}^{pl}_{n+1} = \mathbb{C}^{el} : \boldsymbol{E}_{n+1} - \mathbb{C}^{el} : \boldsymbol{E}^{pl}_{n} - \mathbb{C}^{el} : \triangle\tilde{\boldsymbol{E}}^{pl}_{n+1} \\ \triangle\tilde{\boldsymbol{E}}^{pl}_{n+1} = \frac{\triangle t_{n+1}}{\triangle t_n}\triangle\boldsymbol{E}^{pl}_{n}\end{gathered} \tag{12}$$

where $\boldsymbol{E}^{pl}_{n}$ represents the implicit incremental plastic strain tensor at time step n, $\triangle t_n$ and $\triangle t_{n+1}$ indicate the lengths of time steps at two consecutive steps. The algorithmic tangent operator (under loading[1]) is therefore computed as:

$$\tilde{\mathbb{C}}^{alg}_{n+1} = \frac{\partial \tilde{\boldsymbol{S}}_{n+1}(\triangle\tilde{\boldsymbol{E}}^{pl}_{n+1})}{\partial \boldsymbol{E}_{n+1}} = \frac{\partial(\mathbb{C}^{el} : \boldsymbol{E}_{n+1} - \mathbb{C}^{el} : \boldsymbol{E}^{pl}_{n} - \mathbb{C}^{el} : \triangle\tilde{\boldsymbol{E}}^{pl}_{n+1})}{\partial \boldsymbol{E}_{n+1}} = \mathbb{C}^{el} \tag{13}$$

[1] $\tilde{\mathbb{C}}^{alg}_{n+1}$ is equal to the elastic modulus in unloading.

In a similar manner, for isotropic continuum damage models, we choose the explicitly interpolated material state variable in the hybrid integration as the incremental plastic multiplier $\triangle\tilde{\lambda}_{n+1}$, i.e., $\triangle\tilde{\lambda}_{n+1} = (\triangle t_{n+1}/\triangle t_n)\triangle\lambda_n$. We can then write its explicit damaged stress and algorithmic tangent operator (under loading[2]) as:

$$\tilde{\boldsymbol{S}}_{n+1} = (1-\tilde{D}_{n+1})\boldsymbol{S}^0_{n+1} = (1-\tilde{D}_{n+1})\mathbb{C}^{el} : \boldsymbol{E}_{n+1}; \qquad \tilde{D}_{n+1} = \tilde{D}_{n+1}(D_n, \triangle\tilde{\lambda}_{n+1}) \tag{14}$$

$$\tilde{\mathbb{C}}^{alg}_{n+1} = \frac{\partial\tilde{\boldsymbol{S}}_{n+1}}{\partial\boldsymbol{E}_{n+1}} = (1-\tilde{D}_{n+1})\mathbb{C}^{el} \tag{15}$$

where $\boldsymbol{S}^0_{n+1}$ is the effective stress tensor, and $\tilde{D}_{n+1}$ represents the explicit state of the damage variable which is a function of its previous implicit state D_n and the current explicit incremental plastic multiplier $\triangle\tilde{\lambda}_{n+1}$.

In the hybrid integration scheme, the loading tangent operators of the elasto-plastic model in Equation (13) and the damage model in Equation (15) are trivially equal to the elastic modulus $\mathbb{C}^{el}$ and $(1-\tilde{D}_{n+1})\mathbb{C}^{el}$. Hence, the hybrid integration scheme preserves the positive-definiteness of the governing equations and also allows to assemble the global stiffness matrix only once before online simulations. The global stiffness matrix remains constant for the elasto-plastic regime and only needs partial updates on matrix entries associated with the softening IPs by Equation (15). As softening is often highly localized in small regions, the global stiffness can be incrementally updated during the entire elasto-plastic-hardening-softening process [8]; saving significant memory footprints with robust convergence performance.

2.4 Energy analysis

Assuming a microscopic IP in an RVE is subject to an iso-thermal elasto-plastic deformation, we can compute its total work rate per unit volume $\dot{W}$ via thermodynamics principles [30] as:

$$\dot{W} = \dot{\psi} + \Phi \tag{16}$$

where $\dot{\psi}$ represents the rate of Helmholtz free energy and Φ accounts for the rate of dissipated energy including the dissipation from plasticity, damage, damping, etc. For general elasto-plastic material behaviors, we can decompose the rate of work into elastic and plastic parts:

$$\dot{W} = \dot{W}^{el} + \dot{W}^{pl} \tag{17}$$

where the elastic work rate $\dot{W}^{el}$ at an arbitrary microscopic IP is equal to the rate of recoverable elastic free energy or strain energy $\dot{\psi}^{el}$, while the plastic work rate $\dot{W}^{pl}$ is equal to the sum of the conditionally recoverable plastic free energy $\dot{\psi}^{pl}$ and the irrecoverable dissipation rate [31]. That is:

$$\dot{W}^{el} = \dot{\psi}^{el} = \boldsymbol{S}_m : \dot{\boldsymbol{E}}^{el}_m; \qquad \dot{W}^{pl} = \dot{\psi}^{pl} + \Phi = \boldsymbol{S}_m : \dot{\boldsymbol{E}}^{pl}_m \tag{18}$$

We write the total rate of work per unit volume $\dot{W}$ of an RVE as the multiplication of the micro stress $\boldsymbol{S}_m$ and the rate of microscopic total strain $\dot{\boldsymbol{E}}_m$:

$$\dot{W} = \boldsymbol{S}_m : \dot{\boldsymbol{E}}^{el}_m + \boldsymbol{S}_m : \dot{\boldsymbol{E}}^{pl}_m = \boldsymbol{S}_m : \dot{\boldsymbol{E}}_m \tag{19}$$

[2] $\tilde{\mathbb{C}}^{alg}_{n+1} = (1-\tilde{D}_{n+1})\mathbb{C}^{el}$ in unloading.

where we use the additive decomposition rule for the strain. We compute the total work by integrating the work rate over the time interval and spatial domain Ω_{0m}. Additionally, we compute the total work by invoking the Hill-Mandel energy condition from Equation (1) while assuming the strain rates are in the hyperspace of the virtual strains:

$$\begin{aligned} W &= \int_t \int_\Omega \dot{W}\, d\Omega_{0m}\, dt = \int_t \int_\Omega \boldsymbol{S}_m : \dot{\boldsymbol{E}}_m\, d\Omega_{0m}\, dt \\ &= |\Omega_{0m}| \int_t \boldsymbol{S}_M : \dot{\boldsymbol{E}}_M\, dt = |\Omega_{0m}| \int_t \boldsymbol{S}_M\, d\boldsymbol{E}_M \end{aligned} \tag{20}$$

We now decompose this total work to its components to find the constraints that the effective macroscale stress and strain fields should satisfy. We write W as a summation of total strain energy W^{el} and total plastic work W^{pl}:

$$W = W^{el} + W^{pl} \tag{21}$$

Assuming linear elasticity, we can show the total strain energy of the RVE as:

$$\begin{aligned} W^{el} &= \int_t \int_\Omega \boldsymbol{S}_m : \dot{\boldsymbol{E}}_m^{el}\, d\Omega_{0m}\, dt = \int_\Omega \int_{\boldsymbol{E}_m^{el}} \boldsymbol{S}_m\, d\boldsymbol{E}_m^{el}\, d\Omega_{0m} \\ &= \frac{1}{2} \int_\Omega \boldsymbol{E}_m^{el} : \mathbb{C}_m^{el} : \boldsymbol{E}_m^{el}\, d\Omega_{0m} \geq 0 \end{aligned} \tag{22}$$

where $\mathbb{C}_m^{el}$ represents the elastic modulus at a micro IP. W^{el} in Equation (22) is equal to $(1-D_m)\mathbb{C}^{el}$ if damage occurs (with a micro damage parameter D_m) and $\mathbb{C}^{el}$ if there is no damage. Since $0 \leq D_m \leq 1$, it is straight forward to show $W^{el} \geq 0$.

Similarly, we can compute W^{pl} by spatiotemporally integrating $\dot{W}^{pl}$, and it is equal to the sum of total dissipated energy and total plastic free energy as:

$$W^{pl} = W^{di} + W^{pf} \tag{23}$$

where W^{di} can be expressed as the spatiotemporal integration of the non-negative dissipation rate:

$$W^{di} = \int_t \int_\Omega \Phi\, d\Omega_{0m}\, dt \geq 0 \tag{24}$$

where the non-negativity is due to the fact that $\Phi \geq 0$. The total plastic free energy equals the integrated rate of plastic free energy:

$$W^{pf} = \int_t \int_\Omega \dot{\psi}^{pl}\, d\Omega_{0m}\, dt = \int_\Omega \psi^{pl}\, d\Omega_{0m} \tag{25}$$

where ψ^{pl} stands for the density of the plastic free energy in the RVE and it can be decomposed into isotropic and anisotropic parts [31] as:

$$\psi^{pl} = \psi_{iso}^{pl} + \psi_{ani}^{pl}; \qquad \psi_{ani}^{pl} = \psi_{kin}^{pl} - \psi_{dis}^{pl} \tag{26}$$

where ψ^{pl}_{iso}, ψ^{pl}_{ani}, ψ^{pl}_{kin}, and ψ^{pl}_{dis} represent the constituents of plastic free energy density from isotropic, anisotropic, kinematic, and distortional deformations, respectively (ψ^{pl}_{dis} is related to the distortional strain hardening with directional distortion of the yield surface but exploring this relation is not in the scope of this work). We can calculate ψ^{pl}_{iso} and ψ^{pl}_{kin} via [32]:

$$\psi^{pl}_{iso} = \frac{c_1}{2\rho}\bar{k}^2; \qquad \psi^{pl}_{kin} = \frac{c_2}{2\rho}\bar{\alpha}_{ij}\bar{\alpha}_{ij} \tag{27}$$

where $\bar{k}$ and $\bar{\alpha}_{ij}$ are, respectively, the thermodynamic conjugates to the size of the yield surface and the deviatoric back stress tensor that represents the center of the yield surface, and ρ is the material density. c_1 and c_2 are two non-negative material constants depending on the type of material models. We can therefore express the total plastic free energy as:

$$W^{pf} = \int_{\Omega} (\frac{c_1}{2\rho}\bar{k}^2 + \frac{c_2}{2\rho}\bar{\alpha}_{ij}\bar{\alpha}_{ij})\, d\Omega_{0m} \geq 0 \tag{28}$$

By plugging Equations (22, 24, 28) into Equation (20), we show that for an arbitrary macroscale IP associated with an RVE that is subject to a generic deformation with hardening and softening, the effective macroscale stress and strain fields satisfy the following constraint:

$$\int_t \boldsymbol{S}_M \, d\boldsymbol{E}_M \geq 0 \tag{29}$$

We incorporate this constraint together with the non-decreasing damage variable described in Section 2.2 as the two physics constraints into the data-driven model in Section 3.

3 Physics-informed Data-driven Surrogate

To reduce the computational costs of multiscale simulations that involve hardening and softening, we follow the data-driven framework that is schematically illustrated in Figure 1 (b)-(d). Our framework uses the following three modules to build a physics-informed material model that surrogates the nested microstructural analyses. The first two modules create the training dataset for the material model and the last module builds an RNN using the generated data and domain knowledge.

- **Module 1: Exploration of the deformation space.** Deformation paths are extremely high-dimensional as they have a sequential nature. To efficiently explore such a high-dimensional space, we utilize random processes and design of experiments (DoE). This exploration is an extremely important step because in a multiscale simulation each macro IP (and hence its corresponding RVE) undergoes a unique deformation path that depends on the IP's location, the material properties, the applied macro boundary and initial conditions, and the geometry of the macro component.

- **Module 2: Response collection.** The first major computational bottleneck of our framework is obtaining the microstructural responses to the deformation paths generated in Module 1. These high costs are primarily associated with simulating softening and we address them by using ROMs which leverage hybrid time integration to avoid softening-induced solver divergence.

- **Module 3: Physics-informed surrogate modeling.** Fitting an accurate surrogate to the generated training data is a challenging and time-consuming process since the generalization power[3] of the RNN strongly depends on its architecture, training data size, and the training mechanism. We use domain knowledge to dramatically reduce the sensitivity of the RNN to these factors such that it can be used as a transferable constitutive law in a wide range of multiscale simulations.

We describe these three modules in more details below. In Section 3.1 we elaborate on the data generation process which builds a set of independent and systematically sampled microstructural deformation-response sequences. This dataset is then used in Section 3.2 to train an RNN that serves as the data-driven material model at the microscale. To improve this model's accuracy on unseen deformation paths, we incorporate two physics constraints in Section 3.3. In Section 3.4, we show the integration procedure of our surrogate in multiscale solvers.

3.1 Design of Experiments

While using domain knowledge reduces the reliance on training data, building a transferable[4] RNN is still a data intensive and computationally expensive task since thousands of costly samples are needed. Hence, it is important to ensure that the training data provides as much information as possible by maximally exploring the deformation space. Below, we first define this space and then describe our sampling mechanism.

In the DoE, we assume every strain path starts from a relaxing state (with zero initial strains and no residual stresses) and evolves to a final state by n_{load} load steps. Additionally, we presume that (1) the maximum strain value in any direction and at any load step is smaller than the user-defined threshold ζ_1, and (2) our material's bulk modulus is fairly large such that the deformation-induced material volume change is within the user-defined limit ζ_2. We use these two practical constraints to reduce the sampling space and we express them as:

$$|E_n^i| \leq \zeta_1; \qquad |E_n^{vol}| \leq \zeta_2 \tag{30}$$

where E_n^i represents the i^{th} component of the strain vector at load step $n \in \{1, 2, ..., n_{load}\}$, $i \in \{1, 2, 3, 4, 5, 6\}$ indicates the six components of 3D strains in which $i = \{1, 2, 3\}$ designate normal strains and $i = \{4, 5, 6\}$ represent shear strain components. We note that $E_n^{vol} = E_n^1 + E_n^2 + E_n^3$ is the volumetric strain standing for the material volume change after deformation.

Without loss of generality, we require all deformation paths to have n_{load} load steps. We characterize each dimension of a load path[5] with n_c evenly spaced control points whose strain values are sampled via a space-filling algorithm such as the Sobol sequence. To obtain the strain values at all n_{load} load steps, we use a one-dimensional zero-mean GP to interpolate the values assigned to the control points. The correlation function of this GP is:

$$r(n, n') = exp(-w(n - n')^2) \tag{31}$$

[3]The accuracy of the surrogate in predicting the microstructural response given deformation paths that are not seen in training.

[4]By Transferable we mean an RNN that can be used as the constitutive law at all macro IPs in a wide range of multi-scale simulations.

[5]In $3D$, a load path consists of 6 strain sequences where each sequence is of length n_{load}.

where n and n' are two load steps, w is the scale parameter that controls the roughness of the interpolated curve, and the exponent 2 ensures that the generated path is differentiable. In our dataset, we change the value of w across the DoE samples to increase the variability in the generated paths. That is, we use a single Sobol sequence to generate both w and the values of all strain components at all control points.

To ensure the values of all six strain components at the control points satisfy the two constraints in Equation (30), we generate a large DoE via the Sobol sequence (which is extremely fast) and then select n_p valid points from it. Specifically, we choose the independent variables whose bounds are known as the DoE dimensions, i.e., E_c^{vol} and E_c^i with $i \in \{1, 2, 4, 5, 6\}$. Then, we find the third normal strain for the control point c via $E_c^3 = E_c^{vol} - E_c^1 - E_c^2$.

We create a total of n_p deformation paths where each one represents the temporal evolution of the six independent strain components. We plot ten example strain paths in Figure 2 which demonstrates that the random shear strains span the entire hypercube-shaped deformation space constrained by ζ_1 while the normal strain components are additionally confined between the two hyper-planes that represent the volumetric strain constraint defined by ζ_2. We also plot the 2D projections of the random strain sequences in Figure 2 to show that the normal and shear components start from the relaxing state without any strain values. In addition, we observe that the highly complex deformation histories consist of multiple loading-unloading-reloading cycles.

After we generate the random strain paths, we use ROMs to compute the microstructural effective responses for each strain sequence. Specifically, we impose the microstructural displacement boundary conditions by the affine boundary condition as:

$$\boldsymbol{u}_m(\boldsymbol{p}) = \boldsymbol{E}_M \Delta \boldsymbol{p} \qquad \forall \boldsymbol{p} \in \Gamma_{0m} \tag{32}$$

where the microstructural displacement boundary condition $\boldsymbol{u}_m$ depends on the macro strain tensors $\boldsymbol{E}_M$ (generated from GP interpolations) and the relative coordinates $\Delta \boldsymbol{p}$ of the nodes on the microstructural boundary Γ_{0m}. From this BVP, we solve the microstructural local stress $\boldsymbol{S}_m$, and compute the effective stress $\boldsymbol{S}_M$ via Equation (2). Additionally, we compute the RVE's effective damage parameter [27] by:

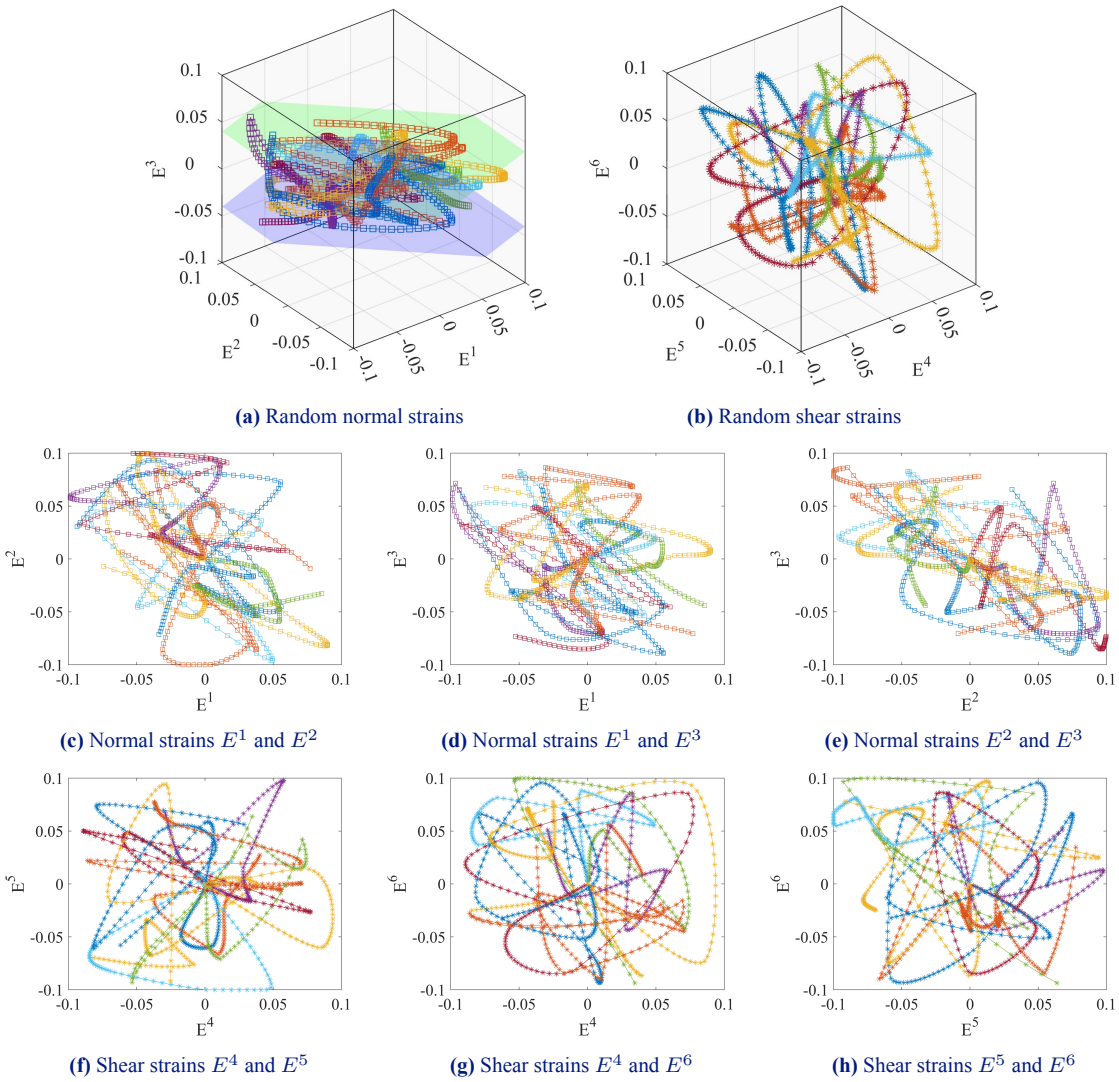
$$D_M = 1 - \frac{\|\boldsymbol{S}_M : \boldsymbol{S}_M^0\|}{\|\boldsymbol{S}_M^0 : \boldsymbol{S}_M^0\|} \tag{33}$$

where the homogenized damage parameter D_M indicates the damage status of the RVE. Its value depends on the values of the effective stress $\boldsymbol{S}_M$ and the reference stress $\boldsymbol{S}_M^0$ without damage as in Equation (5).

3.2 Pure Data-Driven Surrogate

In this section we first review the working principles of RNNs and then explain how to use the generated data in Section 3.1 to train a pure data-driven RNN (in Section 4.1.2 we compare the accuracy of this RNN to the physics-informed RNN of Section 3.3). We also discuss the limitations of excluding domain knowledge from the training process.

RNN is a special type of NN that is designed to learn from sequences (e.g., time series data) which cannot be accurately learned by basic NNs such as feed-forward neural networks (FFNNs,



(a) Random normal strains

(b) Random shear strains

(c) Normal strains E^1 and E^2

(d) Normal strains E^1 and E^3

(e) Normal strains E^2 and E^3

(f) Shear strains E^4 and E^5

(g) Shear strains E^4 and E^6

(h) Shear strains E^5 and E^6

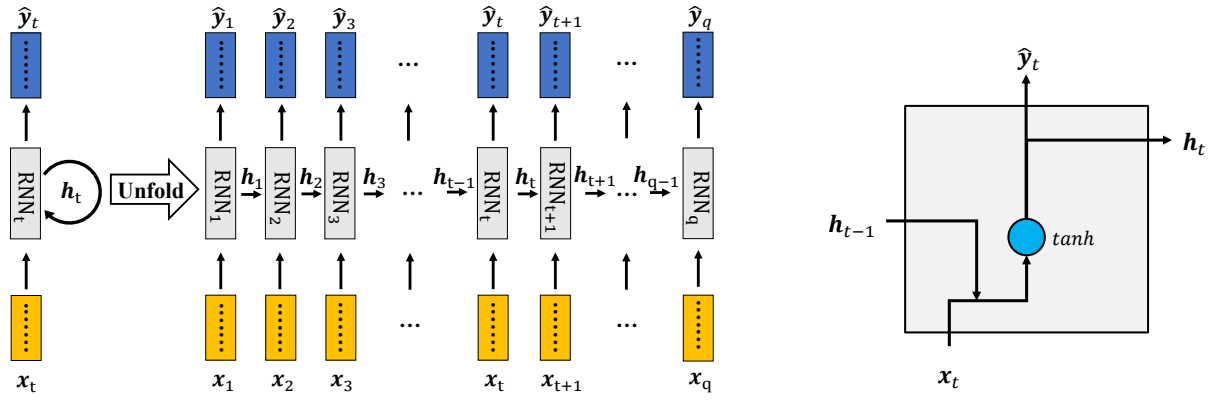
Figure 2 Example deformation paths in 3D simulations: Ten random strain paths are illustrated. The normal and shear strain components are shown in (**a**) and (**b**), respectively. The $2D$ projections of these strain paths are provided in (**c**)-(**e**) for normal components and in (**f**)-(**g**) for shear components.

aka multi-layer perceptrons). FFNNs are a collection of neurons arranged in multiple layers such that each neuron has one-way connections to the neurons of the subsequent layer. In an FFNN, each neuron performs a relatively simple mathematical operation where a (typically) nonlinear activation function is applied to the summation of a bias term and a weighted sum of the neuron's inputs as [33]:

$$\boldsymbol{x}^l = \boldsymbol{f}\left(\boldsymbol{W}^l \boldsymbol{x}^{l-1} + \boldsymbol{b}^l\right) \tag{34}$$

where $\boldsymbol{f}$ is a vector of activation functions (e.g., hyperbolic tangent, rectified linear unit or ReLU, leaky ReLU, or swish), $\boldsymbol{x}^{l-1}$ are the outputs of the previous layer's neurons, $\boldsymbol{W}^l$ and $\boldsymbol{b}^l$ are the weight matrix and bias vector of the layer l, respectively, and $\boldsymbol{x}^l$ are the outputs of layer l.

Operations such as the one in Equation (34) cannot efficiently learn from sequences as their structure does not have a mechanism to leverage the past in predicting the current response. RNNs



(a) Folded and unfolded RNN

(b) Vanilla RNN cell

Figure 3 Computational graph of pure data-driven RNN: **(a)** Folded and unfolded representations of an RNN that maps a sequence of inputs to a sequence of outputs. $\boldsymbol{h}_t$ is the state variable and captures the effects of the past on the present. **(b)** Internal structure and data flow in an RNN cell at time instance t where a hyperbolic tangent function maps the weighted current inputs and hidden variables from the previous time step to the current outputs and hidden variables.

[34] address this issue via the so-called state variables[6] that capture the effects of the past events (i.e., past inputs/outputs in the sequence) on the current response. To demonstrate this, we draw an RNN cell and its equivalent unfolded computational graph in Figure 3(a) where the RNN cell relates the input sequence $\boldsymbol{x}_t$ to a series of outputs $\boldsymbol{y}_t$ with t representing the pseudo-time (equivalent to our load steps). The mathematical operations that an RNN cell performs at time step t are schematically demonstrated in Figure 3(b) and read as:

$$\boldsymbol{h}_t = \tanh(\boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{xh}\boldsymbol{x}_t + \boldsymbol{b}_h) \tag{35a}$$

$$\hat{\boldsymbol{y}}_t = \boldsymbol{W}_{hy}\boldsymbol{h}_t + \boldsymbol{b}_y \tag{35b}$$

where the hidden state $\boldsymbol{h}_t$ depends on the current input state $\boldsymbol{x}_t$ and the previous hidden state $\boldsymbol{h}_{t-1}$. In addition, $\boldsymbol{W}_{xh}$, $\boldsymbol{W}_{hh}$, and $\boldsymbol{W}_{hy}$ are weighting matrices corresponding to input-to-hidden, hidden-to-hidden, and hidden-to-output affine transformations, respectively. $\boldsymbol{b}_h$ and $\boldsymbol{b}_y$ are the bias terms associated with $\boldsymbol{h}_t$ and estimated outputs, respectively.

As indicated in Figure 3, regardless of the sequence length, an RNN always has the same input size because it is specified in terms of transition from one state to another state rather than being specified in terms of a variable-length history of states. That is, the same transition function with the same parameters is used at every time step. This parameter sharing provides RNNs with major advantages over FFNNs in sequence learning.

RNNs are data-intensive models especially when learning complex and long sequences. They also suffer from numerical issues such as vanishing and exploding gradients [35] that prevent the RNN from learning long-range dependencies. To address these issues, more advanced cells such as long short-term memory (LSTM) [36] and gated recurrent unit (GRU) [37] are developed. In this work, we employ GRU cells and review their structure in Appendix B.

[6] Interestingly, state variables are also used in classical constitutive laws such as the over-stress tensor that is used when modeling multi-axial large-strain kinematic hardening.

While GRU cell are more efficient than vanilla RNN cells, they still need large training data to achieve *sufficient* accuracy. This sufficiency condition is driven by our application, that is, our RNN should be accurate enough such that a multiscale simulation converges to the ground truth when the RNN is used as the microstructural material model at all macro IPs. In addition to being data-intensive, the performance of RNNs is sensitive to factors such as the architecture and training mechanism (e.g., batch size, learning rate, regularization weight, etc.). To reduce the reliance on data and sensitivity to these factors, in Section 3.3 we propose to use domain knowledge in designing the architecture and loss function of the RNN.

3.3 Physics-Informed Surrogate

The samples in our training data are expensive since obtaining the microstructural response under a deformation path requires running $3D$ elasto-plastic simulations with damage. Hence, we cannot afford building a very large training dataset which, in turn, challenges building an accurate RNN. To address this issue, we leverage domain knowledge to design the loss function and architecture of our RNN. Our additions allow to reduce the number of trials that an analyst has to test before finding a good model. This reduction is particularly advantageous since training RNNs relies on back-propagation through time (BPTT) which cannot be parallelized (as the forward propagation graph is inherently sequential, i.e., each time step can only be predicted after the previous one is already computed) and is also memory intensive (since all the values computed in the forward pass must be stored until they are reused during the backward pass).

3.3.1 Architecture

In a multiscale simulation, the RNN is expected to predict the current stress tensor $(\boldsymbol{S}_t)$ and damage variable (D_t) given the history of the strain tensor $(\boldsymbol{E}_0, \cdots, \boldsymbol{E}_t)$. That is, a sequence of strain tensors (up to and including the current load step) is the input of the RNN while the current stress tensor and damage variable are the RNN's outputs.

As schematically shown in Figure 4 we make several important changes to the pure data-driven RNN of Section 3.2 which directly learns the relation between $(\boldsymbol{E}_0, \cdots, \boldsymbol{E}_t)$ and $(\boldsymbol{S}_t, D_t)$. Specifically, we append the outputs of the RNN cells with two FFNNs and choose $(\boldsymbol{S}^0_t, D_t)$ as the outputs, that is, we use the effective reference stresses (which exclude the effect of damage) as opposed to the damaged stresses. In addition, we hard-code the physical requirement on the damage variable into the network such that the predictions are always positive and non-decreasing for any arbitrary deformation path that is fed to the network.

The rationale behind learning $\boldsymbol{S}^0_t$ instead of $\boldsymbol{S}_t$ is that it forces the state variables of the RNN to learn hardening (characterized via the effective reference stresses) and softening (characterized by the damage variable) in a somewhat decoupled manner. The damaged stresses are then obtained by combining the two output pairs in the FFNNs as (the hats indicate that the variables are predicted by the network):

$$\hat{\boldsymbol{S}}_t = (1 - \hat{D}'_t)\hat{\boldsymbol{S}}^0_t \tag{36}$$

which is also used in continuum damage mechanics, see Equation (5). The advantage of this decoupling or hard-coding the RNN to learn $(\boldsymbol{S}^0_t, D_t)$ is that it reduces the network size as the network no longer needs to internally learn the relation in Equation (36). Given that we cannot build a

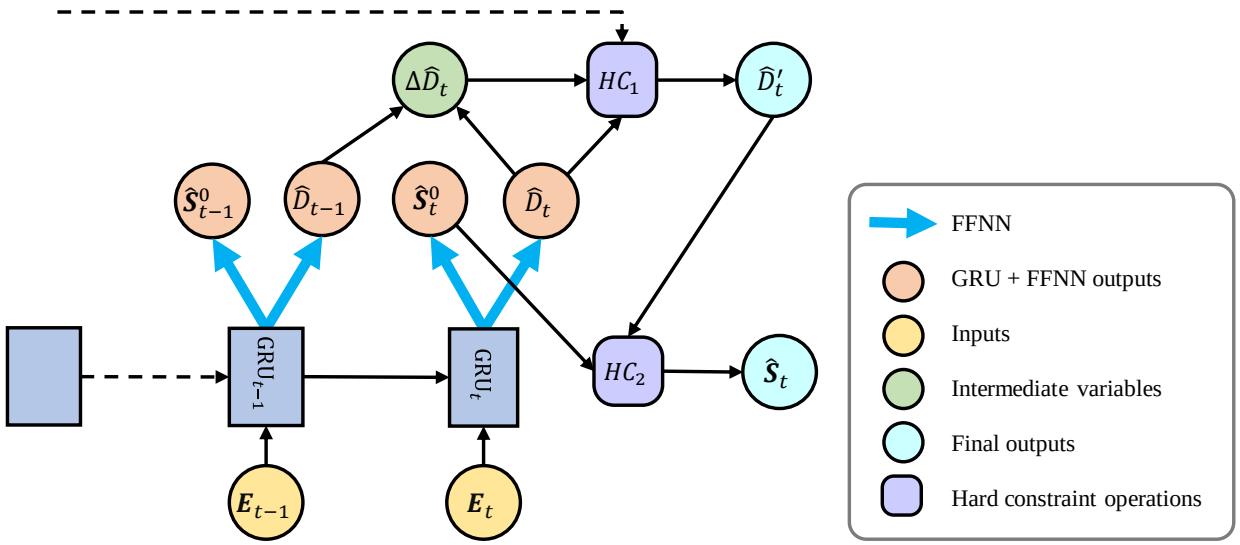


Figure 4 Computational graph of physics-constrained RNN architecture: In this architecture, we illustrate the GRU cells at different time instances with their corresponding inputs and outputs. Intermediate variables are obtained from GRU+FFNN outputs and then used to impose hard constraints. HC_1 corrects $\hat{D}_t$ to be non-decreasing, and HC_2 computes the damaged stress $\hat{\boldsymbol{S}}_t$ using the undamaged stress $\hat{\boldsymbol{S}}^0_t$ and the corrected effective damage parameter $\hat{D}'_t$. In addition, we use solid arrow lines to represent the data flow directly associated with the last two time steps shown in this figure and dash lines to represent the data flow from the previous time steps.

particularly large training dataset, small networks are preferred in our application.

Since our material does not heal itself during the irreversible damage process, $\hat{D}_t$ should be non-decreasing along any deformation path for an arbitrary macro IP. We mathematically represent this requirement as:

$$\dot{D}_t = \frac{\partial D_t}{\partial t} \geq 0 \tag{37}$$

where $\dot{D}_t$ is the damage rate, and D_t is the effective macro damage parameter at time step t that can be computed from the RVEs' homogenized damaged and reference stresses, see Equation (33). Equation (37) is not necessarily satisfied by the pure data-driven network in Figure 3(a) and thus we develop an efficient numerical scheme to explicitly enforce it. To this end, we first use $\hat{D}_t$ and $\hat{D}_{t-1}$ to compute the damage increment $\triangle\hat{D}_t$ and then update the damage parameter via the following scheme that ensure the damage increments are always non-negative:

$$\hat{D}'_t = \hat{D}_t + \sum_{\tau=1}^{t} \left(\Delta\hat{D}_\tau \times \left(0.5 \times \text{sign}(\Delta\hat{D}_\tau) - 0.5 \right) \right) \quad \forall t \in \{1, 2, \ldots, n_{load} - 1\} \tag{38}$$

where $\Delta\hat{D}_\tau = \hat{D}_\tau - \hat{D}_{\tau-1}$, $\hat{D}'_0 = \hat{D}_0 = 0$, and $\hat{D}'_t$ indicates the corrected effective damage parameter which is then used in Equation (36) to compute the damaged stresses. The predicted $(\hat{\boldsymbol{S}}_t, \hat{D}'_t)$ are then compared to the ground truth as described in Equation (40) to minimize the loss function that is described next.

3.3.2 Loss Function

We design a composite loss function that is minimized via mini-batch stochastic gradient descent. The first component of this loss is the reconstruction error which at the arbitrary time instance $t \in \{1, 2, \ldots, n_{load}\}$ is calculated as:

$$l_t^0 = \frac{1}{d_{out}} \frac{1}{n_b} \sum_{b=1}^{n_b} \|\boldsymbol{y}_t^b - \hat{\boldsymbol{y}}_t^b\|_2 \tag{39}$$

where d_{out} is the dimension of outputs, $\|\cdot\|_2$ indicates the l^2 norm of vectors, n_b is the size of the mini-batch, and $\boldsymbol{y}_t$ and $\hat{\boldsymbol{y}}_t$ represent the ground truth and predicted values, respectively. In our case, $\boldsymbol{y}_t = (\boldsymbol{S}_t, D_t)$ and hence $d_{out} = 7$.

The second part of our loss function follows Equation (29) of Section 2.4 which indicates that the total internal work at an arbitrary macro IP can be computed from its associated RVE's homogenized stress and strain tensors, and that its value should be non-negative at any time instance:

$$l_t^1 = \frac{1}{n_b} \sum_{b=1}^{n_b} ReLU(-\sum_t (\hat{\boldsymbol{S}}_t^b : \Delta \boldsymbol{E}_t^b)) \tag{40}$$

where we approximate the total internal work at time step t by summing incremental internal works (which are computed by the predicted current stress $\hat{\boldsymbol{S}}_t^b$) and the incremental strain $\Delta \boldsymbol{E}_t^b$ in a training batch, i.e., $\Delta \boldsymbol{E}_t^b = \boldsymbol{E}_t^b - \boldsymbol{E}_{t-1}^b$. We use the rectified linear unit (ReLU) defined as $ReLU(x) = max(x, 0) \geq 0$ in Equation (40) to ensure the total loss is not penalized if the internal work is positive.

Following the above, we define the RNN's composite loss function as:

$$\mathcal{L} = \sum_{t=1}^{n_{load}} l_t; \qquad l_t = l_t^0 + \lambda l_t^1 \tag{41}$$

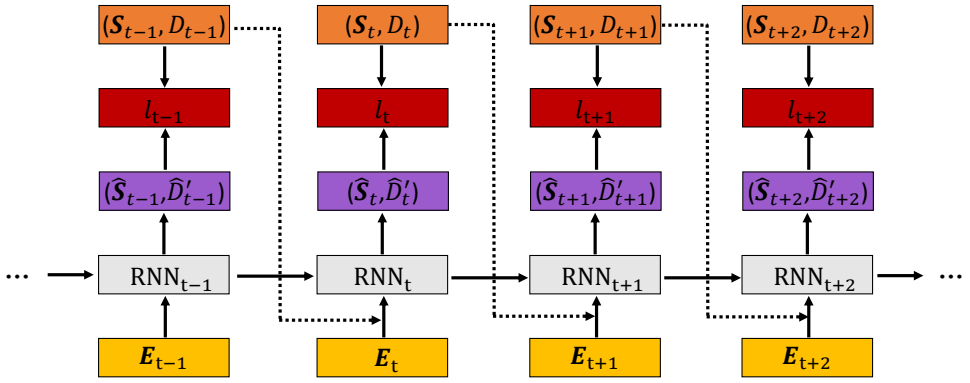
where λ is a scalar that controls the contributions of l_t^1 to the overall loss. To estimate an appropriate value for λ, we train a few RNNs and choose the one whose corresponding RNN achieves the smallest error on validation data. In this work, we use $\lambda = 10^{-6}$. A more systematic way to determine λ is through adaptive weighting study [38].

As described above we enforce the two physics constraints within our RNN architecture by using two different approaches. While we implement the energy constraint in Equation (40) as a soft constraint by adding an associated penalty term in the loss function, we enforce the damage constraint in Equation (37) as a hard constraint by imposing architectural modifications and post-processing RNN cells' outputs by using the intermediate variables in the network. While the hard constraint always guarantees that the required conditions are met, it may lead to a stiffer optimization problem [39] in training. Additionally, hard constraints require architectural modifications which may be difficult to implement for complex physical conditions. In our studies, the architecture in Figure 4 resulted in more accurate models compared to cases where the constraint in Equation (37) was enforced by penalizing the loss function.

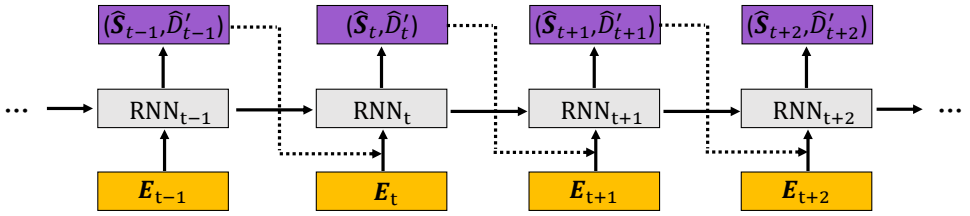
3.3.3 Teacher Forcing

The networks presented so far have recurrent connections between the output at one time step and the hidden units at the next time step. Compared to models with hidden-to-hidden connections, these networks are less powerful since their outputs should capture all of the information about the past that is needed in predicting the current state. One technique for addressing this issue is teacher forcing [40] which refers to networks whose outputs are fed back into the model via recurrent connections. Figure 5 schematically demonstrates such a model with one look-back step where only the previous output is fed back into the model (more look-backs are possible).

As illustrated in Figure 5, training and testing are done slightly differently in the presence of teacher forcing. At the offline training stage, we provide the ground truth (RVE's effective stresses and damage variable) at the previous time step as inputs at the next time step. At the online testing stage, we must use the predictions at the previous time step since the ground truth is unavailable.



(a) Teacher forcing during training



(b) Teacher forcing during testing

Figure 5 Teacher forcing with one look-back step: (**a**) In training, the current input is augmented with the ground truth from the last time step. (**b**) During testing, the inputs are augmented with the predictions from the previous time step.

Teacher forcing reduces the training time and provides smaller closed-loop reconstruction errors. However, networks with teacher forcing typically struggle in open-loop applications where the ground truth is unavailable. While some training techniques (such as intentionally corrupting the outputs before they are fed back into the model) alleviate this issue to some extent, large errors can still appear. As we demonstrate in Section 4, using an RNN in a multiscale simulation is indeed an open-loop application where teacher forcing provides poor performance.

3.4 Surrogate Integration with Multiscale Solvers

Our trained RNN surrogates the computationally expensive micro-solver in a multiscale simulation. The online deployment of the RNN surrogate within an iterative solver poses some difficulties. During training the RNN has access to the deformation and effective response histories at all n_{load} load steps. Comparatively, in the online computations only the previous strains and responses are available and even the converged macro strains at the current load step are unknown (in conventional numerical solvers these macro strains are found iteratively by solving the equilibrium equations, see Section 2.1). We address these difficulties by explicitly modifying the input sequences and implicitly resetting RNN's hidden variables amid iterations.

We embed our trained RNN in a multiscale model per the pseudo-code in Algorithm 1 which primarily aims to integrate the RNN with the Newton-Raphson method. In the presence of nonlinear deformations, the Newton-Raphson method is typically used to iteratively solve for the material's path-dependent responses. This method essentially consists of a double-loop structure where the outer loop incrementally steps along the applied load while the inner loop aims to match internal and external forces by iteratively updating the material response under a fixed loading condition.

Algorithm 1: Integration of RNN in multiscale analysis

$i = 1, 2, \ldots, n_{load}$; `/* Newton step number */`
$j = 1, 2, \ldots, n_{iter}$; `/* Newton iteration number */`
$k = 1, 2, \ldots, n_{mip}$; `/* Number of macroscale IPs */`
$\epsilon = 10^{-6}$; `/* Convergence criterion */`
while $i \leq n_{load}$ **do**
  while $j \leq n_{iter}$ **do**
    (1) Read macro strain $\boldsymbol{E}_i^j$ from macro equilibrium equation
    (2) Append $\boldsymbol{E}_i^j$ to the convergent strain sequence $\{\boldsymbol{E}_1^c, \boldsymbol{E}_2^c, \ldots, \boldsymbol{E}_{i-1}^c, \boldsymbol{E}_i^j\}$
    (3) Add (n_{load} - i) replicate padding of $\boldsymbol{E}_i^j$ to the end of the sequence in (2)
    while $k \leq n_{mip}$ **do**
      (4) Perform RNN inference on the updated strain sequence for each macro IP
    end
    (5) Retrieve RNN's outputs for the effective responses at the step i
    (6) Solve macro equilibrium equation
    if $\|\boldsymbol{f}_{int} - \boldsymbol{f}_{ext}\| \leq \epsilon$ **then**
      Update convergent strain $\boldsymbol{E}_i^c = \boldsymbol{E}_i^j$
      Continue to the next load step: $i \leftarrow i + 1$
      Break ; `/* Iteration convergence */`
    else
      Continue to the next iteration: $j \leftarrow j + 1$
    end
  end
end

In a typical step of a multiscale simulation, we compute the macro strain tensor at an arbitrary IP from equilibrium equations within the inner loop. By appending the strains at the current iteration

to the sequence of previous convergent strains, the length of the strain sequence equals the current load step number which that is smaller than n_{load} which is the required length of the RNN's input sequence. To address this mismatch, we repeat or pad the value of the current strain multiple times at the end of the strain sequence. This padding only makes the strain sequence compatible with the RNN's input, but also implicitly freezes the RNN's hidden variables at the current step within the iteration (inner) loop. This freezing happens because the values of the current hidden variables are decided by the state of network parameters and the inputs from previous time instances, see Equation (35). This freezing is akin to the classic radial return algorithm in plasticity models where material state variables are only updated upon convergence. We also emphasize that the number of load steps in the multiscale simulation should be smaller than or equal to the sequence length of RNN inputs (that is n_{load}) as a larger step number results in data truncation during input data preparation and erroneous RNN inference (in our case, n_{load} is chosen large enough to prevent truncation).

4 Numerical experiments

In this section, we first illustrate the efficacy of our physics-informed RNN in predicting microstructural effective behaviors in Section 4.1 where we assume micro porosity as the only material defects. We then perform the computation of multiscale elasto-plastic hardening and softening simulations in Section 4.2 by integrating our RNN (as a surrogate of microstructural analysis model) with a macro FEM solver. In Section 4.3, we deploy our multiscale surrogate model to perform a mesh convergence study on a component with different spatial discretization levels to simulate macro damage patterns. In the experiments, we record computational costs and perform accuracy analysis.

We build our RNN via TensorFlow in Python. We generate the database of microstructural effective responses on a state-of-the-art high-performance cluster (HPC) which has 60 CPU cores (AMD EPYC processors) and 360 GB RAM. We train the RNN model on the HPC via two GPU units (NVIDIA Tesla v100) with 32 GB RAM. For multiscale simulations, we develop a dedicated program to integrate our RNN model within a multiscale analysis engine which is implemented in Matlab. We note that all data-driven multiscale computations in Sections 4.2 and 4.3 are conducted on a 64-bit Windows desktop with four CPU cores (Intel i7-3770) and 16 GB RAM.

4.1 Surrogate for microscale damage analysis

We assume the material is aluminum alloy A356 (with elastic modulus of $5.7e4$ MPa and the Poisson's ratio of 0.33) whose isotropic elasto-plastic hardening behavior follows an associative plastic flow rule with the Von Mises yield surface defined by:

$$S \leq S_y(\bar{E}^{pl}) \tag{42}$$

where S and S_y are, respectively, the Mises equivalent stress and the yield stress which depends on the equivalent plastic strain $\bar{E}^{pl}$. To model strain hardening, we assume the relation between S_y and $\bar{E}^{pl}$ is piecewise-linear as shown by the hardening curve in Figure 6. For modeling softening, we employ the damage continuum model discussed in Section 2.2 with the fracture strain E_f of 0.067 and the fracture energy G_f of $1.92e4$ N/m.

4.1.1 Database generation of RVE effective responses

We solve micro BVPs on a microstructure whose geometry and mesh are illustrated in Figure 7(a). The microstructure is made out of A356 and contains a spherical pore at its center. Even though classic FEM with sufficiently fine mesh, e.g., see Figure 7(b), can provide high-fidelity solutions to BVPs, it is generally expensive, especially for the response database generation. To improve computational efficiency, we apply mechanistic DCA-based ROM [7, 8, 41] to solve the BVPs. Compared to FEM, our ROM strikes a good balance between efficiency and accuracy by agglomerating elements into clusters, e.g., see Figure 7(c) where material IPs in the same cluster are assumed to share identical elasto-plastic behaviors.

We note that a mesh independence study is often required in material softening simulations to choose a proper spatial discretization for solution convergence [26, 41], see Section 2.2. We conduct the microscale mesh convergence investigation in Appendix A where we systematically compare the softening behaviors between FEM and ROM for the RVE in Figure 7(a) and show that our ROM with $1,200$ clusters predicts consistent post-failure behaviors as the FEM while being 10 times more faster. Hence, we choose the ROM with $1,200$ clusters to build our data base and consider this ROM as the ground truth when validating the data-driven surrogates in the following experiments.

To generate the database, we set the sampling constraint for the strain magnitude as $\zeta_1 = 10\%$ and the constraint of the volumetric strain as $\zeta_2 = 4\%$. For the GP interpolation, we set the number of control points with random strain values as $n_c = 5$. Our database contains a total of $n_p = 30,000$ deformation paths and RVE effective responses where each path includes six strain components, six effective stress components and one effective damage variable at $n_{load} = 101$ sequential loading steps. Generating this database costs about ten-day computational time on the HPC by exploiting parallel computing with 60 CPU cores.

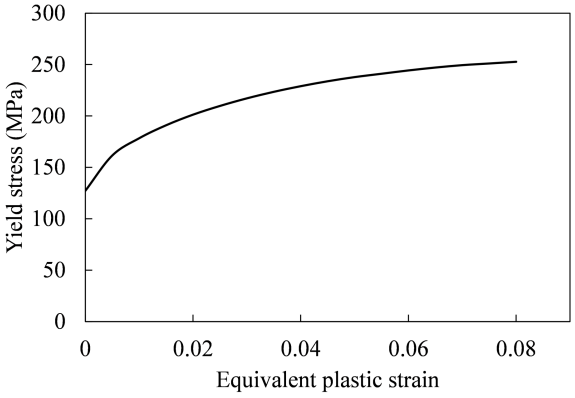


Figure 6 Elasto-plastic hardening behavior: We use a piecewise linear hardening model to define the elasto-plastic behavior of aluminum alloy A356.

4.1.2 Impacts of physics constraints

To demonstrate the impacts of the two physics constraints in Section 3.3, we compare the prediction accuracy of a pure data-driven vanilla model against our RNN model. For this comparison, we randomly choose 200 deformation-responses sequences as a test set. We further randomly select $6,000$, $12,000$, $18,000$, $24,000$ and $29,800$ sequences from the database to form five different training-validation datasets. For all training-validation datasets, we split them into 80% for training set and 20% for validation set. For example, the dataset of the size of $6,000$ has $4,800$ sequences for training and $1,200$ for validation. We point out the training and validation sets serve different purposes, as the training set is used to iteratively update learning parameters during BPTT, while the validation set is used to detect overfitting or underfitting.

During training, we normalize all data sequences and use $1,200$ epochs with a batch size of $n_b = 64$. We choose *Adam* as the optimizer with an adaptive learning rate that starts at 10^{-3} and reduces by 25% when the validation error is not reduced over 30 training epochs. We terminate the training process when the training reaches the maximum number of epochs or the loss function is not improved by 10^{-7} over 50 epochs. We use mean squared error (MSE) to measure accuracy:

$$\text{MSE} = \frac{1}{n_t n_{load} d_{out}} \sum_{m=1}^{n_t} \sum_{t=1}^{n_{load}} \sum_{i=1}^{d_{out}} \left(y_{i,t}^m - \hat{y}_{i,t}^m\right)^2 \tag{43a}$$

$$\text{MSE}_\text{S} = \frac{1}{n_t n_{load} d_S} \sum_{m=1}^{n_t} \sum_{t=1}^{n_{load}} \sum_{i=1}^{d_S} \left(S_{i,t}^m - \hat{S}_{i,t}^m\right)^2 \tag{43b}$$

$$\text{MSE}_\text{D} = \frac{1}{n_t n_{load}} \sum_{m=1}^{n_t} \sum_{t=1}^{n_{load}} \left(D_t^m - \hat{D}_t^m\right)^2 \tag{43c}$$

where MSE accounts for the total prediction error including both stress and damage predictions, while MSE_S and MSE_D are the prediction error for stress and damage, respectively. n_t and d_{out} are the number of data sequences in the test set and the dimension of outputs. $y_{i,t}^m$ and $\hat{y}_{i,t}^m$ are the ground truth and prediction for the i^{th} output component at the t^{th} load step for the m^{th} test sample. In addition, d_S, $S_{i,t}^m$, $\hat{S}_{i,t}^m$, D_t^m and $\hat{D}_t^m$ are the number of 3D stress components, true stress, predicted stress, true damage and predicted damage variables, respectively, i.e., $y_{i,t}^m = (S_{i,t}^m, D_t^m)$. We note that the values of n_t, d_{out} and d_S are equal to 200, 7 and 6, respectively. Furthermore, we set the penalty parameter as $\lambda = 10^{-6}$ in the customized loss function in Equation 41.

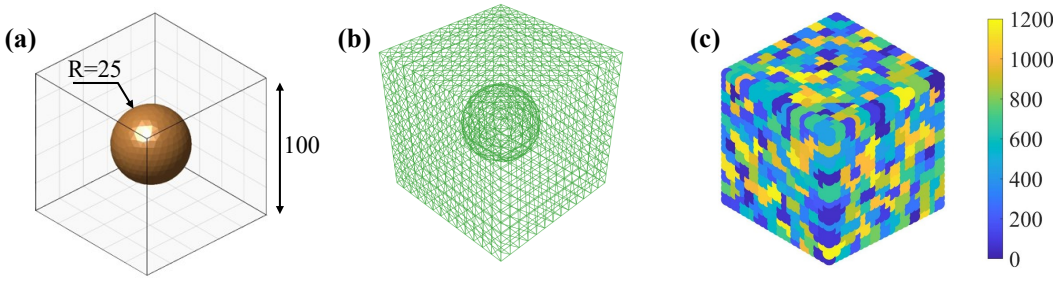


Figure 7 The geometry, dimension and mesh of our RVE: (**a**) The dimension (unit: μm) of our RVE that contains a spherical pore at center with a pore volume fraction of 6.25%; (**b**) The RVE is discretized by $15,000$ finite elements; and (**c**) Our ROM with $1,200$ clusters.

After the 10 models are trained on the five different training-validation datasets, we compare their prediction errors on the same test set that is unseen amid the entire training process, see Figure 8 and Figure 9. For the overall MSE, we find that as the sizes of training-validation datasets increase from $6,000$ to $29,800$, the MSEs of both models decrease dramatically from about 7×10^{-5} to 3×10^{-5}. It is clear that the overall MSE of our proposed model is always lower than that of the vanilla model.

To provide more insights into the performance of two models across the different data set sizes, we report the MSEs that each model achieves in terms of stresses and damage variable, see Figure 9 and Equation (43). It is evident that the proposed model provides a higher accuracy than the vanilla model. Particularly, on the smaller datasets ($6,000$ or $12,000$ sequences) with a limited amount of training data, the prediction error of our physics constrained RNN is about 60% lower than the vanilla model. As the size of the training data increases, we observe that the gap decreases. A similar trend is observed for the damage variable except that the gap in the performance is smaller.

To visualize the predicted effective responses by our physics-constrained surrogate model, we randomly select four strain paths from the testset and compare our predictions against the ground truth, see Figure 10. We observe that our RNN provides accurate predictions for all effective stresses and the damage variable even though the deformation paths are quite complex. In particular, we observe that as the damage variable increases to 1.0 amid material deformation, the magnitudes of the effective stresses are correspondingly reduced (which indicates that the RVE's load-carrying capacity significantly decreases).

4.1.3 Impacts of teacher forcing

As discussed in Section 3.3, teacher forcing, which augments ground truth or predictions from previous steps to the input at the current step during training, can provide an effective way for improving the accuracy of RNNs. To quantify the impact of teacher forcing, we compare the total MSE of the predicted stress and damage variables over the test dataset between our physics-

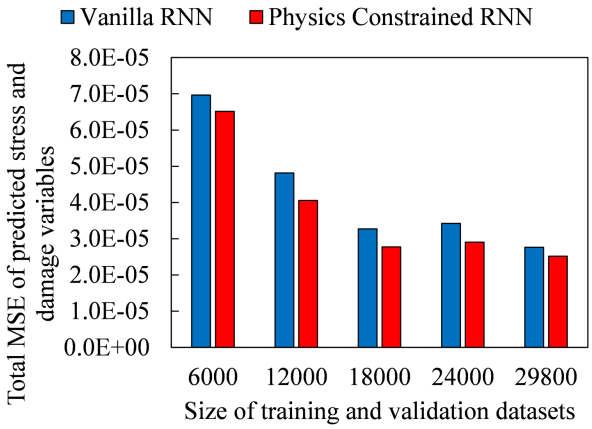
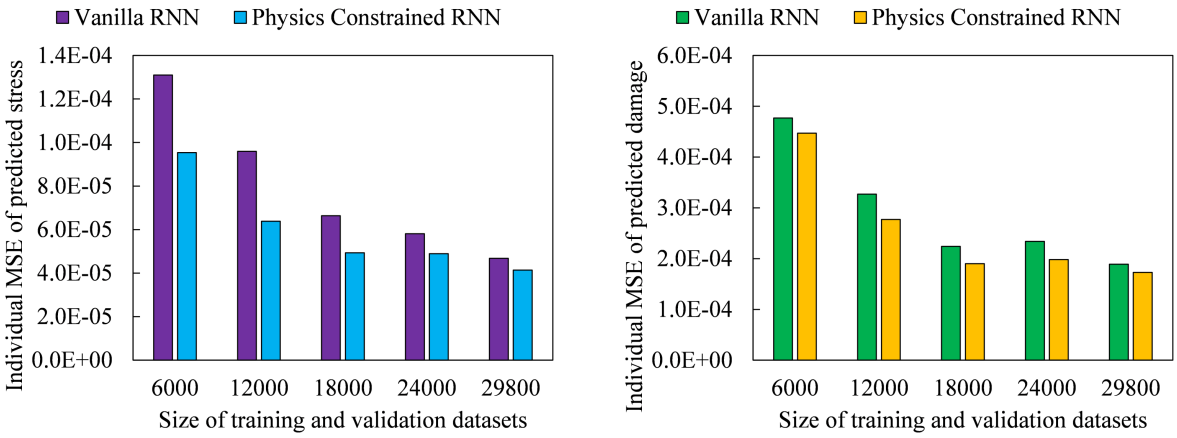


Figure 8 Convergence study: We compare the total MSE between the pure data-driven model and the proposed physics constrained RNN model to demonstrate the effectiveness of using constraints and domain knowledge in designing the RNN. The errors are computed on the same test dataset which has 200 deformation-response sequences.



(a) MSE comparison of stress variables

(b) MSE comparison of damage variables

Figure 9 MSE on predicted stress and damage: We compare the vanilla RNN and our physics constrained RNN models based on MSE in stresses and damage variables.

Table 1 Impacts of the teacher forcing on single-scale predictions: Comparison of the total MSE and individual MSEs (10^{-5}) of the predicted effective stress and damage variables over the test set between our physics constrained RNN model (trained by $29,800$ sequences) without teacher forcing and the same model with teacher forcing using one or five look back steps.

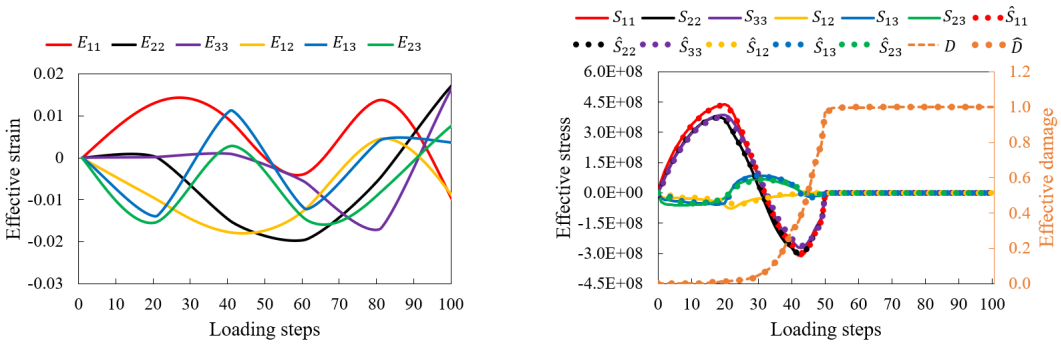
	No teacher forcing	Teacher forcing (NLB=1)	Teacher forcing (NLB=5)
Total MSE	2.52	1.98	1.91
MSE_S	4.14	4.62	4.54
MSE_D	17.30	9.67	9.24

constrained RNN model with and without teacher forcing technique in Table 1.

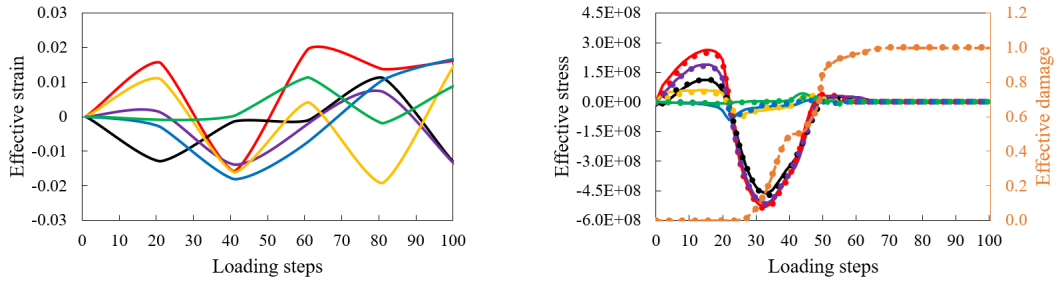
We implement two teacher forcing models here: the first model with the number of look back step (NLB) of one, and the second model with the NLB of five. From Table 1, we observe that compared to the model without teacher forcing, the two teacher forcing models reduce the total MSE by 21.4% (NLB=1) and 24.2% (NLB=5), respectively. Comparing the individual MSEs, we find that teacher forcing reduces the prediction error of effective damage while not for the stress. Therefore, in single scale RVE simulations, the teacher forcing improves our RNN's overall prediction accuracy. However, as we show next, teacher forcing significantly reduces the performance in multiscale simulations.

4.2 Surrogate for multiscale damage analysis

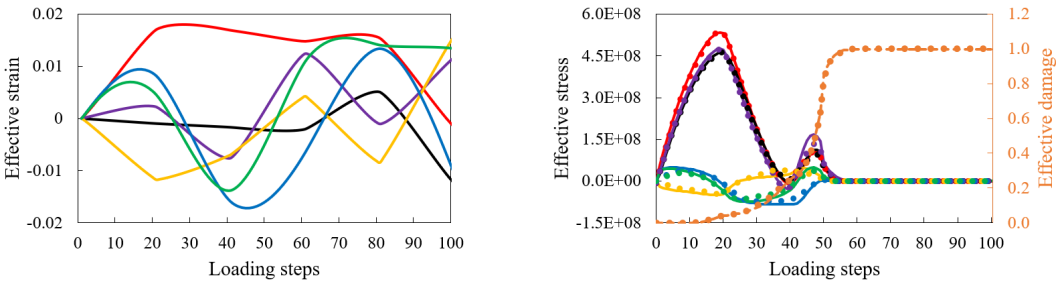
We design a 3D L-shape bracket for our multiscale simulation, Figure 11(a). The bracket is subject to a Dirichlet boundary condition on the left side while its right surface is fully fixed. We assume the bracket contains a multiscale domain around the sharp corner where we expect strain concentrations to occur. Specifically, we associate each IP of the multiscale domain with a porous RVE as illustrated in Figure 7. To save computational costs, we assume there is a mono-scale domain outside the multiscale domain where the IPs are not associated to any RVEs. We mesh the



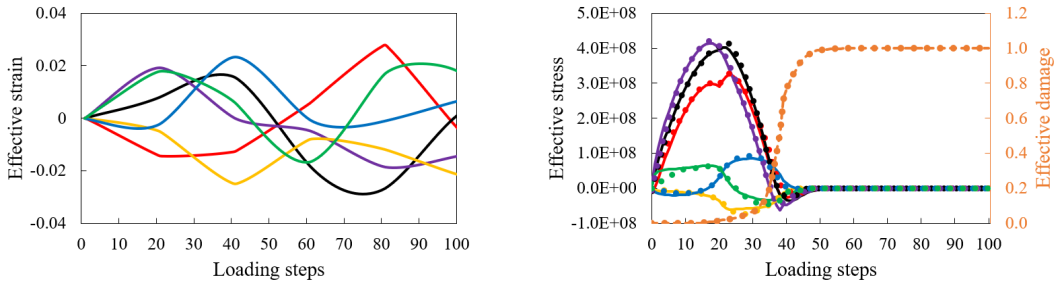
(a) Random strains and effective responses of test sample 1



(b) Random strains and effective responses of test sample 2



(c) Random strains and effective responses of test sample 3



(d) Random strains and effective responses of test sample 4

Figure 10 RVE deformation and responses from RNN vs ground truth: We demonstrate four test samples with different strain paths (the first column where each path contains six strain components with 100 steps) and their associated effective stresses and damage variable(the second column).

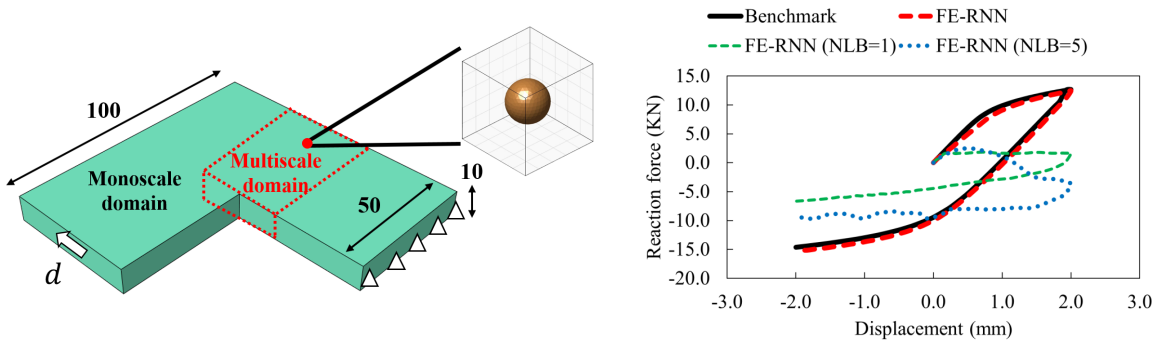
bracket by $5,300$ tetrahedral elements of reduced integration. The multiscale domain contains 360 elements each of which is associated with an RVE that is decomposed by $1,200$ clusters.

We first demonstrate the accuracy of the multiscale model for elasto-plastic hardening behaviors under complex cyclic loading histories. To this end, we subject the bracket to a loading-unloading-reloading condition by setting the Dirichlet boundary condition as $d = 0 \rightarrow 2 \rightarrow 0 \rightarrow -2$ mm. We compare the resulting reaction force and displacement curves between our proposed FE-RNN approach and the benchmark FE-ROM method in Figure 11(b).

As shown in Figure 11(b) we observe that teacher forcing, either with one or five look back steps, provides erroneous results which are due to the fact teachers forcing leverages the historical predictions (i.e., stresses and damage variable predicted for previous load steps) in estimating the current stresses and damage variables. However, these historical predictions are highly noisy since they suffer from errors that are (1) accumulated: since erroneous predictions are fed back into the model and propagated forward (see Figure 5) , and (2) compound: since the predictions at any macro IP affect the predictions at other IPs. Following these results, we adopt our FE-RNN model without teacher forcing for all multiscale simulations in the following experiments (note that this multiscale simulation is different from the single-scale study in Section 4.1.3 where we use different RNNs to surrogate the effective responses of an RVE).

We compare the Von-Mises stress distributions between the benchmark and our FE-RNN model by setting the boundary condition as $d = -2$ mm, see Figure 12. We observe a generally good agreement between the two models despite minor local discrepancy at the sharp corner. A plausible reason for the discrepancy is that RNN's prediction accuracy decreases for extreme values with insufficient training data points or poor extrapolation ability.

In our second multiscale experiment we simulate the elasto-plastic hardening and softening of the same L-shape bracket where its Dirichlet boundary condition is set as $d = 10$ mm. To prevent the occurrence of the non-physical single-layer fracture bands as discussed in Section 2.2, we apply a non-local damage function (see Equation 7) with a strain localization bandwidth of $l_d = 15$ mm, see Figure 13(a) for a comparison between l_d and the mesh size of the bracket. The force-displacement curves are compared in 13(b) where the general trends of the two methods match well especially in



(a) Geometry, dimensions (unit:mm) and boundary conditions

(b) Reaction force and displacement

Figure 11 Multiscale model of the L-shape bracket: (**a**) Every macroscale IP in the multiscale domain is associated with a microscale porous RVE; and (**b**) Comparison of the load-displacement curves of the elasto-plastic hardening behaviors between benchmark and the proposed FE-RNN models with and without teacher forcing.

the hardening section. Minor discrepancy manifests in the softening regime where the data-driven model tends to break earlier which underestimates the component's load-carrying capacity by about 2.5%. The underlying reason is that softening behaviors dramatically increase the complexity of the material's governing equations and our training data has much fewer information on softening than on hardening.

We now compare the distributions of damage variables and Von-Mises stresses when the boundary condition is set as $d = 10$ mm, see Figure 14 and Figure 15, respectively. We see that both field variables have good agreements between the two approaches. In Figure 14, we observe that the fracture bands initiate from the sharp corner and stretch towards the right surface. We can also clearly see the effects of imposing non-local damage functions in avoiding non-physical single-layer fracture bands. As for the stress distributions in Figure 15, both approaches show that the local stress values are significantly reduced within fracture bands that indicates a loss of load-carrying capacity in the fractured elements. We observe minor discrepancy of local stresses at the front tip of the fracture bands between the two methods: while the benchmark indicates relatively low stresses at the highlighted region, our FE-RNN model suggests stress concentrations which triggers more damage if the component is further deformed. These stress concentrations explain why our data-driven model predicts an earlier damage occurrence than the benchmark in Figure 13(b).

The discrepancy between the proposed model and benchmark can be further quantified by the histogram of errors as shown in Figure 16. In terms of damage variables, it is quite clear from Figure 16(a) that the two approaches yield identical solutions in the majority (more than 80%) of elements. Based on the distributions of stress errors in Figure 16(b), we can see relatively large errors in about 2% of all elements. It should be noted, however, for most elements, their absolute errors are smaller than about 10% between the two methods.

To quantify computational efficiency, we break down the costs of different steps in this multi-scale model as shown in Table 2. Comparing to the mechanistic models (FE-ROM and FE^2), our data-driven model (FE-RNN) requires additional costs on database generation and model training. Even though expensive, we only need to perform the two steps once, and after training we can deploy the trained RNN model for any multiscale simulation without extra costs. In terms of the online clock time, our model shows superior efficiency to the mechanics models (FE-ROM and

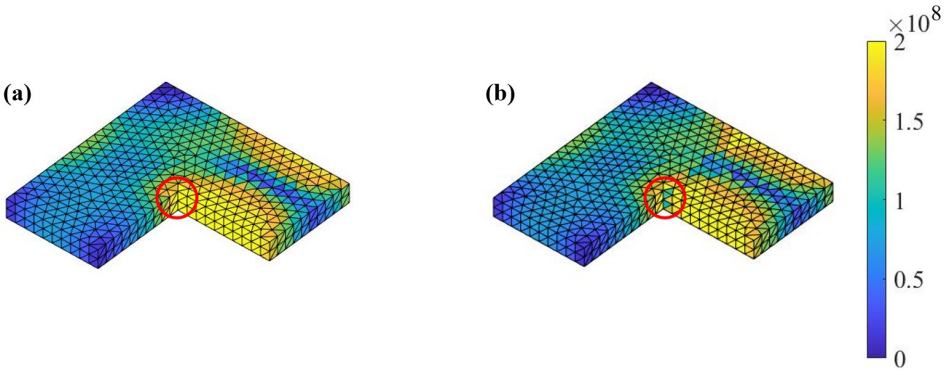
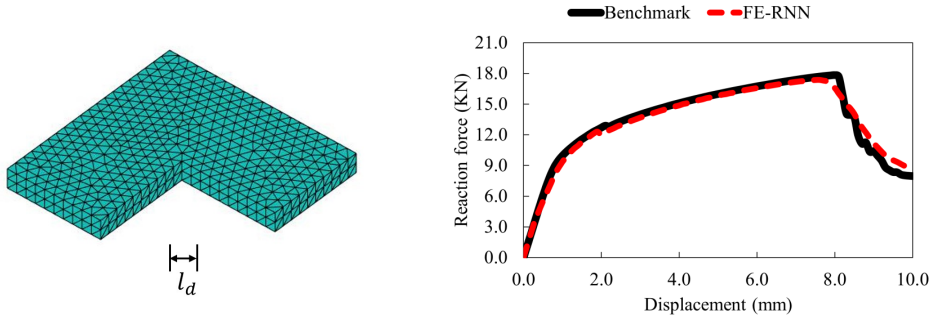


Figure 12 Comparison of Von-Mises stress distributions in hardening simulation: (**a**) The ground truth distribution of Von-Mises stresses (unit: Pa) and (**d**) Von-Mises stresses by the proposed FE-RNN model.



(a) Macroscale mesh and strain localization bandwidth (l_d)

(b) Reaction force and displacement

Figure 13 Damage simulations of the L-shape bracket: (**a**) The macroscale discretization and strain localization bandwidth applied in the damage function; and (**b**) Comparison of the softening load-displacement curves between benchmark and the proposed FE-RNN model.

FE^2) with about $125\times$ and $1240\times$ accelerations, respectively. It is noted that we do not directly perform the FE^2 due to its prohibitively demanding costs, its computational time is estimated by comparing to the ROM on a smaller multiscale simulation whose time comparison is demonstrated in Figure A.3 of Appendix A. We also note that, while we perform the training process on two GPU processors, we carry out both the database generation of the FE-RNN and the multiscale simulations of FE-ROM and FE^2 by paralleling 60 CPU cores with 360 GB RAM on an HPC. Comparatively, our proposed RNN model only needs four CPU cores on a desktop computer for the multiscale computation, providing feasible solutions to the engineers without accessibility to large computational resources.

4.3 Mesh convergence study by multiscale damage surrogate

One of the major challenges of using continuum mechanics to simulate softening is preventing the fracture bands from residing in single-element-wide layers. A popular solution to this challenge is to apply non-local functions to constrain damage patterns to different spatial discretization levels. To this end, we apply the proposed RNN model to a new 3D model in this section and assess its robustness in predicting damage behavior while changing the mesh size.

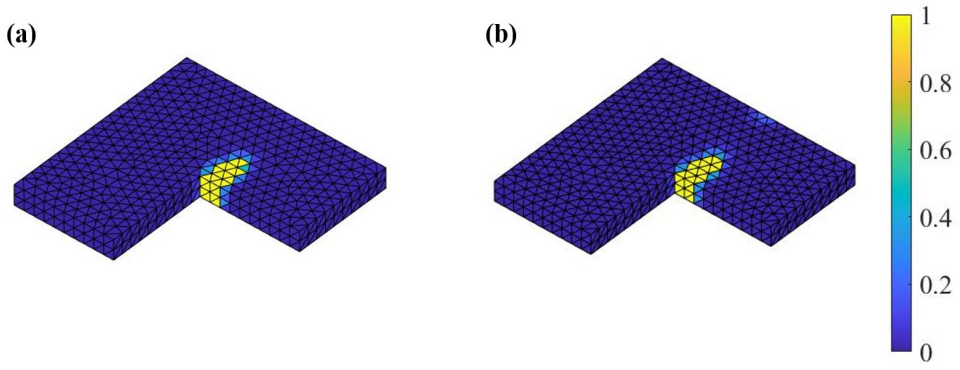


Figure 14 Comparison of damage patterns: (**a**) The distribution of benchmark damage variables; and (**d**) Damage variables via our FE-RNN model where yellow indicates a full material fracture while blue represents an intact state.

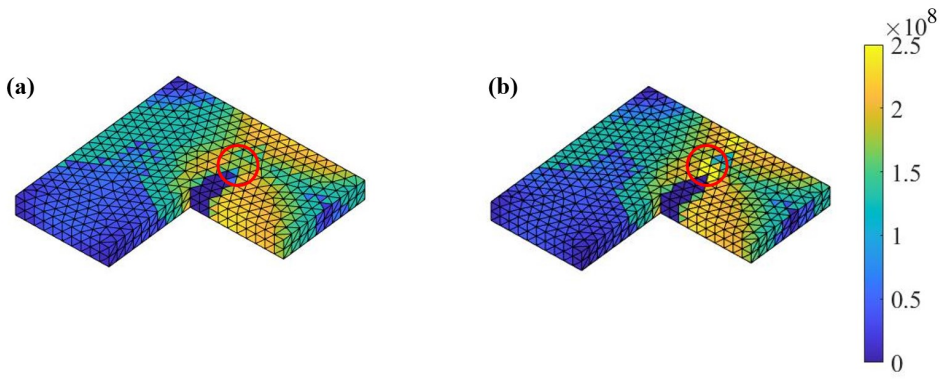
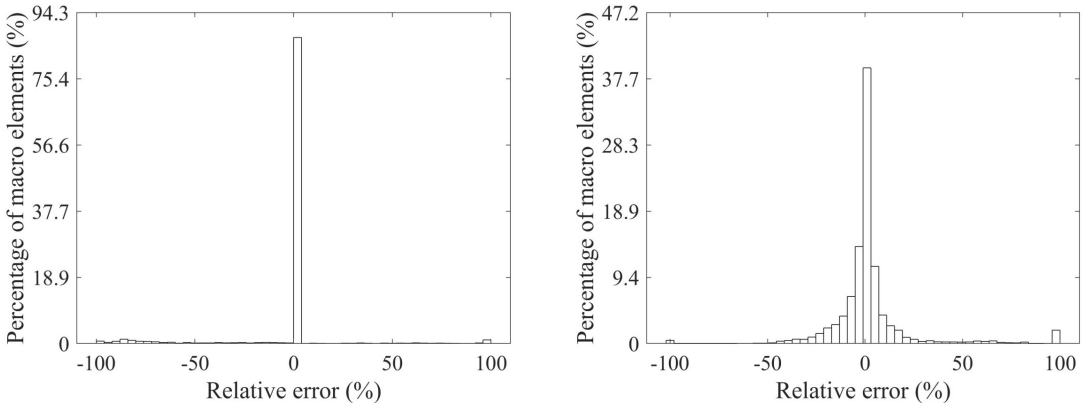


Figure 15 Comparison of Von-Mises stress distributions in softening simulation: (**a**) The distribution of benchmark Von-Mises stresses (unit: Pa); and (**d**) Von-Mises stresses by our FE-RNN model.

Table 2 Breakdown of computational costs of the L-shape bracket model: Despite considerable costs in database generation and training of the RNN, its efficiency (measured by clock time) in the multiscale simulations is about $125\times$ and $1240\times$ higher than ROM and FE^2, respectively, where the time estimation of FE^2 comes from the time comparison in Figure A.3(b).

	FE-RNN (proposed)	FE-ROM (benchmark)	FE^2 (estimated)
Database generation	239.5×60 CPU-hour	-	-
RNN training	4.3×2 GPU-hour	-	-
Multiscale simulation	0.4×4 CPU-hour	49.8×60 CPU-hour	494.0×60 CPU-hour

The geometry, dimensions, and boundary conditions of the double notched specimen is demonstrated in Figure 17(a). The left surface of the specimen is fixed and its right surface is extended by $d = 0.6$ mm. In this experiment, we model the entire specimen as a multiscale structure where each macro IP is associated with a porous RVE. For the mesh convergence study, we discretize the



(a) Relative errors of damage variables

(b) Relative errors of Von-Mises stresses

Figure 16 Histogram of relative errors of field variables: (**a**) Relative errors of the values of damage variables between benchmark and our FE-RNN model in Figure 14; and (**b**) Relative errors of the values of Von-Mises stresses in Figure 15.

macro specimen with three different mesh sizes: a coarse mesh with $7,000$ elements, a medium mesh with $18,000$ elements, and a fine mesh with $28,000$ elements.

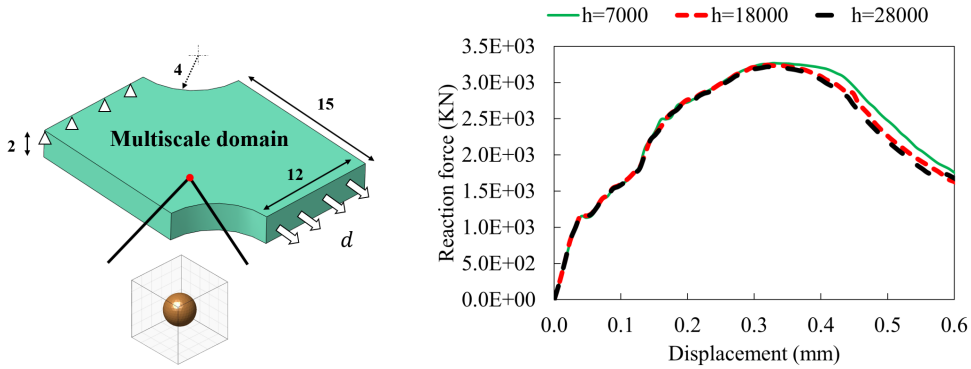
We demonstrate the reaction force-displacement curves in Figure 17(b) which indicates that the three mesh levels achieve very close elasto-plastic hardening responses and are slightly different in the softening regime. Specifically, we note that as the mesh level increases from medium to fine, the post-failure force-displacement responses tend to converge.

We also probe the convergence by inspecting the stress distributions and damage patterns, see Figure 18. In Figure 18(a) and (b), we can clearly see that at all mesh levels the damage initiates from the inner circular surfaces and propagates across the specimen as it deforms. The influence of imposing non-local function is evident: it not only successfully avoids non-physical single-element-wide damage layers, but also constrains the fracture bandwidth regardless of the mesh sizes. Additionally, as indicated in Figure 18(a) stress concentrations consistently appear at both fracture front tips and around sharp corners.

We report the simulation time of this multiscale double notched specimen in Table 3. We emphasize that due to the superior efficiency, we can apply our trained FE-RNN to any multiscale models with no extra costs of data generation and model training. Additionally, our trained FE-RNN model is memory light and can be run on a desktop with four CPU cores and 16GB RAM. Base on the time comparison in Table 2, simulation of the multiscale model with 28,000 elements requires the clock time of 1,304.8 hours (54.4 days) and 12,942.8 hours (539.3 days) by paralleling 60 CPU cores with 360 GB RAM by FE-ROM and FE^2 methods, respectively.

5 Conclusions

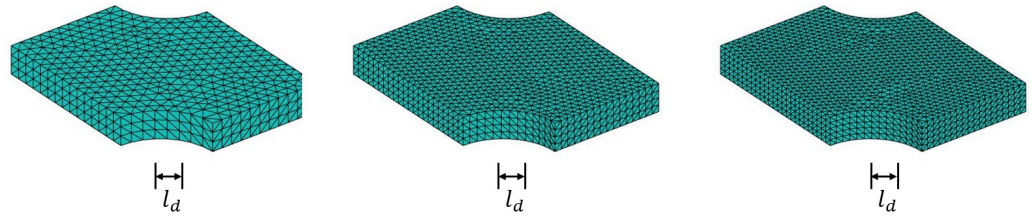
We propose a physics-constrained deep learning model that surrogates the homogenized path-dependent microstructural behaviors in 3D large-scale multiscale simulations. Our deep learning model is an RNN trained on a database that is generated by efficiently sampling deformation path spaces (spanned by strains) and obtaining the corresponding microstructural responses via a



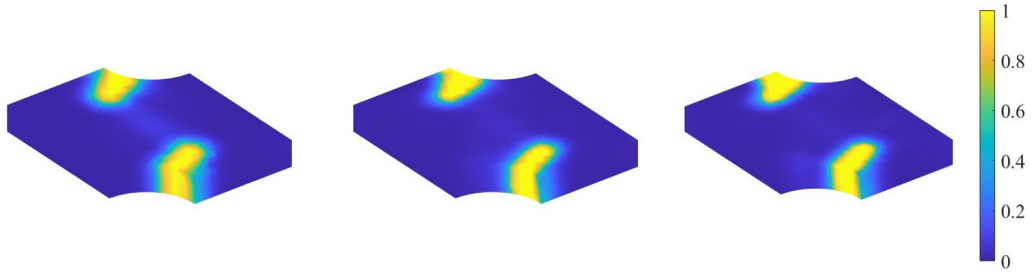
(a) Geometry, dimensions (unit: mm) and boundary conditions

(b) Reaction force and displacement

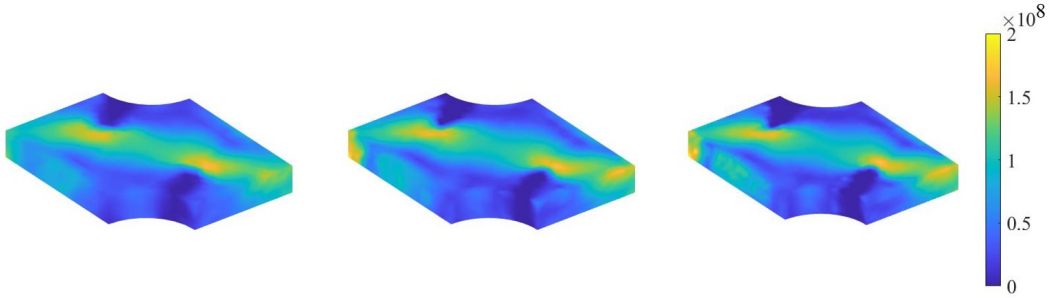
Figure 17 Multiscale model of double-notched specimen: (**a**) Every macroscale integration point is associated with a microscale porous RVE; and (**b**) Convergence study of the softening load-displacement curves with different macro discretization levels.



(a) Mesh sizes and strain localization bandwidth (l_d)



(b) Damage patterns



(c) Von-Mises stress distributions

Figure 18 Mesh convergence study of field variables by our FE-RNN: With the increase of the number of macro-elements in (**a**), both the damage patterns in (**b**) and stress distributions in (**c**) show convergence.

Table 3 Computational time of the double notched model: The multiscale simulations of the double notched specimen are performed by the proposed FE-RNN model on a modest desktop with four CPU cores where the computational costs of one-time data generation and model training are reported in Table 2.

Number of macro-elements	Multiscale simulation time
$7,000$	3.1×4 CPU-hour
$18,000$	7.5×4 CPU-hour
$28,000$	10.5×4 CPU-hour

physics-based reduced order model. To reduce the reliance on expensive data while increasing the accuracy, we leverage two constraints while building our RNN. The first constraint is based on our energy analysis on thermodynamic consistency of microstructural deformation and we implement this constraint by adding a penalty term to our RNN's loss function. The second constraint draws inspiration from the irreversible nature of damage processes and we implement it as a hard con-

straint by directly manipulating the temporal variation of the outputs within the RNN architecture. In addition, we incorporate the teacher forcing technique into our RNN model and demonstrate its impacts in both single and multiscale simulations.

We validate the accuracy of our model in both single and multiscale simulations that involve path-dependent deformations with hardening and softening. Importantly, we show that our model is accurate enough to provide reliable multiscale simulations with complex and cyclic loadings, particularly, we can reliably capture softening behaviors such that the solutions are post-failure convergent and mesh independent.

Our experiments reveal that while the costs of database generation and model training are considerable, these costs are amortized in online computations. For example, our data-driven model is about four orders of magnitude faster than classic FE^2 approach in terms of CPU hours. Such high efficiency makes our model promising for many computationally intensive tasks that would require large computational resources (multi-core CPUs and GPUs) or need long simulation times.

We plan to extend our data-driven model in a number of different directions our future works. First, minimization of inference error is critical especially for iterative solvers. While teacher-forcing shows accuracy improvement in single scale simulation, we are interested in how its performance improvement can be translated to the online iterative multiscale computations without available ground truth values (simply adding noise during training does not seem to help based on our studies). Second, we are also interested in studying the impacts of spatially varying material properties and microstructural morphology on the behaviors of macro components. However, adding such variations would dramatically increase the dimension of sampling space and therein the number of sampling points. In such scenario, to reduce sampling efforts, an adaptive sampling strategy [42] for sequence learners needs to be investigated. Lastly, we plan to make our model probabilistic for outer-loop applications such as uncertainty quantification [43] and material/structure design optimization [44–47].

Declaration of competing interest

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

Acknowledgments

The authors appreciate the supports from ACRC consortium members. The authors also thank Dr. Ling Wu and Dr. Ludovic Noels for helpful discussions and constructive suggestions. Ramin Bostanabad also acknowledges NSF funding (award numbers 2103708 and 2211908).

Appendices

A Deflated clustering analysis

Simulation of microstructural softening via classic FE^2 method involves demanding computational costs, which is prohibitive for generating big training data for machine learning models. To accelerate the database generation, we adopt our previously developed mechanistic ROM, i.e., deflated clustering analysis (DCA) [7, 8]. Its high efficiency comes from two facts: (1) the number of unknown variables in the system is dramatically reduced from a large number of finite elements

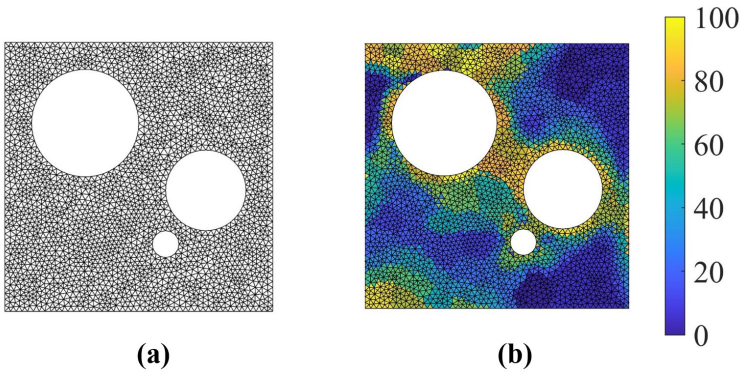


Figure A.1 Demonstration of clustering in ROM: The domain of a generic 2D RVE with $5,000$ elements in **(a)** are decomposed into 100 clusters in **(b)** where elements in the same cluster are assigned with the same color.

to a few clusters by agglomerating elements via clustering as shown in Figure A.1, and (2) the algebraic equations of the reduced system contains much fewer close-to-zero eigenvalues that results in better convergence comparing to the classic FE system.

Our DCA utilizes k-means clustering, i.e., an unsupervised machine learning technique for data interpretation and grouping, to agglomerate neighboring elements into a set of interactive irregular-shape clusters. The clustering begins with feeding the coordinates of element centroids into a feature space where randomly scattered cluster seeds serve as initial cluster means. Clusters accepts or rejects elements by iteratively minimizing the within-cluster variance until all elements are assigned to a cluster. The clustering procedure can be mathematically stated as a minimization problem as:

$$\boldsymbol{C} = \min_{\boldsymbol{C}} \sum_{I=1}^{k} \sum_{n \in C^I} \|\boldsymbol{\varphi}_n - \bar{\boldsymbol{\varphi}}_I\|^2 \tag{A-1}$$

where $\boldsymbol{C}$ represents the k clusters with $\boldsymbol{C} = \{C^1, C^2, \ldots, C^k\}$. $\boldsymbol{\varphi}_n$ and $\bar{\boldsymbol{\varphi}}_I$ indicate the coordinates of the centroid of the n^{th} element and the mean of the coordinates of the I^{th} cluster, respectively. A clustering example is illustrated in Figure A.1 where the discrete domain of a 2D generic RVE with $5,000$ elements are decomposed into 100 clusters.

We construct clustering-based reduced mesh via Delaunay triangularization by connecting cluster centroids where the topological relations between clusters are preserved from the original FE mesh. By assuming the motions of cluster centroids are directly related to clustering nodes, we can compute the nodal displacements via polynomial augmented radian point interpolation [48] as:

$$\boldsymbol{u}_c = \boldsymbol{R}\boldsymbol{a} + \boldsymbol{Z}\boldsymbol{b} \tag{A-2}$$

where $\boldsymbol{u}_c$ represent the displacements of cluster centroids. $\boldsymbol{a}$ is the coefficient vector of the radial basis function matrix $\boldsymbol{R}$, and $\boldsymbol{b}$ is the coefficient vector of the polynomial basis matrix $\boldsymbol{Z}$. Meanwhile, the radial coefficient and the polynomial basis need to satisfy the following equation for every node per cluster and every polynomial basis function to ensure solution uniqueness [48] as:

$$\boldsymbol{Z}\boldsymbol{a} = \boldsymbol{0} \tag{A-3}$$

The displacements of cluster centroids are augmented with rotational degrees of freedom to represent the six rigid body motions in a 3D deflation space [49], including three translations and three rotations. Upon the completion of non-linear analysis on the reduced mesh, the displacement solutions can be projected back to the original FE mesh by:

$$\boldsymbol{u}_i^j = \boldsymbol{W}_i^j \boldsymbol{\lambda}_j \tag{A-4}$$

where $\boldsymbol{u}_i^j$ represents the displacement vector at the i^{th} node in the j^{th} cluster. In addition, $\boldsymbol{\lambda}_j$ is the rigid body motion of the centroid of the j^{th} cluster, while the $\boldsymbol{W}_i^j$ indicates the deflation matrix for the i^{th} node in the j^{th} cluster as:

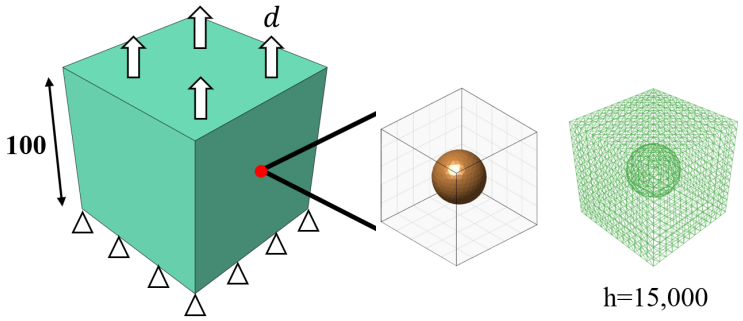
$$\boldsymbol{\lambda}_j = [u_{jx}, u_{jy}, u_{jz}, \theta_{jx}, \theta_{jy}, \theta_{jz}]^T; \quad \boldsymbol{W}_i^j = \begin{bmatrix} 1 & 0 & 0 & 0 & z_i^j & -y_i^j \\ 0 & 1 & 0 & -z_i^j & 0 & x_i^j \\ 0 & 0 & 1 & y_i^j & -x_i^j & 0 \end{bmatrix} \tag{A-5}$$

where u_{jx} and θ_{jx} are the displacement and rotation of the j^{th} cluster along x axis, and the (x_i^j, y_i^j, z_i^j) are the relative 3D coordinates of the i^{th} node with respect to the centroid of the j^{th} cluster. By assuming all elements in the same cluster share identical stress and strain fields, microstructural effective responses can be reproduced in a highly efficient manner such that the unknown variables are dramatically decreased from FE system that accounts for distinct field variables per element to the reduced system with much fewer distinct solutions per cluster.

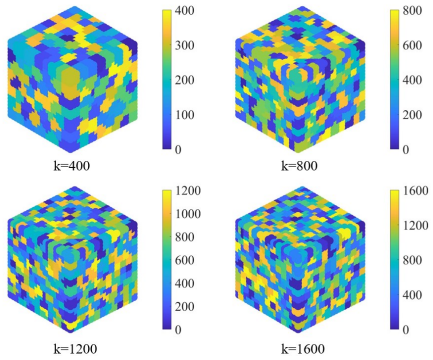
To demonstrate the efficacy of DCA-based ROM, we compare its simulation results on a 3D multiscale cube against the classic FE^2 method in Figure A.2. The macro-cube is fully constrained at its bottom surface, and it is subject to an upward extension on the top surface with $d = 7$ mm. The cube is meshed with 12 tetrahedral elements of reduced-integration (one IP at the center of each tetrahedron). We assume each macro-IP is associated with the same porous RVE containing one spherical pore in the middle as shown in Figure A.2(a). To illustrate the effects of clustering on the RVE's effective softening behaviors, we adopt four clustering levels on the same RVE mesh ($15,000$ elements) with the number of clusters (k) as 400, 800, $1,200$ and $1,600$ as in the Figure A.2(b).

We compare the reaction force-displacement curves from FE^2 and FE-ROM in Figure A.3(a). By considering the FE^2 solutions as the benchmark, we observe that: (1) the FE-ROM solutions of $k = 400$ generally overestimates the component's strength which is due to insufficient clustering in the RVE that results in artificially strong material responses as discussed in [5, 7]; and (2) With more clusters, the FE-ROM responses, especially the post-failure behaviors, become more and more closer to the benchmark. In specific, we observe that when the numbers of clusters increase to $1,200$ and $1,600$, FE-ROMs achieve sufficiently accurate results compared to FE^2 benchmark.

We further quantify the computational costs of the different solvers in Figure A.3(b). While all experiments are performed on a HPC by paralleling 60 CPU cores with 360 GB RAM, the clock time of FE^2 is the longest, accounting for 24.9 hours. The clock time of the ROM with

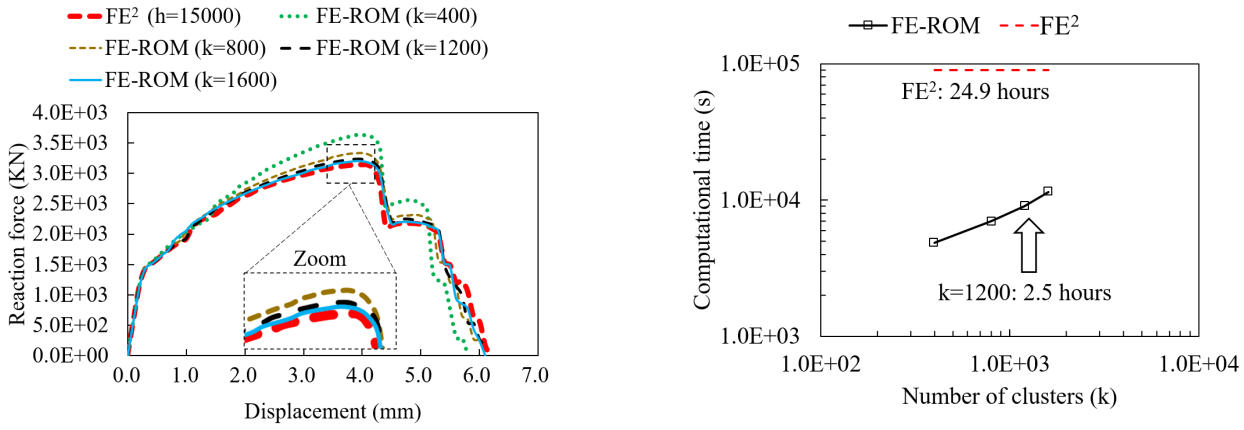


(a) Geometry, dimensions (unit: mm), and boundary conditions of the macro-model; Geometry and the finite element mesh of the porous RVE



(b) RVE clustering

Figure A.2 Multiscale cube model: (**a**) Every integration point of the macro-cube model is associated with a porous RVE; and (**b**) The RVE domain is discretized by different numbers of clusters.



(a) Reaction force and displacement

(b) Computational time

Figure A.3 Results of the multiscale cube model: (**a**) Comparison of the softening load-displacement curves between FE^2 and FE-ROM with different clusters; and (**b**) Comparison of computational time.

$1,200$ and $1,600$ clusters is about 2.5 and 3.2 hours, resulting in the acceleration factors of 9.9 and 7.8, respectively. Considering the fact that the ROM of $k = 1,200$ is about 28% faster than its counterpart of $k = 1,600$ while achieving similar accuracy, we adopt $k = 1,200$ as the clustering of choice for the generation of the RVE softening database and for the FE-ROM multiscale simulations in Section 4.

B Gated recurrent unit

To alleviate vanishing and exploding gradient issues of RNNs in processing long sequential data, long short term memory (LSTM) and gated recurrent unit (GRU) are typically used. GRU is a variant of the LSTM that, while providing similar accuracy, is more parsimonious and hence computationally more efficient. It is for this reason that we choose GRU as the memory cell in our proposed RNN architecture as in Figure 4.

To demonstrate the working mechanism of GRUs, we three interconnected cells of a GRU layer in Figure B.1. In a GRU layer, a typical cell at an arbitrary time step t generates predictions $\hat{\boldsymbol{y}}_t$ and internal memory-like hidden variables $\boldsymbol{h}_t$ after reading in the current inputs $\boldsymbol{x}_t$ and the hidden variables $\boldsymbol{h}_{t-1}$ from the previous cell. Compared to the RNN cell in Figure 3(b), the GRU cell uses reset and update gates to regulate its internal information flow. The reset gate $\boldsymbol{r}_t$ reads $\boldsymbol{x}_t$ and $\boldsymbol{h}_{t-1}$ to determine the candidate hidden state $\hat{\boldsymbol{h}}_t$ by filtering out less important information passing from the previous cell. Its operations include:

$$\boldsymbol{r}_t = \sigma\left(\boldsymbol{W}_{hr}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{xr}\boldsymbol{x}_t + \boldsymbol{b}_r\right) \tag{B-1a}$$

$$\tilde{\boldsymbol{h}}_t = \tanh\left(\boldsymbol{r}_t \odot \boldsymbol{W}_{h\tilde{h}}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{x\tilde{h}}\boldsymbol{x}_t + \boldsymbol{b}_{\tilde{h}}\right) \tag{B-1b}$$

where σ is the sigmoid activation function that returns a value in the range of $[0, 1]$, $tanh$ is the hyperbolic tangent function, and $\odot$ represents the Hadamard product. $\boldsymbol{W}_{hr}$, $\boldsymbol{W}_{xr}$, $\boldsymbol{W}_{h\tilde{h}}$, $\boldsymbol{W}_{x\tilde{h}}$ are the weight matrices associated with the hidden state, the input state, the hidden-to-candidate hidden state and the input-to-candidate hidden state, respectively. $\boldsymbol{b}_r$ and $\boldsymbol{b}_{\tilde{h}}$ are the biases applied to the sigmoid function in the reset gate and the hyperbolic tangent function, respectively.

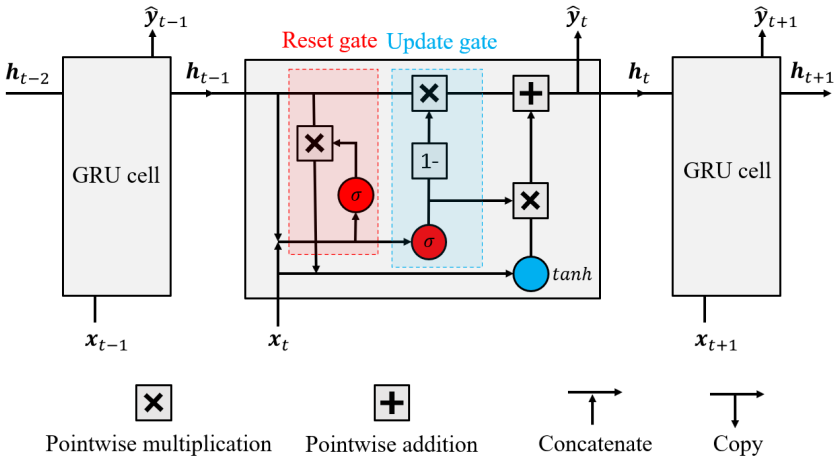


Figure B.1 Architectures of GRU layer and cells: The internal structure and mathematical operations are demonstrated in the GRU cell at time step t.

The update gate (which has its own weights and biases) similarly operates on $\boldsymbol{x}_t$ and $\boldsymbol{h}_{t-1}$: it linearly interpolates the previous hidden state $\boldsymbol{h}_{t-1}$ and the candidate hidden state $\tilde{\boldsymbol{h}}_t$ to update the memory-like hidden state $\boldsymbol{h}_t$ which is then passed to the next cell:

$$\boldsymbol{u}_t = \sigma\left(\boldsymbol{W}_{hu}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{xu}\boldsymbol{x}_t + \boldsymbol{b}_u\right) \tag{B-2a}$$

$$\boldsymbol{h}_t = \boldsymbol{u}_t \odot \boldsymbol{h}_{t-1} + (1 - \boldsymbol{u}_t) \odot \tilde{\boldsymbol{h}}_t + \boldsymbol{b}_h \tag{B-2b}$$

where $\boldsymbol{W}_{hu}$ and $\boldsymbol{W}_{xu}$ are the weights applied onto the hidden state and input state in the update gate. $\boldsymbol{b}_u$ and $\boldsymbol{b}_h$ are the two biases associated to the sigmoid function and the generation of current hidden state. The che cell output at the current time step $\hat{\boldsymbol{y}}_t$ is then obtained by linearly transforming the hidden state:

$$\hat{\boldsymbol{y}}_t = \boldsymbol{W}_{hy}\boldsymbol{h}_t + \boldsymbol{b}_y \tag{B-3}$$

where $\boldsymbol{W}_{hy}$ and $\boldsymbol{b}_y$ are the weights and biases associated with the current output state $\hat{\boldsymbol{y}}_t$. We note that all the weights and biases of the GRU networks are iteratively updated by BPTT during training.

References

[1] Frédéric Feyel and Jean-Louis Chaboche. “FE2 multiscale approach for modelling the elastoviscoplastic behaviour of long fibre SiC/Ti composite materials”. In: *Computer methods in applied mechanics and engineering* 183.3-4 (2000), pp. 309–330.

[2] Pascale Kanouté, DP Boso, Jean-Louis Chaboche, and BA1170 Schrefler. “Multiscale methods for composites: a review”. In: *Archives of Computational Methods in Engineering* 16.1 (2009), pp. 31–75.

[3] George J Dvorak. “Transformation field analysis of inelastic composite materials”. In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 437.1900 (1992), pp. 311–327.

[4] Sophie Roussette, Jean-Claude Michel, and Pierre Suquet. “Nonuniform transformation field analysis of elastic–viscoplastic composites”. In: *Composites Science and Technology* 69.1 (2009), pp. 22–27.

[5] Zeliang Liu, MA Bessa, and Wing Kam Liu. “Self-consistent clustering analysis: an efficient multi-scale scheme for inelastic heterogeneous materials”. In: *Computer Methods in Applied Mechanics and Engineering* 306 (2016), pp. 319–341.

[6] Shaoqiang Tang, Lei Zhang, and Wing Kam Liu. “From virtual clustering analysis to self-consistent clustering analysis: a mathematical study”. In: *Computational Mechanics* 62.6 (2018), pp. 1443–1460.

[7] Shiguang Deng, Carl Soderhjelm, Diran Apelian, and Ramin Bostanabad. “Reduced-order multiscale modeling of plastic deformations in 3D alloys with spatially varying porosity by deflated clustering analysis”. In: *Computational Mechanics* 70.3 (2022), pp. 517–548.

[8] Shiguang Deng, Diran Apelian, and Ramin Bostanabad. “Adaptive spatiotemporal dimension reduction in concurrent multiscale damage analysis”. In: *Computational Mechanics* (2023), pp. 1–33.

[9] R. Planas, N. Oune, and R. Bostanabad. “Evolutionary Gaussian Processes”. In: *Journal of Mechanical Design* 143.11 (2021), p. 111703. ISSN: 1050-0472. DOI: Artn11170310.1115/1.4050746. URL: %3CGo%20to%20ISI%3E://WOS:000702466200013.

[10] N. Oune and R. Bostanabad. “Latent map Gaussian processes for mixed variable metamodeling”. In: *Computer Methods in Applied Mechanics and Engineering* 387 (2021), p. 114128. ISSN: 0045-7825. DOI: ARTN11412810.1016/j.cma.2021.114128. URL: %3CGo%20to%20ISI%3E://WOS:000708647400003.

[11] W. Chen, A. Iyer, and R. Bostanabad. “Data Centric Design: A New Approach to Design of Microstructural Material Systems”. In: *Engineering* 10 (2022), pp. 89–98. ISSN: 2095-8099. DOI: 10.1016/j.eng.2021.05.022. URL: %3CGo%20to%20ISI%3E://WOS:000807736400013.

[12] Loujaine Mehrez, Jacob Fish, Venkat Aitharaju, Will R. Rodgers, and Roger Ghanem. “A PCE-based multiscale framework for the characterization of uncertainties in complex systems”. In: *Computational Mechanics* 61.1-2 (2017), pp. 219–236. ISSN: 0178-7675 1432-0924. DOI: 10.1007/s00466-017-1502-4. URL: https://doi.org/10.1007/s00466-017-1502-4.

[13] Tomonari Furukawa and Genki Yagawa. “Implicit constitutive modelling for viscoplasticity using neural networks”. In: *International Journal for Numerical Methods in Engineering* 43.2 (1998), pp. 195–219.

[14] Tomonari Furukawa and Mark Hoffman. “Accurate cyclic plastic analysis using a neural network material model”. In: *Engineering Analysis with Boundary Elements* 28.3 (2004), pp. 195–204.

[15] Mauricio Fernández, Shahed Rezaei, Jaber Rezaei Mianroodi, Felix Fritzen, and Stefanie Reese. “Application of artificial neural networks for the prediction of interface mechanics: a study on grain boundary constitutive behavior”. In: *Advanced Modeling and Simulation in Engineering Sciences* 7.1 (2020), pp. 1–27.

[16] Xiaoxin Lu, Julien Yvonnet, Fabrice Detrez, and Jinbo Bai. “Multiscale modeling of nonlinear electric conductivity in graphene-reinforced nanocomposites taking into account tunnelling effect”. In: *Journal of Computational Physics* 337 (2017), pp. 116–131.

[17] Jaber Rezaei Mianroodi, Nima H Siboni, and Dierk Raabe. “Teaching solid mechanics to artificial intelligence—a fast solver for heterogeneous materials”. In: *Npj Computational Materials* 7.1 (2021), pp. 1–10.

[18] Ehsan Haghighat, Maziar Raissi, Adrian Moure, Hector Gomez, and Ruben Juanes. “A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics”. In: *Computer Methods in Applied Mechanics and Engineering* 379 (2021), p. 113741.

[19] Iman Peivaste, Nima H Siboni, Ghasem Alahyarizadeh, Reza Ghaderi, Bob Svendsen, Dierk Raabe, and Jaber Rezaei Mianroodi. “Machine-learning-based surrogate modeling of microstructure evolution using phase-field”. In: *Computational Materials Science* 214 (2022), p. 111750.

[20] M Mozaffar, R Bostanabad, W Chen, K Ehmann, Jian Cao, and MA Bessa. “Deep learning predicts path-dependent plasticity”. In: *Proceedings of the National Academy of Sciences* 116.52 (2019), pp. 26414–26420.

[21] Kun Wang and WaiChing Sun. “A multiscale multi-permeability poroplasticity model linked by recursive homogenizations and deep learning”. In: *Computer Methods in Applied Mechanics and Engineering* 334 (2018), pp. 337–380.

[22] Ling Wu, Nanda Gopala Kilingar, Ludovic Noels, et al. “A recurrent neural network-accelerated multi-scale model for elasto-plastic heterogeneous materials subjected to random cyclic and non-proportional loading paths”. In: *Computer Methods in Applied Mechanics and Engineering* 369 (2020), p. 113234.

[23] F Ghavamian and A Simone. “Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network”. In: *Computer Methods in Applied Mechanics and Engineering* 357 (2019), p. 112594.

[24] Hernan J Logarzo, German Capuano, and Julian J Rimoli. “Smart constitutive laws: Inelastic homogenization through machine learning”. In: *Computer methods in applied mechanics and engineering* 373 (2021), p. 113482.

[25] Fermin Otero, Sergio Oller, and Xavier Martinez. “Multiscale computational homogenization: review and proposal of a new enhanced-first-order method”. In: *Archives of Computational Methods in Engineering* 25.2 (2018), pp. 479–505.

[26] Zdenek P Bazant. “Can multiscale-multiphysics methods predict softening damage and structural failure?” In: *International Journal for Multiscale Computational Engineering* 8.1 (2010).

[27] Zeliang Liu, Mark Fleming, and Wing Kam Liu. “Microstructural material database for self-consistent clustering analysis of elastoplastic strain softening materials”. In: *Computer Methods in Applied Mechanics and Engineering* 330 (2018), pp. 547–577.

[28] Michael Smith. “ABAQUS Standard User’s Manual”. In: *Dassault Systèmes Simulia Corp* (2009), Version 6.9.

[29] Javier Oliver, Alfredo Edmundo Huespe, and JC Cante. “An implicit/explicit integration scheme to increase computability of non-linear material and contact/friction problems”. In: *Computer Methods in Applied Mechanics and Engineering* 197.21-24 (2008), pp. 1865–1889.

[30] Miroslav Silhavy. *The mechanics and thermodynamics of continuous media*. Springer Science & Business Media, 2013.

[31] Han Yang, Sumeet Kumar Sinha, Yuan Feng, David B McCallen, and Boris Jeremić. “Energy dissipation analysis of elastic–plastic materials”. In: *Computer Methods in Applied Mechanics and Engineering* 331 (2018), pp. 309–326.

[32] Heidi P Feigenbaum and Yannis F Dafalias. “Directional distortional hardening in metal plasticity within thermodynamics”. In: *International Journal of Solids and Structures* 44.22-23 (2007), pp. 7526–7542.

[33] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural networks* 2.5 (1989), pp. 359–366.

[34] Zachary C Lipton, John Berkowitz, and Charles Elkan. “A critical review of recurrent neural networks for sequence learning”. In: *arXiv preprint arXiv:1506.00019* (2015).

[35] Boris Hanin. “Which neural net architectures give rise to exploding and vanishing gradients?” In: *Advances in neural information processing systems* 31 (2018).

[36] Ralf C Staudemeyer and Eric Rothstein Morris. “Understanding LSTM–a tutorial into long short-term memory recurrent neural networks”. In: *arXiv preprint arXiv:1909.09586* (2019).

[37] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. “Visualizing and understanding recurrent networks”. In: *arXiv preprint arXiv:1506.02078* (2015).

[38] Zixue Xiang, Wei Peng, Xu Liu, and Wen Yao. “Self-adaptive loss balanced Physics-informed neural networks”. In: *Neurocomputing* 496 (2022), pp. 11–34.

[39] Pablo Márquez-Neila, Mathieu Salzmann, and Pascal Fua. “Imposing hard constraints on deep networks: Promises and limitations”. In: *arXiv preprint arXiv:1706.02025* (2017).

[40] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[41] Shiguang Deng, Carlos Mora, Diran Apelian, and Ramin Bostanabad. “Data-Driven Calibration of Multifidelity Multiscale Fracture Models Via Latent Map Gaussian Process”. In: *Journal of Mechanical Design* 145.1 (2022), p. 011705.

[42] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. “Scheduled sampling for sequence prediction with recurrent neural networks”. In: *Advances in neural information processing systems* 28 (2015).

[43] Ramin Bostanabad, Biao Liang, Jiaying Gao, Wing Kam Liu, Jian Cao, Danielle Zeng, Xuming Su, Hongyi Xu, Yang Li, and Wei Chen. “Uncertainty quantification in multiscale simulation of woven fiber composites”. In: *Computer Methods in Applied Mechanics and Engineering* 338 (2018), pp. 506–532.

[44] Mikhail Osanov and James K Guest. “Topology optimization for architected materials design”. In: *Annual Review of Materials Research* 46 (2016), pp. 211–233.

[45] Zheng-Dong Ma, Noboru Kikuchi, Christophe Pierre, and Basavaraju Raju. “Multidomain topology optimization for structural and material designs”. In: (2006).

[46] Shiguang Deng and Krishnan Suresh. “Multi-constrained 3D topology optimization via augmented topological level-set”. In: *Computers & Structures* 170 (2016), pp. 1–12.

[47] Shiguang Deng and Krishnan Suresh. “Multi-constrained topology optimization via the topological sensitivity”. In: *Structural and Multidisciplinary Optimization* 51.5 (2015), pp. 987–1001.

[48] Gui-Rong Liu. *Meshfree methods: moving beyond the finite element method*. CRC press, 2009.

[49] TB Jönsthövel, MB Van Gijzen, C Vuik, C Kasbergen, and A Scarpas. “Preconditioned conjugate gradient method enhanced by deflation of rigid body modes applied to composite materials”. In: *Computer Modeling in Engineering and Sciences (CMES)* 47.2 (2009), p. 97.