Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 377 (2021) 113695
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Sobolev training of thermodynamic-informed neural networks for interpretable elasto-plasticity models with level set hardening

Nikolaos N. Vlassis, WaiChing Sun*

Department of Civil Engineering and Engineering Mechanics, Columbia University, 614 SW Mudd, Mail Code: 4709, New York, NY 10027, United States of America

Received 15 October 2020; received in revised form 22 December 2020; accepted 22 January 2021
Available online xxxx

Abstract

We introduce a deep learning framework designed to train smoothed elastoplasticity models with interpretable components, such as the stored elastic energy function, yield surface, and plastic flow that evolve based on a set of deep neural network predictions. By recasting the yield function as an evolving level set, we introduce a deep learning approach to deduce the solutions of the Hamilton–Jacobi equation that governs the hardening/softening mechanism. This machine learning hardening law may recover any classical hand-crafted hardening rules and discover new mechanisms that are either unbeknownst or difficult to express with mathematical expressions. Leveraging Sobolev training to gain control over the derivatives of the learned functions, the resultant machine learning elastoplasticity models are thermodynamically consistent, interpretable, while exhibiting excellent learning capacity. Using a 3D FFT solver to create a polycrystal database, numerical experiments are conducted and the implementations of each component of the models are individually verified. Our numerical experiments reveal that this new approach provides more robust and accurate forward predictions of cyclic stress paths than those obtained from black-box deep neural network models such as the recurrent neural network, the 1D convolutional neural network, and the multi-step feed-forward model.

Keywords: Sobolev training; Multiscale; Polycrystals; Isotropic yield function; Recurrent neural network; Physics-informed constraints

1. Introduction

Plastic deformation of materials is a history-dependent process manifested by irreversible and permanent changes of microstructures, such as dislocation, pore collapses, growth of defects, and phase transition. Macroscopic constitutive models designed to capture history-dependent constitutive responses can be categorized into multiple families, such as hypoplasticity, elastoplasticity, and generalized plasticity [1–4]. For example, hypoplasticity models often do not distinguish the reversible and the irreversible strain [5–7]. Unlike the classical elastoplasticity models where the plastic flow is normal to the stress gradient of the plastic potential and the evolution of it is governed by a set of hardening rules [8–11], hypoplasticity models do not employ a yield function to characterize the initial yielding. Instead, the relationship between the strain rate and the stress rate is captured by a set of evolution laws

* Corresponding author.
E-mail address: wsun@columbia.edu (W. Sun).

https://doi.org/10.1016/j.cma.2021.113695

originated from a combination of phenomenological observations and physics constraints. Interestingly, the early design of neural network models such as Ghaboussi et al. [12], Furukawa and Yagawa [13], Pernot and Lamarque [14] and Lefik et al. [15], would often adopt this approach with a purely supervised learning strategy to adjust the weights of neurons to minimize the errors. Using the strain from current and previous time steps to estimate the current stress, these models would essentially predict the stress rate without utilizing a yield function and, hence, can be viewed as hypoplasticity models with machine learning derived evolution laws. The major issue that limits the adaptations of neural network constitutive models is the lack of interpretability and the vulnerability to over-fitting. While there are existing regularization techniques such as dropout layers [16], cross-validation [17,18], and/or increasing the size of the database that could be helpful, it remains difficult to assess their credibility without the interpretability of the underlying laws deduced from the neural network. Another approach could involve symbolic regression through reinforcement learning [3] or genetic algorithms [19] that may lead to explicitly written evolution laws, however, the fitness of these equations is often at the expense of readability.

Another common approach to predict plastic deformation is the classical elasto-plasticity model where an elasticity model is coupled with a yield function that evolves with a set of internal variables that represents the history of the materials. Within the framework of the classical elasto-plasticity model — where constitutive models are driven by the evolution of yield surface and the underlying elastic model, there has been a significant number of works dedicated to refining the initial shapes and forms of the yield functions in the stress space (e.g. [20–22]) and the corresponding hardening laws that govern the evolution of these yield functions with the plastic strain (e.g. [23–28]). Generalized plasticity [1–3] bypasses the usage of both the stress gradient of the plastic potential or the yield surface to predict plastic flow direction. Instead, an additional phenomenological relation is deduced from experiments to predict the plastic flow direction as a function of internal variables. In both cases, there are several key upshots brought by the existence of the yield function. For instance, the existence of a yield function facilitates the geometric interpretation of plasticity and, therefore, enables us to connect mechanics concepts, such as the thermodynamic law, with geometric concepts, such as convexity in the principal stress space [18,29,30]. Furthermore, the existence of a distinctive elastic region in the stress or strain space also allows introducing a multi-step transfer learning strategy. In this case, the machine learning of the elastic responses can be viewed as a pre-training step for the plasticity machine learning where predicted elastic responses can be used to determine the underlying split of the elastic and the plastic strain upon the initial yielding and, hence, allows one to deduce more accurate hardening and plastic flow rules.

1.1. Why Sobolev training for plasticity

Recently, there have been attempts to rectify the limitations of machine learning models that do not distinguish or partition the elastic and the plastic strain. Xu et al. [31], for instance, introduce a smooth transition function to create a finite transition zone between the elastic and plastic ranges for an incremental constitutive law generated from supervised learning.

Previous works on machine learning plasticity are often black-box models where strain history is used as input to predict the stress via a deep neural net trained with a pure supervised learning strategy [32–34]. Since the rationale behind the predictions is stored in the weights of the neurons, it is difficult to interpret. To circumvent this limited interpretability, previous work such as Mozaffar et al. [35] and later [36] introduce machine learning techniques to deduce the yield function and subsequently deduce the optimal linear or distortion hardening mechanism that minimizes the L_2 norm of the yield function discrepancy. Wang et al. [3], on the other hand, views the possibility of building different plasticity models as a directed multi-graph and introduces a reinforcement learning algorithm to deduce the optimal configuration of plasticity models among all the available options (e.g. isotropic, kinematic, rotation, and frictional hardening, isotropic/anisotropic elasticity) to generate fully interpretable plasticity models. Nevertheless, these previous approaches are not capable of deducing new hardening/softening mechanisms previously unbeknownst to modelers. This research has three goals: (1) creating interpretable machine learning elastoplasticity models, (2) formulating a new learning strategy that enables the discovery of **new** softening/hardening mechanisms, and (3) leveraging the interpretability of the models to enforce thermodynamic laws to make predictions compatible to universal principles of mechanics. Sobolev training plays an important role for us to achieve these goals by ensuring the regularity of the energy functionals and level set functions. By adopting the Lode's coordinate system to simplify the parametrization, we introduce a simple training

program that can generate accurate and robust stress predictions but also yield the elastic energy and an elasto-plastic tangent operator that is sufficiently smooth for numerical predictions — one of the technical barriers that prevented the adoption of neural network models since their inception in the 90s [37].

1.2. Organization of the content and notations

The organization of the rest of the paper is as follows. We first provide a detailed account of the different designs of Sobolev higher-order training introduced to generate the elastic stored energy functional, yield function, flow rules, and hardening models in their corresponding parametric space. The setup of control experiments with other common alternative black-box models is then described. We then demonstrate how to leverage this new design of an interpretable machine learning framework to analyze the thermodynamic behavior of the machine learning derived constitutive laws while illustrating the geometrical interpretation of the proposed modeling framework. A brief highlight for the adopted return mapping algorithm implementation that leverages automatic differentiation is provided in Section 3, followed by the numerical experiments and the conclusions that outline this work's major findings.

As for notations and symbols in this current work, bold-faced letters denote tensors (including vectors which are rank-one tensors); the symbol '$\cdot$' denotes a single contraction of adjacent indices of two tensors (e.g. $\boldsymbol{a} \cdot \boldsymbol{b} = a_i b_i$ or $\boldsymbol{c} \cdot \boldsymbol{d} = c_{ij} d_{jk}$); the symbol ':' denotes a double contraction of adjacent indices of a tensor of rank two or higher (e.g. $\boldsymbol{C} : \boldsymbol{\epsilon}^{\boldsymbol{e}} = C_{ijkl} \epsilon^e_{kl}$); the symbol '$\otimes$' denotes a juxtaposition of two vectors (e.g. $\boldsymbol{a} \otimes \boldsymbol{b} = a_i b_j$) or two symmetric second-order tensors (e.g. $(\boldsymbol{\alpha} \otimes \boldsymbol{\beta})_{ijkl} = \alpha_{ij} \beta_{kl}$). Moreover, $(\boldsymbol{\alpha} \oplus \boldsymbol{\beta})_{ijkl} = \alpha_{jl} \beta_{ik}$ and $(\boldsymbol{\alpha} \ominus \boldsymbol{\beta})_{ijkl} = \alpha_{il} \beta_{jk}$. We also define identity tensors $(\boldsymbol{I})_{ij} = \delta_{ij}$, $(\boldsymbol{I}^4)_{ijkl} = \delta_{ik}\delta_{jl}$, and $(\boldsymbol{I}^4_{\text{sym}})_{ijkl} = \frac{1}{2}(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{kj})$, where δ_{ij} is the Kronecker delta. As for sign conventions, unless specified otherwise, we consider the direction of the tensile stress and dilative pressure as positive. Unless otherwise specified, the strain measure listed in this paper is not expressed in percentage.

2. Framework for Sobolev training of elastoplasticity models

This section presents the framework to train the multiple deep neural networks that predict the elastic stored energy functional, the yield function, and the plastic flow that will constitute the machine learning elastoplasticity model. We first discuss the neural network architecture designed to enable the learned functions with the necessary degree of continuity for functional predictions (Section 2.1). We then introduce a number of loss functions designed to optimize hyperelastic responses of materials with different material symmetries (Section 2.2). Finally, we elaborate on the theoretical basis that enables us to treat yield function as a signed distance function level set, formulate a supervised learning task that deduces the learned evolving yield function against a monotonically increasing accumulated plastic strain, and discuss the simplified implementation allowed by the material symmetry. The method that leverages an interpretable design to enforce and validate the thermodynamic constraints and the generation of the auxiliary data points of the signed distance function level set to enhance the robustness of the learned yield functions are also discussed (Section 2.3).

2.1. Neural network design for Sobolev training of a smooth scalar functional with physical constraints

Here we will provide a brief account of the design of the neural network suitable for Sobolev training of a scalar functional that requires a specific minimum degree of continuity. The design presented here will facilitate the specific supervised learning tasks formulated for an elastic energy functional, a yield function, and a plastic flow, which are documented in Sections 2.2 and 2.3. Our goal is to obtain a learned function belonging to a Sobolev space, a space of functions possessing sufficient derivatives for our modeling purposes. To facilitate the Sobolev training, we must ensure that (1) the space of the learned function is spanned by the basis functions that possess the sufficient degree of continuity and (2) the loss function in the supervised learning is associated with the norm equipped for the corresponding Sobolev space. To meet the first criterion, one must employ activation functions with sufficient differentiability. In principle, this can be easily achieved by picking activation functions of a sufficient degree of continuity (e.g. sigmoid and hyperbolic tangent activation functions). However, stacked neuron layers with these activation functions are often suffering from the vanishing and exploding gradient problems and, hence, not suitable for our purpose [38,39]. Meanwhile, the Rectified Linear Unit (ReLU) activation function is

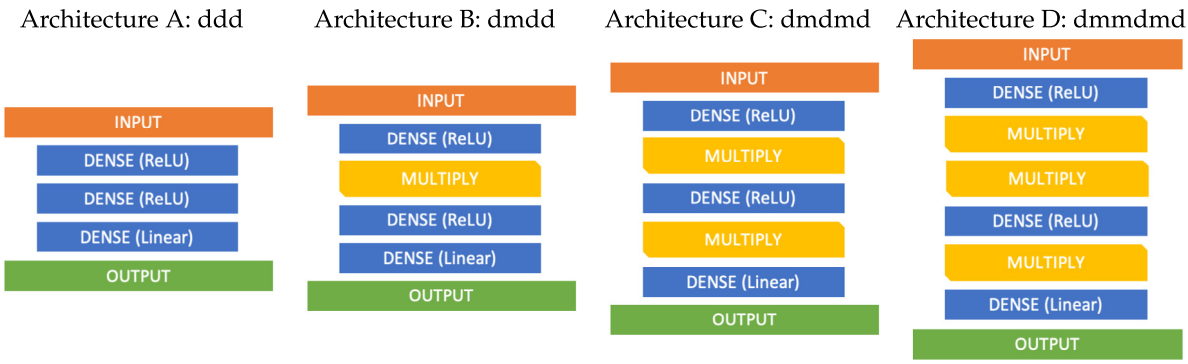


Fig. 1. Four neural network architectures (A–D) with different combinations of dense and multiply layers — the number of Multiply layers increases from left to right. The letters d and m represent the Dense and Multiply layers respectively that form an architecture (e.g. dmdd represents the stacked layer structure consisting of Dense → Multiply → Dense → Dense layers).

usually deployed instead as the default choice for multilayer perceptron and convolutional neural networks due to the generally good performance and faster learning capacity. To attain the required degree of continuity while circumventing the vanishing gradient problem, we introduce a simple technique which we refer to as Multiply layers. A Multiply layer receives the output vector $\boldsymbol{h}^{n-1}$ of the preceding layer as input and simply outputs $\boldsymbol{h}^{n}$, such that,

$$\boldsymbol{h}^{n} = \text{Multiply}(\boldsymbol{h}^{n-1}) = \boldsymbol{h}^{n-1} \circ \boldsymbol{h}^{n-1}, \tag{1}$$

where $\circ$ is the element-wise product of two vectors. These layers are placed in between two hidden dense layers of the network to modify the output of the preceding layer. This technique enables us to create neural networks that produce a learned function of an arbitrary degree of continuity without introducing any additional weights or handcrafting any custom activation functions.

The performance of these different neural networks that complete the higher-order Sobolev training is demonstrated in the numerical experiments showcased in Section 5.1. Several variations of the standard two-layer architecture we experimented with are shown in Fig. 1.

Remark 1. The placement and the number of intermediate Multiply layers are hyper-parameters that can be fine-tuned along with the rest of the hyperparameters of the neural network (e.g. dropout rate, number of neurons per layer, number of layers). The tuning of these hyperparameters can be performed manually or through automatic hyperparameter tuning algorithms (cf. [40,41]).

2.2. Sobolev training of a hyperelastic energy functional

The first component of the elastoplastic framework we train is the elastic stored energy ψ^{e}. The elastic energy functional is not only useful for predicting the stress from the elastic strain but can also be used to re-interpret the experimental data to identify the accumulated plastic strain and the plastic flow directions that are crucial for the training of the yield function and the plastic flow. In the infinitesimal strain regime, the hyperelastic energy functional $\psi^{\mathrm{e}}(\boldsymbol{\epsilon}^{\mathrm{e}}) \in \mathbb{R}^{+}$ can be defined as a non-negative valued function of elastic infinitesimal strain of which the first derivative is the Cauchy stress tensor $\boldsymbol{\sigma} \in \mathbb{S}$ and the Hessian is the tangential elasticity tensor $\boldsymbol{c}^{e} \in \mathbb{M}$:

$$\boldsymbol{\sigma} = \frac{\partial \psi^{\mathrm{e}}\left(\boldsymbol{\epsilon}^{\mathrm{e}}\right)}{\partial \boldsymbol{\epsilon}^{\mathrm{e}}}\,, \quad \boldsymbol{c}^{\mathrm{e}} = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\epsilon}^{\mathrm{e}}} = \frac{\partial^{2} \psi^{\mathrm{e}}\left(\boldsymbol{\epsilon}^{\mathrm{e}}\right)}{\partial \boldsymbol{\epsilon}^{\mathrm{e}} \otimes \partial \boldsymbol{\epsilon}^{\mathrm{e}}}\,, \tag{2}$$

where $\mathbb{S}$ is the space of the second-order symmetric tensors and $\mathbb{M}$ is the space of the fourth-order tensors that possess major and minor symmetries [17]. The true hyperelastic energy functional ψ^{e} of the material is approximated by the neural network learned function $\widehat{\psi}^{\mathrm{e}}(\boldsymbol{\epsilon}^{\mathrm{e}}|\boldsymbol{W}, \boldsymbol{b})$ with the elastic strain tensor $\boldsymbol{\epsilon}^{\mathrm{e}}$ as the input, parametrized by weights $\boldsymbol{W}$ and biases $\boldsymbol{b}$ obtained from a supervised learning procedure.

To guarantee the accuracy and smoothness of the stress and tangent operators stemmed from the learned energy functionals, we adopt a Sobolev training framework, introduced by Czarnecki et al. [42], and extend it to higher-order derivative constraints. By leveraging the differentiability achieved by the Multiply layers and adopting an H^2 norm as the training objective, we introduce an alternative that renders the neural network model applicable for implicit solvers while eliminating the potential spurious oscillations of the tangent operators. The new training objective for the hyperelastic energy functional approximator $\widehat{\psi}^{\mathrm{e}}$ includes constraints for the predicted energy, stress, and stiffness values. This training objective, modeled after an H^2 norm, for the training samples $i \in [1, \dots, N]$ would have the following form:

$$W', \boldsymbol{b}' = \underset{\boldsymbol{W},\boldsymbol{b}}{\operatorname{argmin}} \left(\frac{1}{N} \sum_{i=1}^{N} \left(\gamma_1 \left\| \psi_i^{\mathrm{e}} - \widehat{\psi}_i^{\mathrm{e}} \right\|_2^2 + \gamma_2 \left\| \frac{\partial \psi_i^{\mathrm{e}}}{\partial \boldsymbol{\epsilon}_i^{\mathrm{e}}} - \frac{\partial \widehat{\psi}_i^{\mathrm{e}}}{\partial \boldsymbol{\epsilon}_i^{\mathrm{e}}} \right\|_2^2 + \gamma_3 \left\| \frac{\partial^2 \psi_i^{\mathrm{e}}}{\partial \boldsymbol{\epsilon}_i^{\mathrm{e}} \otimes \partial \boldsymbol{\epsilon}_i^{\mathrm{e}}} - \frac{\partial^2 \widehat{\psi}_i^{\mathrm{e}}}{\partial \boldsymbol{\epsilon}_i^{\mathrm{e}} \otimes \partial \boldsymbol{\epsilon}_i^{\mathrm{e}}} \right\|_2^2 \right) \right). \tag{3}$$

In the above formulation, the conventional L_2 norm-based training objective can be obtained by setting parameters $\gamma_2 = \gamma_3 = 0$. An H_1 norm-based training objective can be recovered from Eq. (3) by setting $\gamma_3 = 0$. An H_1 norm loss function for a hyperelastic energy functional can be seen in Vlassis et al. [18].

In this work, we generate elasto-plastic models that are practical for implicit solvers. This, however, was considered a difficult task in the earlier attempts of using a neural network as a replacement for constitutive laws (cf. Hashash et al. [37]) where the proposed solution would be to either bypass the calculation of a tangent with an explicit time integrator or to introduce finite differences on the stress predictions.

2.2.1. Simplified training for isotropic elasticity

In this section, our primary focus is on deriving small strain isotropic hyperelasticity models which can written in a spectral form with the principal strains as the only inputs. As such, the H_2 training objective of Eq. (3) for the training samples $i \in [1, \dots, N]$ can be rewritten in terms of principal values as:

$$\begin{aligned} W', \boldsymbol{b}' = \underset{\boldsymbol{W},\boldsymbol{b}}{\operatorname{argmin}} \Bigg(\frac{1}{N} \sum_{i=1}^{N} \Bigg(\gamma_1 \left\| \psi_i^{\mathrm{e}} - \widehat{\psi}_i^{\mathrm{e}} \right\|_2^2 + \sum_{A=1}^{3} \gamma_2 \left\| \frac{\partial \psi_i^{\mathrm{e}}}{\partial \epsilon_{A,i}^{\mathrm{e}}} - \frac{\partial \widehat{\psi}_i^{\mathrm{e}}}{\partial \epsilon_{A,i}^{\mathrm{e}}} \right\|_2^2 \\ + \sum_{A=1}^{3} \sum_{B=1}^{3} \gamma_3 \left\| \frac{\partial^2 \psi_i^{\mathrm{e}}}{\partial \epsilon_{A,i}^{\mathrm{e}} \partial \epsilon_{B,i}^{\mathrm{e}}} - \frac{\partial^2 \widehat{\psi}_i^{\mathrm{e}}}{\partial \epsilon_{A,i}^{\mathrm{e}} \partial \epsilon_{B,i}^{\mathrm{e}}} \right\|_2^2 \Bigg) \Bigg), \end{aligned} \tag{4}$$

where ϵ_A^{e} for $A = 1, 2, 3$ are the principal values of the elastic strain tensor $\boldsymbol{\epsilon}^{\mathrm{e}}$. The approximated energy functional for this training objective is a function of the three input principal strains and not of a full second-order tensor of 6 input components (reduced from 9 by assuming symmetry). This treatment reduces the dimension of the parametric space and facilitates learning by minimizing complexity.

2.2.2. Simplified training for two-invariant elasticity

The training objective can be further simplified to two input variables by adopting an invariant space, commonly used in geotechnical studies when the intermediate principal stress does not exhibit a dominating effect on the elastic response or when the intermediate principal stress is not measured at all due to limitations of the experiment apparatus [43,44]. In this case, a small-strain isotropic hyperelastic law can equivalently be described with two strain invariants (volumetric strain ϵ_v^{e}, deviatoric strain ϵ_s^{e}). The strain invariants are defined as:

$$\epsilon_v^{\mathrm{e}} = \operatorname{tr}\left(\boldsymbol{\epsilon}^{\mathrm{e}}\right), \quad \epsilon_s^{\mathrm{e}} = \sqrt{\frac{2}{3}} \left\| \boldsymbol{e}^{\mathrm{e}} \right\|, \quad \boldsymbol{e}^{\mathrm{e}} = \boldsymbol{\epsilon}^{\mathrm{e}} - \frac{1}{3} \epsilon_v^{\mathrm{e}} \mathbf{1}, \tag{5}$$

where $\boldsymbol{\epsilon}^{\mathrm{e}}$ is the small strain tensor and $\boldsymbol{e}^{\mathrm{e}}$ the deviatoric part of the small strain tensor. Using the chain rule, the Cauchy stress tensor can be described in the invariant space as follows:

$$\boldsymbol{\sigma} = \frac{\partial \psi^{\mathrm{e}}}{\partial \epsilon_v^{\mathrm{e}}} \frac{\partial \epsilon_v^{\mathrm{e}}}{\partial \boldsymbol{\epsilon}^{\mathrm{e}}} + \frac{\partial \psi^{\mathrm{e}}}{\partial \epsilon_s^{\mathrm{e}}} \frac{\partial \epsilon_s^{\mathrm{e}}}{\partial \boldsymbol{\epsilon}^{\mathrm{e}}}. \tag{6}$$

In the above, the mean pressure p and deviatoric (von Mises) stress q can be defined as:

$$p = \frac{\partial \psi^{\mathrm{e}}}{\partial \epsilon_v^{\mathrm{e}}} \equiv \frac{1}{3} \operatorname{tr}(\boldsymbol{\sigma}), \quad q = \frac{\partial \psi^{\mathrm{e}}}{\partial \epsilon_s^{\mathrm{e}}} \equiv \sqrt{\frac{3}{2}} \|\boldsymbol{s}\|, \tag{7}$$

where $\boldsymbol{s}$ is the deviatoric part of the Cauchy stress tensor. Thus, the Cauchy stress can be expressed in terms of the stress invariants as:

$$\boldsymbol{\sigma} = p\mathbf{1} + \sqrt{\frac{2}{3}} q \widehat{\boldsymbol{n}}, \tag{8}$$

where

$$\widehat{\boldsymbol{n}} = \boldsymbol{e}^{\mathrm{e}} / \left\| \boldsymbol{e}^{\mathrm{e}} \right\| = \sqrt{2/3} / \epsilon_s^{\mathrm{e}} \boldsymbol{e}^{\mathrm{e}}. \tag{9}$$

The H_2 training objective of Eq. (4) for the training samples $i \in [1, \ldots, N]$ can now be rewritten in terms of the two strain invariants:

$$\begin{aligned} \boldsymbol{W}', \boldsymbol{b}' = \underset{\boldsymbol{W},\boldsymbol{b}}{\operatorname{argmin}} \Big(\frac{1}{N} \sum_{i=1}^{N} \Big(& \gamma_1 \left\| \psi_i^{\mathrm{e}} - \widehat{\psi}_i^{\mathrm{e}} \right\|_2^2 + \gamma_4 \left\| p_i - \widehat{p}_i \right\|_2^2 \\ & + \gamma_5 \left\| q_i - \widehat{q}_i \right\|_2^2 + \sum_{\alpha=1}^{2} \sum_{\beta=1}^{2} \gamma_6 \left\| D_{\alpha\beta,i}^{\mathrm{e}} - \widehat{D}_{\alpha\beta,i}^{\mathrm{e}} \right\|_2^2 \Big) \Big), \end{aligned} \tag{10}$$

where:

$$D_{11}^{\mathrm{e}} = \frac{\partial^2 \psi}{\partial \epsilon_v^{\mathrm{e}\,2}}, \quad D_{22}^{\mathrm{e}} = \frac{\partial^2 \psi}{\partial \epsilon_s^{\mathrm{e}\,2}}, \quad \text{and} \quad D_{12}^{\mathrm{e}} = D_{21}^{\mathrm{e}} = \frac{\partial^2 \psi}{\partial \epsilon_v^{\mathrm{e}} \partial \epsilon_s^{\mathrm{e}}}. \tag{11}$$

To aid the implementation by a third party, a pseudocode for the supervised learning of the isotropic hyperelastic energy functional is provided (see Algorithm 1).

Algorithm 1 H^2 training of a neural network hyperelastic energy functional for isotropic material.

Require: Data set of N samples: strain measures $\boldsymbol{\epsilon}^{\mathrm{e}}$, energy measures ψ^{e}, stress measures $\boldsymbol{\sigma}$, and stiffness measures $\boldsymbol{c}^{\mathrm{e}}$.

1. Project data samples onto invariant space

Initialize empty set of training samples $\left(\epsilon_{v,i}^{\mathrm{e}}, \epsilon_{s,i}^{\mathrm{e}}, p_i, q_i, D_{11,i}^{\mathrm{e}}, D_{22,i}^{\mathrm{e}}, D_{12,i}^{\mathrm{e}}\right)$ for i in [0, ..., N].

for i in [0,...,N] **do**

Compute $\left(\epsilon_{v,i}^{\mathrm{e}}, \epsilon_{s,i}^{\mathrm{e}}, p_i, q_i, D_{11,i}^{\mathrm{e}}, D_{22,i}^{\mathrm{e}}, D_{12,i}^{\mathrm{e}}\right)$, given $\left(\boldsymbol{\epsilon}_i^{\mathrm{e}}, \boldsymbol{\sigma}_i, \boldsymbol{c}_i^{\mathrm{e}}\right)$ (see Section 2.2.1).

Rescale $\left(\epsilon_{v,i}^{\mathrm{e}}, \epsilon_{s,i}^{\mathrm{e}}, p_i, q_i, D_{11,i}^{\mathrm{e}}, D_{22,i}^{\mathrm{e}}, D_{12,i}^{\mathrm{e}}, \psi_i^{\mathrm{e}}\right)$ into $\left(\overline{\epsilon_{v,i}^{\mathrm{e}}}, \overline{\epsilon_{s,i}^{\mathrm{e}}}, \overline{\psi_i^{\mathrm{e}}}, \overline{p_i}, \overline{q_i}, \overline{D_{11,i}^{\mathrm{e}}}, \overline{D_{22,i}^{\mathrm{e}}}, \overline{D_{12,i}^{\mathrm{e}}}, \overline{\psi_i^{\mathrm{e}}}\right)$ via Eq. (12).

end for

2. Train neural network $\widehat{\psi}^{\mathrm{e}}(\overline{\epsilon_v^{\mathrm{e}}}, \overline{\epsilon_s^{\mathrm{e}}})$ with loss function Eq. (10).

3. Output trained energy functional $\widehat{\psi}^{\mathrm{e}}$ neural network and **exit**.

Remark 2 (*Rescaling of the Training Data*)**.** In every loss function in this work, we have introduced scaling coefficients γ_α to remind the readers that it is possible to change the weighting to adjust the relative importance of different terms in the loss function. These scaling coefficients may also be viewed as the weighting function in a multi-objective optimization problem. In practice, we have normalized all data to avoid the vanishing or exploding gradient problem that may occur during the back-propagation process [45]. As such, normalization is performed before the training as a pre-processing step. The X_i sample of a measure X is scaled to a unit interval via,

$$\overline{X_i} := \frac{X_i - X_{\min}}{X_{\max} - X_{\min}}, \tag{12}$$

where $\overline{X_i}$ is the normalized sample point. $X_{\min}$ and $X_{\max}$ are the minimum and maximum values of the measure X in the training data set such that all different types of data used in this paper (e.g. energy, stress, stress gradient, stiffness) are all normalized within the range [0, 1].

2.3. Training of an evolving yield function as a level set

This section introduces the theoretical framework that regards the evolution of a yield surface as a level set evolution problem. To facilitate a geometrical interpretation without the burden of generating a large database, we restrict our attention to the construction of a yield function that remains pressure-insensitive but may otherwise evolve in any arbitrary way on the π-plane, including moving, expanding, contracting, and deforming the elastic region. The goal of the supervised learning task is to determine the optimal way the yield function should evolve such that it is consistent with the observed experimental data collected after the plastic yielding and obeying the thermodynamic constraints that can be interpreted geometrically in the stress space.

2.3.1. Reducing the dimension of the data by leveraging symmetries

Here, we provide a brief review of the geometrical interpretation of the stress space and how it can be used to reduce the dimensions of the data representation and reduce the difficulty of the machine learning tasks. In this work, we consider a convex elastic domain $\mathbb{E}$ defined by a yield surface f. This yield function is a function of Cauchy stress $\boldsymbol{\sigma}$ and the internal variable ξ that represent the history-dependent behavior of the material, i.e., (cf. [30]),

$$\xi = \int_0^t \dot{\lambda} dt, \tag{13}$$

where ξ is a monotonically increasing function of time and $\dot{\lambda}$ is the rate of change of the plastic multiplier where $\dot{\boldsymbol{\epsilon}}^p = \dot{\lambda}\partial g/\partial \boldsymbol{\sigma}$ and g is the plastic potential. The yield function returns a negative value in the elastic region and equals zero when the material is yielding. The stress on the boundary $f(\boldsymbol{\sigma}, \xi) = 0$ is, therefore, the yielding stress and all admissible stress belong to the closure of the elastic domain, i.e.,

$$\overline{\mathbb{E}} := \{(\boldsymbol{\sigma}, \xi) \in \mathbb{S} \times \mathbb{R}^1 | f(\boldsymbol{\sigma}, \xi) \leq 0\}. \tag{14}$$

First, we assume that the yield function depends only on the principal stress. This treatment reduces the dimension of the stress representation from six to three. Then, we assume that the plastic yielding is not sensitive to the mean pressure. As such, the shape of the yield surface in the principal stress space can be sufficiently described by a projection on the π-plane and, hence, further reduce the dimensions of the independent stress input from three to two. To further simplify the interpolation of the yield surface, we introduce a polar coordinate system on the π-plane such that different monotonic stress paths commonly obtained from triaxial tests can be easily described via the Lode's angle (see Fig. 2).

Recall that the π-plane refers to a projection of the principal stress space based on the space diagonal defined by $\sigma_1 = \sigma_2 = \sigma_3$. More specifically, the π-plane is defined by the equation:

$$\sigma_1 + \sigma_2 + \sigma_3 = 0. \tag{15}$$

The transformation from the original stress space coordinate system $(\sigma_1, \sigma_2, \sigma_3)$ to the π-plane can be decomposed into two specific rotations $\boldsymbol{R}$ and $\boldsymbol{R}''$ of the coordinate system (cf. [30]) such that,

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{Bmatrix} = \boldsymbol{R}'\boldsymbol{R}'' \begin{Bmatrix} \sigma_1'' \\ \sigma_2'' \\ \sigma_3'' \end{Bmatrix} = \begin{bmatrix} \sqrt{2}/2 & 0 & \sqrt{2}/2 \\ 0 & 1 & 0 \\ -\sqrt{2}/2 & 0 & \sqrt{2}/2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \sqrt{2/3} & 1/\sqrt{3} \\ 0 & -1/\sqrt{3} & \sqrt{2/3} \end{bmatrix} \begin{Bmatrix} \sigma_1'' \\ \sigma_2'' \\ \sigma_3'' \end{Bmatrix}. \tag{16}$$

For pressure-insensitive plasticity, σ_3'' is not needed, as the principal stress differences are a function of σ_1'' and σ_2'' and are independent of σ_3''. We opt to describe the stress states of the material on the π-plane using two stress invariants, the polar radius ρ and the Lode's angle θ [46]. These invariants are derived by solving the characteristic equation of the deviatoric component $\boldsymbol{s} \in \mathbb{S}$ of the Cauchy stress tensor, following [30]:

$$s^3 - J_2 s - J_3 = 0, \tag{17}$$

where s is a principal value of $\boldsymbol{s}$, and

$$J_2 = \frac{1}{2}\text{tr}(\boldsymbol{s}^2), \qquad J_3 = \frac{1}{3}\text{tr}(\boldsymbol{s}^3), \tag{18}$$

are respectively the second and third invariants of the tensor $\boldsymbol{s}$. Utilizing the identity:

$$\cos^3\theta - 3/4\cos\theta - 1/4\cos 3\theta = 0, \tag{19}$$

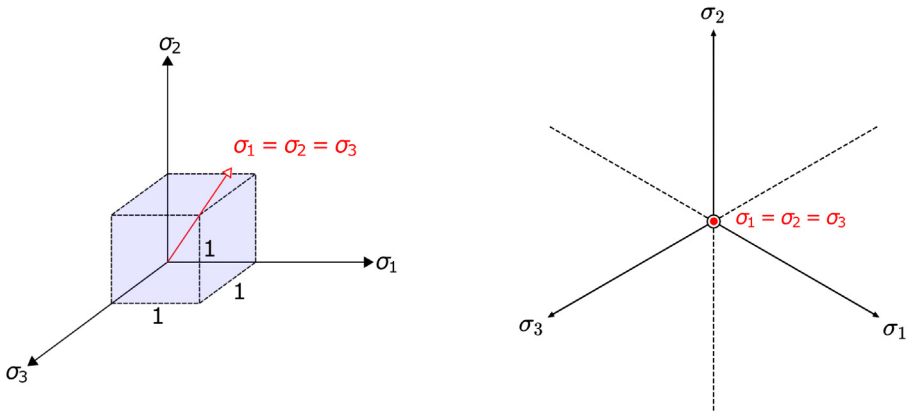


Fig. 2. The principle stress space (LEFT) and the corresponding π-plane (RIGHT). The π-plane is perpendicular to the space diagonal and is passing through the origin of the principal stress axes. Figure reproduced from [30].

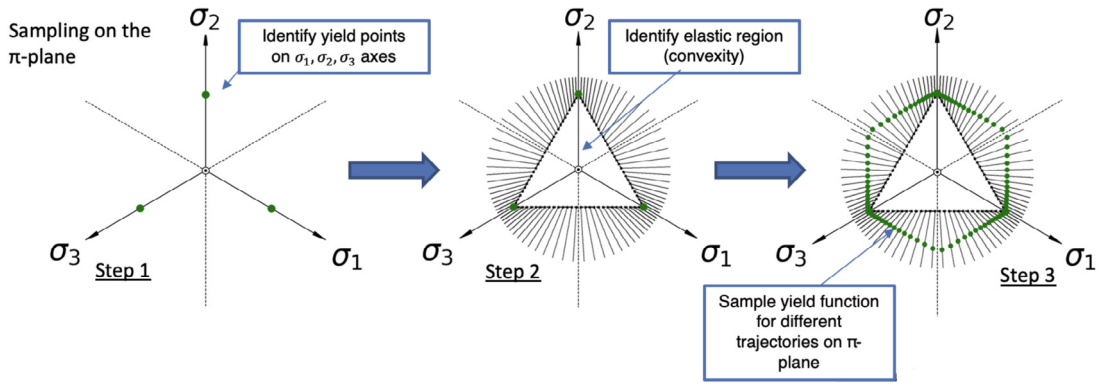


Fig. 3. The procedure to generate the polycrystal plasticity data set on the π-plane. Three simulations are first performed to locate the yielding points along the principal axes (LEFT). An initial trial elastic convex region is a triangle formed by connecting these three points (MIDDLE). The rest of the simulations are then performed outside of this triangle to refine the shape of the yield surface. The π-plane is split into splices of equal arc length and data points are sampled radially (RIGHT). The initial yielding point for each loading path is located once yielding is detected in the DNS.

and writing s in polar coordinates such that:

$$s = \rho \cos\theta, \tag{20}$$

and substituting in (17), the polar radius and the Lode's angle can be retrieved as:

$$\rho = 2\sqrt{J_2/3}, \qquad \text{and} \qquad \cos 3\theta = \frac{3\sqrt{3}J_3}{2J_2^{3/2}}. \tag{21}$$

In terms of the π-plane coordinates σ_1'' and σ_2'', the Lode's coordinates ρ and θ can be respectively written as:

$$\rho = \sqrt{\sigma_1''^2 + \sigma_2''^2}, \qquad \text{and} \qquad \tan\theta = \frac{\sigma_2''}{\sigma_1''}. \tag{22}$$

Thus, for an isotropic pressure-independent plasticity model, the yield surface can equivalently be described by an approximator using either the principal stresses σ_1, σ_2, and σ_3 or the stress invariant ρ, and, θ such that:

$$\overline{f}(\sigma_1, \sigma_2, \sigma_3, \xi) = \widehat{f}(\rho, \theta, \xi) = 0. \tag{23}$$

In this case, both the isotropy of the yield function and the symmetry along the hydrostatic axis significantly ease the training process. First, the material symmetry reduces the dimensionality of the data and, hence, reduces the

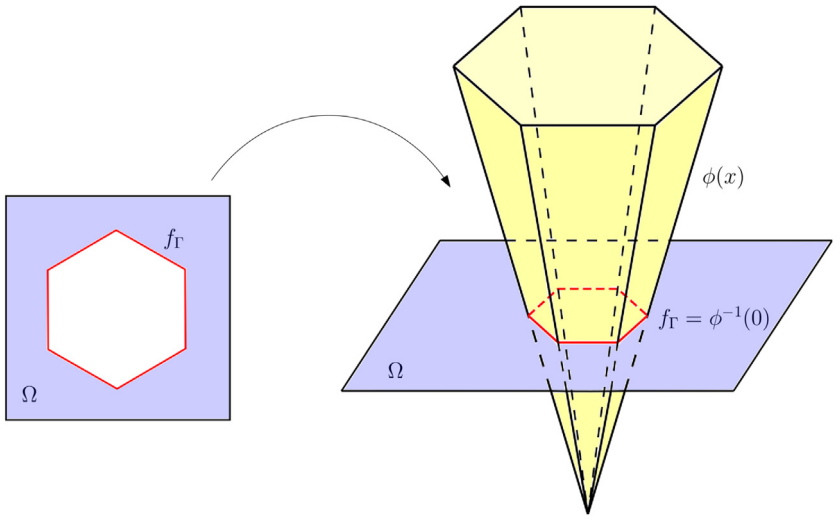


Fig. 4. Illustration of the relationship between the boundary of $\overline{\mathbb{E}}$ defined at f_Γ (yield surface) (LEFT) and the level set yield function defined everywhere in the stress space (RIGHT).

demand for data points [16,17]. Second, the geometrical interpretation of yield function on the π-plane provides clear guidance for more effective data exploration. In our numerical examples, the material database constitutes recorded constitutive responses obtained from direct numerical simulations of polycrystal assembles. The π-plane is partitioned by Lode's angles where each partitioned angle is assigned a stress path that moves along the radial direction on the π-plane. By assuming the convexity of the yield surface, we identify the boundary of the initial elastic region by linking the initial yielding points identified at the prescribed stress paths (see Fig. 3).

2.3.2. Data preparation for training of the yield function as a level set

Identifying the set of stress at which the initial plastic yielding occurs is a necessary but not sufficient condition to generate a yield surface. In fact, a yield surface $f(\boldsymbol{\sigma}, \xi)$ must be well-defined not just at $f = 0$ but also anywhere in the product space $\mathbb{S} \times R^1$. Another key observation is that, for the yield surface to function properly, the value of $f(\boldsymbol{\sigma}, \xi)$ inside and outside the yield surface may vary, provided that the orientation of the stress gradient remains consistent. For instance, consider two classical J_2 yield functions,

$$f_1(\boldsymbol{\sigma}, \xi) = \sqrt{2J_2} - \kappa \leq 0 \;\; ; \;\; f_2(\boldsymbol{\sigma}, \xi) = \sqrt{J_2} - \kappa/\sqrt{2} \leq 0. \tag{24}$$

These two models will yield identical constitutive responses except that, in each incremental step, the plastic multiplier deduced from f_1 is $\sqrt{2}$ times smaller than that of f_2, as the stress gradient of f_1 is $\sqrt{2}$ times larger than that of f_2. With these observations in mind, we introduce a level set approach where the yield surface is postulated to be a signed distance function level set and the evolution of the yield function is governed by a Hamilton–Jacobi equation that is not solved but generated from a supervised learning via these following steps.

1. **Generate auxiliary data points to train the signed distance function yield function**. In this first step, we first attempt to construct a signed distance function ϕ in the stress space where the internal variable is fixed on a given value, i.e. $\xi = \overline{\xi}$ when yielding. Let Ω be the solution domain of the stress space of which the signed distance function is defined. Assume that the yield function can be sufficiently described in the π-plane. For simplicity, we will adopt the polar coordinate system to parametrize the signed distance function ϕ that is used to train the yield surface, i.e.,

$$\boldsymbol{x}(\sigma_{11}, \sigma_{22}, \sigma_{33}, \sigma_{12}, \sigma_{23}, \sigma_{13}) = \overline{\boldsymbol{x}}(\sigma_1, \sigma_2, \sigma_3) = \widehat{\boldsymbol{x}}(\rho, \theta). \tag{25}$$

The signed distance function (see, for instance Fig. 4) is defined as

$$\phi(\widehat{\mathbf{x}}, t) = \begin{cases} d(\widehat{\mathbf{x}}) & \text{outside } f_\Gamma \text{(inadmissible stress)} \\ 0 & \text{on } f_\Gamma \text{(yielding)} \\ -d(\widehat{\mathbf{x}}) & \text{inside } f_\Gamma \text{ (elastic region)} \end{cases}, \tag{26}$$

where $d(\widehat{\boldsymbol{x}})$ is the minimum Euclidean distance between any point $\boldsymbol{x}$ of Ω and the interface $f_\Gamma = \{\widehat{\boldsymbol{x}} \in \mathbb{R}^2 | f(\widehat{\boldsymbol{x}}) = 0\}$, defined as:

$$d(\widehat{\boldsymbol{x}}) = \min\left(|\widehat{\boldsymbol{x}} - \widehat{\boldsymbol{x}}_\Gamma|\right), \tag{27}$$

where $\boldsymbol{x}_\Gamma$ is the yielding stress for a given ξ. The signed distance function is obtained by solving the Eikonal equation $|\nabla^{\widehat{\boldsymbol{x}}}\phi| = 1$ while prescribing the signed distance function as 0 at $\boldsymbol{x} \in f_\Gamma$. In the polar coordinate system, the Eikonal equation reads,

$$(\frac{\partial \phi}{\partial \rho})^2 + \frac{1}{\rho^2}(\frac{\partial \phi}{\partial \theta})^2 = 1. \tag{28}$$

Note there is a singularity at the polar coordinate of the π-plane at $\rho = 0$ and, hence, the origin point is not used as an auxiliary point to train the yield function. The Eikonal solution could be simply solved by a fast marching solver in 2D polar coordinates as well. It is noted that the selection of the signed distance function to be used as the level set representation is not limiting. Other level set functions are expected to fulfill the same purpose. However, the signed distance function was chosen for its simplicity to implement.
Fig. 5 shows several examples of signed distance functions converted for classical yield surfaces or deduced from direct numerical simulations.

2. **Obtain the velocity function to constitute the Hamilton–Jacobi hardening of the yield function** After we generate a sequence of signed distance functions for different ξ, we might introduce an inverse problem to obtain the velocity function for the Hamilton–Jacobi equation that evolves the signed distance function. Recall that the Hamilton–Jacobi equation may take the following forms:

$$\frac{\partial \phi}{\partial t} + \boldsymbol{v} \cdot \nabla^{\widehat{\boldsymbol{x}}}\phi = 0, \tag{29}$$

where $\boldsymbol{v}$ is the normal velocity field that defines the geometric evolution of the boundary and, in the case of plasticity, is chosen to describe the observed hardening mechanism. The velocity field is given by:

$$\boldsymbol{v} = F\boldsymbol{n}, \tag{30}$$

where F is a scalar function describing the magnitude of the boundary change and $\boldsymbol{n} = \nabla^{\widehat{\boldsymbol{x}}}\phi / |\nabla^{\widehat{\boldsymbol{x}}}\phi|$. Using $\nabla\phi \cdot \nabla\phi = |\nabla\phi|^2$ in Eq. (29), the level set Hamilton–Jacobi equation for a stationary yield function can be simplified as,

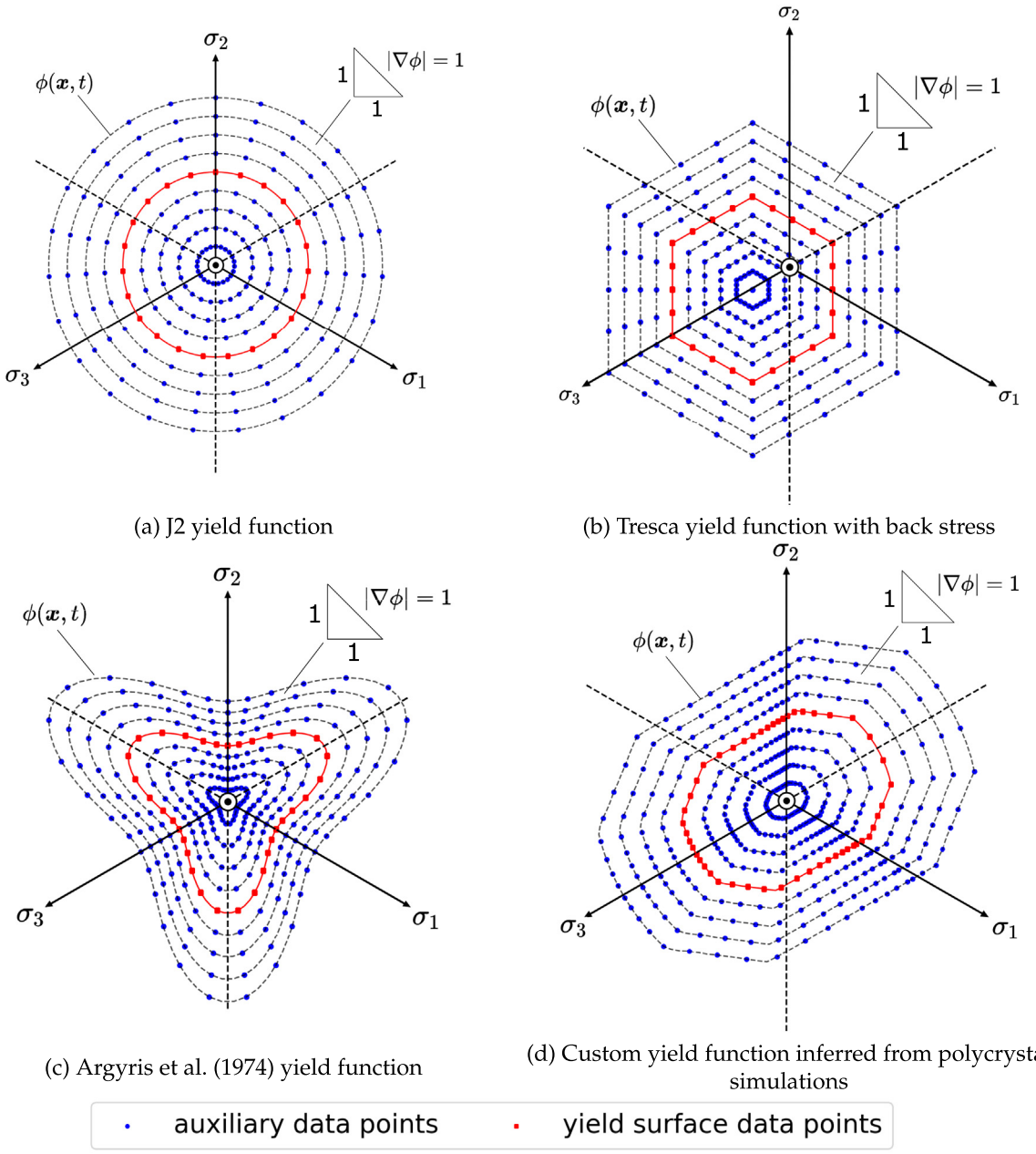
$$\frac{\partial \phi}{\partial t} + F|\nabla^{\widehat{\boldsymbol{x}}}\phi| = 0. \tag{31}$$

Note that t is a pseudo-time. Since the snapshot of ϕ we obtained from Step 1 remains a signed distance function, then $|\nabla^{\widehat{\boldsymbol{x}}}\phi| = 1$. Next, we replace the pseudo-time t with ξ. Assuming that the experimental data collected from different stress paths are collected data points N times beyond the initial yielding point, each time with the same incremental plastic strain $\Delta\lambda$, then Step 1 will provide us a collection of signed distance functions $\{\phi_0, \phi_1, \ldots, \phi_{n+1}\}$ corresponding to $\{\xi_0, \xi_1, \ldots, \xi_{n+1}\}$. Then, the corresponding velocity function can be obtained via finite difference, i.e.,

$$F_i \approx \frac{\phi_i - \phi_{i+1}}{\xi_{i+1} - \xi_i}, \tag{32}$$

where $F_i(\rho, \theta) = F(\rho, \theta, \xi_i)$ and $i = 0, 1, 2, \ldots, n+1$. By setting the signed distance function that fulfills Eq. (31) as the yield function, i.e., $f(\rho, \theta, \xi) = \phi(\rho, \theta, \xi)$, we may use experimental data generated from the loading paths demonstrated in Fig. 3 to train a neural network that predicts a new yield function or velocity function for an arbitrary ξ (see Fig. 6).

More importantly, we show that the evolution of the yield function can be modeled as a level set evolving according to a Hamilton–Jacobi equation. This knowledge may open up many new possibilities to capture hardening without any hand-crafted treatment. To overcome the potential cost to solve the Hamilton–Jacobi equation, we will introduce a supervised learning procedure to obtain the updated yield function for a given strain history represented by the internal variable ξ without explicitly solving the Hamilton–Jacobi equation (see the next sub-sections). Consequently, this treatment will enable us to create a generic elasto-plasticity framework that can replace the hard-crafted yield functions and hardening laws without the high computational costs and the burden of repeated modeling trial-and-errors.



(a) J2 yield function

(b) Tresca yield function with back stress

(c) Argyris et al. (1974) yield function

(d) Custom yield function inferred from polycrystal simulations

Fig. 5. The inferred yield surface (red curves) and the auxiliary data points (blue dots) obtained by solving the level set re-initialization problem that generates the signed distance function (blue contours) for four yield surfaces. The yield function $\phi(\boldsymbol{x}, t)$ is converted into a signed distance function with the contour $\phi(\boldsymbol{x}, t) = 0$ fixed. The isocontour curves represent the projection of the signed distance function level set on the π-plane [47]. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2.3.3. Training a yield function with associative plastic flow

Assuming an associative flow rule, the generalized Hooke's law utilizing a yield function neural network approximator $\widehat{f}$ can be written in rate form as:

$$\dot{\boldsymbol{\sigma}} = \boldsymbol{c}^{\mathrm{e}} : \left(\dot{\boldsymbol{\epsilon}} - \dot{\lambda} \frac{\partial \widehat{f}}{\partial \boldsymbol{\sigma}} \right). \tag{33}$$

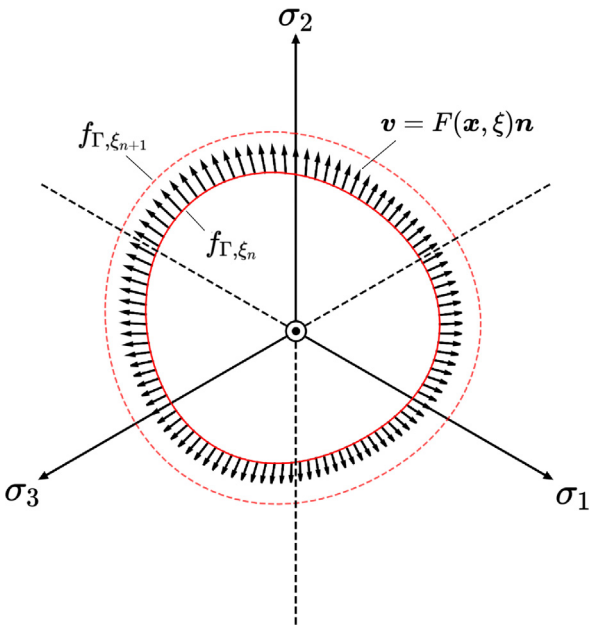


Fig. 6. The evolution of the yield surface f_Γ is connected to a level set $\phi(\boldsymbol{x},\xi)$ extension problem. The velocity field $\boldsymbol{v}$ of the Hamilton–Jacobi equation (31) emulates the material hardening law. The yield surface evolution and the velocity field are inferred from the data through the neural network training.

In the incremental form, the predictor–corrector scheme is written as:

$$\boldsymbol{\sigma}_{n+1} = \boldsymbol{\sigma}^{\mathrm{tr}}_{n+1} - \Delta\lambda \boldsymbol{c}^{\mathrm{e}}_{n+1} : \left.\frac{\partial \widehat{f}}{\partial \boldsymbol{\sigma}}\right|_{n+1}, \tag{34}$$

where

$$\boldsymbol{\sigma}^{\mathrm{tr}}_{n+1} = \boldsymbol{\sigma}_n + \boldsymbol{c}^{\mathrm{e}}_{n+1} : \Delta\boldsymbol{\epsilon} \tag{35}$$

The strain and stress tensor predictors can be written in the spectral form as follows:

$$\boldsymbol{\sigma}^{\mathrm{tr}}_{n+1} = \sum_{A=1}^{3} \sigma^{\mathrm{tr}}_{A,n+1} \boldsymbol{n}^{\mathrm{tr}(A)}_{n+1} \otimes \boldsymbol{n}^{\mathrm{tr}(A)}_{n+1}, \qquad \boldsymbol{\epsilon}^{\mathrm{etr}}_{n+1} = \sum_{A=1}^{3} \epsilon^{\mathrm{etr}}_{A,n+1} \boldsymbol{n}^{\mathrm{tr}(A)}_{n+1} \otimes \boldsymbol{n}^{\mathrm{tr}(A)}_{n+1}. \tag{36}$$

The predictor–corrector scheme can be rewritten in spectral form, omitting the subscript $(n+1)$,

$$\sum_{A=1}^{3} \sigma_A \boldsymbol{n}^{(A)} \otimes \boldsymbol{n}^{(A)} = \sum_{A=1}^{3} \sigma^{\mathrm{tr}}_A \boldsymbol{n}^{\mathrm{tr}(A)} \otimes \boldsymbol{n}^{\mathrm{tr}(A)} - \Delta\lambda \sum_{A=1}^{3} \left(\sum_{B=1}^{3} a^{\mathrm{e}}_{AB} \widehat{f}_B \right) \boldsymbol{n}^{(A)} \otimes \boldsymbol{n}^{(A)}, \tag{37}$$

$$\boldsymbol{n}^{(A)} \otimes \boldsymbol{n}^{(A)} = \boldsymbol{n}^{\mathrm{tr}(A)} \otimes \boldsymbol{n}^{\mathrm{tr}(A)}. \tag{38}$$

By assuming that the plastic flow obeys the normality rule, we may use the observed plastic flow from the data to regularize the shape of the evolving yield function. To do so, we will leverage the fact that we have already obtained an elastic energy functional from the previous training. The plastic deformation mode can then be obtained by the difference between the trial and the true Cauchy stress at each incremental step where the data are recorded in an experiment or direct numerical simulations, i.e.,

$$\begin{aligned} &\sigma_A = \sigma^{\mathrm{tr}}_A - \Delta\lambda \sum_{B=1}^{3} a^{\mathrm{e}}_{AB} f_B, \\ &f_A = \partial f / \partial \sigma_A = \frac{\epsilon^{\mathrm{etr}}_A - \epsilon^{\mathrm{e}}_A}{\Delta\lambda} \text{ for } A = 1, 2, 3. \end{aligned} \tag{39}$$

At each incremental step of the data-generating simulations, we know the total strain and total stress. Knowing the underlying hyperelastic model, we can utilize an inverse mapping to estimate the elastic strain from the current total stress and hence determine the plastic strain. The quantities f_1, f_2, f_3 correspond to the amount of plastic flow in the principal directions $A = 1, 2, 3$. A neural network approximator of the yield function should have adequately accurate stress derivatives that are necessary for the implementation of the return mapping algorithm, discussed in Section 3, and so as to provide an accurate plastic flow, in the case of associative plasticity. The normalized plastic flow direction vector $\overline{\boldsymbol{f}}_{\text{norm}}$ can be defined as

$$\overline{\boldsymbol{f}}_{\text{norm}} = \langle \widehat{f}_1, \widehat{f}_2, \widehat{f}_3 \rangle / \| \langle \widehat{f}_1, \widehat{f}_2, \widehat{f}_3 \rangle \|, \tag{40}$$

and holds information about the yield function shape in the π-plane.

In the case of the simple MLP feed-forward network, the network can be seen as an approximator function $\widehat{f} = \widehat{f}(\rho, \theta, \xi | \boldsymbol{W}, \boldsymbol{b})$ of the true yield function level set f with input the Lode's coordinates ρ, θ, and the hardening parameter ξ, parametrized by weights $\boldsymbol{W}$ and biases $\boldsymbol{b}$. A classical training objective, following an L_2 norm, would only constrain the predicted yield function values. The corresponding training objective is to minimize the discrepancy measured at N number of sample points $(\widehat{\boldsymbol{x}}, \xi) \in \mathbb{S} \times \mathbb{R}^1$ reads,

$$\boldsymbol{W}', \boldsymbol{b}' = \underset{\boldsymbol{W}, \boldsymbol{b}}{\operatorname{argmin}} \left(\frac{1}{N} \sum_{i=1}^{N} \gamma_7 \left\| f_i - \widehat{f}_i \right\|_2^2 \right), \tag{41}$$

where $f_i = f(\widehat{\boldsymbol{x}}_i, \xi_i)$ and $\widehat{f}_i = f(\widehat{\boldsymbol{x}}_i, \xi_i)$. A second training objective can be modeled after an H_1 norm, constraining both f and its first derivative with respect to the stress state $\sigma_1, \sigma_2, \sigma_3$. For a neural network approximator parametrized as $\overline{f} = \overline{f}(\sigma_1, \sigma_2, \sigma_3, \xi | \boldsymbol{W}, \boldsymbol{b})$ using the principal stresses as inputs, this training objective for the training samples $i \in [1, \dots, N]$ would have the following form:

$$\boldsymbol{W}', \boldsymbol{b}' = \underset{\boldsymbol{W}, \boldsymbol{b}}{\operatorname{argmin}} \left(\frac{1}{N} \sum_{i=1}^{N} \left(\gamma_7 \left\| f_i - \overline{f}_i \right\|_2^2 + \sum_{A=1}^{3} \gamma_8 \left\| \frac{\partial f_i}{\partial \boldsymbol{\sigma}_{\boldsymbol{A}i}} - \frac{\partial \overline{f}_i}{\partial \boldsymbol{\sigma}_{\boldsymbol{A}i}} \right\|_2^2 \right) \right). \tag{42}$$

Utilizing an equivalent representation of the stress state with Lode's coordinates in the π-plane, the above training objective can further be simplified. The normalized flow direction vector $\overline{\boldsymbol{f}}_{\text{norm}}$ in Lode's coordinates can solely be described using an angle θ_f since the vector has a magnitude equal to unity. To constrain the flow direction angle, we modify the loss function of this higher-order training objective by adding a distance function metric between two rotation tensors $\boldsymbol{R}_{\theta,i}$, $\boldsymbol{R}_{\widehat{\theta},i}$, corresponding to $\theta_{f,i}$ and $\widehat{\theta}_{f,i}$ — the flow vector directions in the π-plane for the data and approximated yield function respectively for the ith sample. The two rotation tensors belong to the Special Orthogonal Group, SO(3) and the metric is based on the distance from the identity matrix. For the ith sample, the rotation related term can be calculated as:

$$\overline{\Phi}_i = \left\| \boldsymbol{I} - \boldsymbol{R}_{\theta,i} \left(\boldsymbol{R}_{\widehat{\theta},i} \right)^T \right\|_F = \sqrt{2 \left[3 - \operatorname{tr} \left[\boldsymbol{R}_{\theta,i} \left(\boldsymbol{R}_{\widehat{\theta},i} \right)^T \right] \right]}, \tag{43}$$

where $\| \cdot \|_F$ is the Frobenius norm. For a neural network approximator parametrized via the Lode's coordinates as input, i.e. $\widehat{f} = \widehat{f}(\rho, \theta, \xi | \boldsymbol{W}, \boldsymbol{b})$, the Sobolev training objective for the training samples $i \in [1, \dots, N]$ reads,

$$\boldsymbol{W}', \boldsymbol{b}' = \underset{\boldsymbol{W}, \boldsymbol{b}}{\operatorname{argmin}} \left(\frac{1}{N} \sum_{i=1}^{N} \left(\gamma_7 \left\| f_i - \widehat{f}_i \right\|_2^2 + \gamma_9 \overline{\Phi}_i \right) \right), \tag{44}$$

where we minimize both the discrepancy of the yield function and the direction of the gradient in the stress space.

Remark 3 (*Discrete Data Points for the Yield Function*)**.** Note that the training of the yield function involves not just the points at $f(\rho, \theta) = 0$ but also the new auxiliary data generated from the re-initialization of the level set/yield function. Strictly speaking, the accuracy of the elasto-plastic responses only depends on how well the boundary of the admissible stress range $f(\rho, \theta) = 0$ is kept track of. However, knowing the yield function values inside and outside the admission range is helpful for evolving the yield function with sufficient smoothness. To emphasize the importance of data across $f(\rho, \theta) = 0$, we may introduce a higher weighting factor of these data points for Eq. (44).

Algorithm 2 Training of a pressure independent isotropic yield function level set neural network.

Require: Data set of N samples: stress measures $\boldsymbol{\sigma}$ at yielding and accumulated plastic strain $\overline{\epsilon}_p$, an L_{levels} number of levels (isocontours) for the constructed signed distance function level set (data augmentation), and a parameter $\zeta > 1$ for the radius range of the constructed signed distance function.

1. Project stress onto π-plane

Initialize empty set of π-plane projection training samples (ρ_i, θ_i) for i in $[0, ..., N]$.
for i in [0,...,N] **do**
Spectrally decompose $\boldsymbol{\sigma}_i = \sum_{A=1}^{3} \sigma_{A,i} \boldsymbol{n}_i^{(A)} \otimes \boldsymbol{n}_i^{(A)}$.
Transform $\left(\sigma_{1,i}, \sigma_{2,i}, \sigma_{3,i}\right)$ into $\left(\sigma''_{1,i}, \sigma''_{2,i}, \sigma''_{3,i}\right)$ via Eq. (16).
$\rho_i \leftarrow \sqrt{\sigma''^2_{1,i} + \sigma''^2_{2,i}}$
$\theta_i \leftarrow \tan^{-1}\left(\frac{\sigma''_{2,i}}{\sigma''_{1,i}}\right)$
end for

2. Construct yield function level set (data augmentation)

Initialize empty set of augmented training samples $(\rho_m, \theta_m, \overline{\epsilon}_{p,m}, f_m)$ for m in $[0, ..., N \times L_{\text{levels}}]$.
$m \leftarrow 0$.
for i in [0,...,N] **do**
for j in [0,...,L_{levels}] **do**
$\rho_m \leftarrow \left(\frac{\zeta j}{L_{\text{levels}}}\right) \rho_i$ ▷ the signed distance function is constructed for a radius range of $[0, \zeta\rho_i]$
$\theta_m \leftarrow \theta_i$
$\overline{\epsilon}_{p,m} \leftarrow \overline{\epsilon}_{p,i}$
$f_m \leftarrow \left(\frac{\zeta j}{L_{\text{levels}}}\right) \rho_i - \rho_i$ ▷ the signed distance function value range is $[-\rho_i, (\zeta - 1)\rho_i]$
Rescale $(\rho_m, \theta_m, \overline{\epsilon}_{p,m}, f_m)$ into $(\overline{\rho_m}, \overline{\theta_m}, \overline{\overline{\epsilon}_{p,m}}, \overline{f_m})$ via Eq. (12).
$m \leftarrow m + 1$
end for
end for

3. Train neural network $\widehat{f}(\overline{\rho_m}, \overline{\theta_m}, \overline{\overline{\epsilon}_{p,m}})$ with loss function Eq. (41).

4. Output trained yield function $\widehat{f}$ neural network and exit.

2.3.4. Training yield function and non-associative plastic flow

Here, we present the training for the plastic flow without assuming that the plastic flow follows the normality rule. As such, the yield function and the plastic flow must be trained separately. We adopt the idea of generalized plasticity in which the plastic flow direction is directly deduced from the Sobolev training of a neural network on the experimental data [2].

The yield function training is similar to the associative flow cases, except that the stress gradient of the yield function cannot be directly deduced from the plastic flow due to the non-associative flow rule. Nevertheless, the stress gradient of the yield function may still be constrained by the convexity (if there is no intended phase transition that requires non-convexity of the yield function). Recall that the convexity requires

$$(\boldsymbol{\sigma}^* - \boldsymbol{\sigma}) : \frac{\partial \widehat{f}}{\partial \boldsymbol{\sigma}} \leq 0, \tag{45}$$

where $\boldsymbol{\sigma}^*$ is an arbitrary stress. One necessary condition we can incorporate as a thermodynamic constraint is a special case where we simply set $\boldsymbol{\sigma}^* = 0$, as such we obtain,

$$\sum_{A=1}^{3} \sigma_A \widehat{f}_A \geq 0. \tag{46}$$

One way to enforce that constraint is to apply a penalty term for the loss function in Eq. (41), e.g.,

$$w_{\text{nnp}}\ \text{sign}(-\sum_{A=1}^{3}\sigma_A\widehat{f}_A), \tag{47}$$

where this term will not be activated if the learned yield function is obeying the convexity. However, the sign operator may lead to a jump of the loss function, which is not desirable for training. As a result, a regularized Heaviside step function can be used to replace the sign operator, for instance,

$$\text{sign}^{\text{approx}}(-\sigma_A\widehat{f}_A) = \frac{1}{2}+\frac{1}{2}\tanh(-k\sigma_A\widehat{f}_A), \tag{48}$$

where k controls how sharp the transition is at $\sigma_A\widehat{f}_A = 0$. As shown in our numerical experiments, this additional term may not be required if the raw experimental data itself does not violate the thermodynamic restriction. To obtain a plastic flow, we again obtain the flow information incrementally from the experimental data via the following equations, i.e.,

$$\begin{aligned}\sigma_A &= \sigma_A^{\text{tr}} - \Delta\lambda\sum_{B=1}^{3}a_{AB}^{\text{e}}g_B,\\ g_A &= \partial g/\partial\sigma_A = \frac{\epsilon_A^{\text{etr}}-\epsilon_A^{\text{e}}}{\Delta\lambda}\ \text{for}\ A=1,2,3.\end{aligned} \tag{49}$$

We can gather the plastic flow information by post-processing the simulation data, similar to Eq. (39). Once the plastic flow g_A is determined incrementally for different ξ, we then introduce another supervised learning that reads,

$$\boldsymbol{W}',\boldsymbol{b}' = \underset{\boldsymbol{W},\boldsymbol{b}}{\text{argmin}}\left(\frac{1}{N}\sum_{i=1}^{N}\left(\gamma_{10}\left\|g_{A,i}-\widehat{g}_{A,i}\right\|_2^2\right)\right). \tag{50}$$

The non-negative plastic work is the thermodynamic constraint that requires $\dot{W}^p = \boldsymbol{\sigma} : \dot{\boldsymbol{\epsilon}}^p \geq 0$. The corresponding incremental form for isotropic material reads,

$$\Delta W = \sigma_{n+1} : \Delta\boldsymbol{\epsilon}^p = \Delta\lambda\sum_{A=1}^{3}\sigma_A\widehat{g}_A \geq 0. \tag{51}$$

Notice that the stress beyond the initial yielding point satisfies the yield function $f = 0$. As a result, this inequality can be recast as an additional term for the loss function that trains the yield function (Eq. (44)) such that

$$w_{\text{nnp}}\ \text{sign}(-\Delta\lambda\sum_{A=1}^{3}\sigma_A\widehat{g}_A), \tag{52}$$

where w_{nnp} is the penalty parameter. Notice that when the non-negative plastic work is fulfilled during the training of the neural network, the penalty term would not be activated and will not affect the back-propagation step. Furthermore, if the yield function is convex and the flow rule is associative, this constraint is always fulfilled and, hence not necessary.

3. Implementation highlights: return mapping algorithm with automatic differentiation

Here, we provide a review of the implementation of a fully implicit stress integration algorithm used for the proposed Hamilton–Jacobi hardening framework. For isotropic materials where the elastic strain and stress are co-axial, the stress integration can be done via spectral decomposition as shown in Alg. 3. An upshot of the proposed method is that there is only a small modification necessary to incorporate the Hamilton–Jacobi hardening and the generalized plasticity.

In this current work — unless otherwise stated, all the necessary information for the return mapping algorithm about the elastic and plastic and constitutive responses is derived from the trained neural networks of the hyperelastic energy functional and yield function respectively using the Keras [48] and Tensorflow [49] libraries. No additional explicit forms of constitutive laws are defined. Furthermore, the algorithm requires that all the strain and stress

Algorithm 3 Return mapping algorithm in strain-space in principal axes for an isotropic hyperelastic–plastic model.

Require: Hyperelastic energy functional $\widehat{\psi}^{\mathrm{e}}$ neural network (see Algorithm 1) and yield function $\widehat{f}$ neural network (see Algorithm 2).

1. Compute trial elastic strain

Compute $\boldsymbol{\epsilon}^{\mathrm{e}\,\mathrm{tr}}_{n+1} = \boldsymbol{\epsilon}^{\mathrm{e}}_{n} + \Delta\boldsymbol{\epsilon}$.

Spectrally decompose $\boldsymbol{\epsilon}^{\mathrm{e}\,\mathrm{tr}}_{n+1} = \sum_{A=1}^{3} \epsilon^{\mathrm{e}\,\mathrm{tr}}_{A} \boldsymbol{n}^{\mathrm{tr}(A)} \otimes \boldsymbol{n}^{\mathrm{tr}(A)}$.

2. Compute trial elastic stress

Compute $\sigma^{\mathrm{tr}}_{A} = \partial\widehat{\psi}^{\mathrm{e}}/\partial\epsilon^{\mathrm{e}}_{A}$ at $\epsilon^{\mathrm{e}\,\mathrm{tr}}_{n+1}$.

3. Check yield condition and perform return mapping if loading is plastic

if $\widehat{f}\left(\sigma^{\mathrm{tr}}_{1}, \sigma^{\mathrm{tr}}_{2}, \sigma^{\mathrm{tr}}_{3}, \xi_{n}\right) \leq 0$ **then**

Set $\boldsymbol{\sigma}_{n+1} = \sum_{A=1}^{3} \sigma^{\mathrm{tr}}_{A} \boldsymbol{n}^{\mathrm{tr}(A)} \otimes \boldsymbol{n}^{\mathrm{tr}(A)}$ and exit.

else

Solve for $\epsilon^{\mathrm{e}}_{1}, \epsilon^{\mathrm{e}}_{2}, \epsilon^{\mathrm{e}}_{3}$, and ξ_{n+1} such that $\widehat{f}\left(\sigma^{\mathrm{tr}}_{1}, \sigma^{\mathrm{tr}}_{2}, \sigma^{\mathrm{tr}}_{3}, \xi_{n+1}\right) = 0$.

Compute $\boldsymbol{\sigma}_{n+1} = \sum_{A=1}^{3} \left(\partial\widehat{\psi}^{\mathrm{e}}/\partial\epsilon^{\mathrm{e}}_{A}\right) \boldsymbol{n}^{\mathrm{tr}(A)} \otimes \boldsymbol{n}^{\mathrm{tr}(A)}$ and exit.

end if

variables are in the principal axes. However, as it was stated in Section 2, in order to facilitate the machine learning algorithms, we have opted to train with the strain invariants $\epsilon^{\mathrm{e}}_{v}$ and $\epsilon^{\mathrm{e}}_{s}$, and the stress invariants ρ and θ. Integrating the machine learning algorithms with the return mapping algorithm requires a set of coordinate system transformations, which, in turn, require the calculation of the partial derivatives of said transformations to use in the chain rule formulation. The partial derivative calculation is performed using the Autograd library [50] for automatic differentiation.

Autograd enables the automatic calculation of the partial derivatives of explicitly defined functions. Thus, we can easily define the transformation of any input parameter space for our neural networks to the principal space and readily have the necessary partial derivatives for the chain rule implementation. This allows us to use equivalent expressions of our neural network approximators $\widehat{\psi}^{\mathrm{e}}(\epsilon^{\mathrm{e}}_{v}, \epsilon^{\mathrm{e}}_{s})$ and $\widehat{f}\,(\rho, \theta, \xi)$ in the principal space, such that:

$$\widehat{\psi}^{\mathrm{e}}(\epsilon^{\mathrm{e}}_{v}, \epsilon^{\mathrm{e}}_{s}) = \widehat{\psi}^{\boldsymbol{\epsilon}}_{\mathrm{principal}}\left(\epsilon^{\mathrm{e}}_{1}, \epsilon^{\mathrm{e}}_{2}, \epsilon^{\mathrm{e}}_{3}\right) \qquad \text{and} \qquad \widehat{f}\,(\rho, \theta, \xi) = \widehat{f}_{\mathrm{principal}}\left(\epsilon^{\mathrm{e}}_{1}, \epsilon^{\mathrm{e}}_{2}, \epsilon^{\mathrm{e}}_{3}, \xi\right). \tag{53}$$

In this work, integrating the neural network approximators in the return mapping requires the following coordinate system transformations $(\epsilon^{\mathrm{e}}_{v}, \epsilon^{\mathrm{e}}_{s}) \longleftrightarrow (\epsilon^{\mathrm{e}}_{1}, \epsilon^{\mathrm{e}}_{2}, \epsilon^{\mathrm{e}}_{3})$, $(\rho, \theta) \longleftrightarrow (\sigma_1, \sigma_2, \sigma_3)$, $\left(\sigma''_1, \sigma''_2\right) \longleftrightarrow (\sigma_1, \sigma_2, \sigma_3)$, and $\left(\sigma''_1, \sigma''_2\right) \longleftrightarrow (\rho, \theta)$. These transformations require a large number of chain rules increasing the possibility of formulation errors, as well as rendering replacing the networks' input space less flexible. Thus, we opt for the automation of this process using Autograd.

Due to the fact that the machine learning training has created a mapping that automatically generates an updated yield function whenever the internal variables ξ are updated, there is no need to add additional constraints for the linearized hardening rules. The return mapping algorithm can be described with a system of four equations that are solved iteratively. For a local iteration k, we solve for the solution vector $\boldsymbol{x}$ such that $\boldsymbol{A}^{k} \cdot \Delta\boldsymbol{x} = \boldsymbol{r}\left(\boldsymbol{x}^{k}\right)$, $\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^{k} - \Delta\boldsymbol{x}$, $k \leftarrow k+1$ until the residual norm $\|\boldsymbol{r}\|$ is below a set error threshold. The residual vector $\boldsymbol{r}$ and the local tangent $\boldsymbol{A}^{k}$ can be assembled for the calculation of $\boldsymbol{x}$ by a series of neural network evaluations and automatic differentiations, such that:

$$\boldsymbol{r}(\boldsymbol{x}) = \begin{Bmatrix} \epsilon^{\mathrm{e}}_{1} - \epsilon^{\mathrm{etr}}_{1} + \Delta\lambda\widehat{g}_{1} \\ \epsilon^{\mathrm{e}}_{2} - \epsilon^{\mathrm{etr}}_{2} + \Delta\lambda\widehat{g}_{2} \\ \epsilon^{\mathrm{e}}_{3} - \epsilon^{\mathrm{etr}}_{3} + \Delta\lambda\widehat{g}_{3} \\ \widehat{f}\left(\epsilon^{\mathrm{e}}_{1}, \epsilon^{\mathrm{e}}_{2}, \epsilon^{\mathrm{e}}_{3}, \xi\right) \end{Bmatrix}, \quad \boldsymbol{A}^{k} = \boldsymbol{r}'\left(\boldsymbol{x}^{k}\right) = \begin{bmatrix} c_{11} & c_{12} & c_{13} & \widehat{g}_{1} \\ c_{21} & c_{22} & c_{23} & \widehat{g}_{2} \\ c_{31} & c_{32} & c_{33} & \widehat{g}_{3} \\ \partial\widehat{f}/\partial\epsilon^{\mathrm{e}}_{1} & \partial\widehat{f}/\partial\epsilon^{\mathrm{e}}_{2} & \partial\widehat{f}/\partial\epsilon^{\mathrm{e}}_{3} & \partial\widehat{f}/\partial\xi \end{bmatrix}, \quad \text{and}$$

$$\boldsymbol{x} = \begin{Bmatrix} \epsilon_1^{\mathrm{e}} \\ \epsilon_2^{\mathrm{e}} \\ \epsilon_3^{\mathrm{e}} \\ \Delta\lambda \end{Bmatrix}, \tag{54}$$

where $\epsilon_I^{\mathrm{e\ tr}}$ is the trial state principal strain, $\widehat{g}_I = \partial \widehat{f}/\partial \sigma_I$ for an associative flow rule and:

$$c_{IJ} = \delta_{IJ} + \Delta\lambda \frac{\partial \widehat{g}_I}{\partial \epsilon_J^{\mathrm{e}}}, \quad I, J = 1, 2, 3. \tag{55}$$

This framework is also readily available to implement in finite element simulations (Section 5.5). We can assemble the algorithmic consistent tangent $\boldsymbol{c}_{n+1}$ in principal axes for a global Newton iteration n:

$$\boldsymbol{c}_{n+1} = \sum_{A=1}^{3}\sum_{B=1}^{3} a_{AB}\boldsymbol{m}^{(A)} \otimes \boldsymbol{m}^{(B)} + \frac{1}{2}\sum_{A=1}^{3}\sum_{B\neq A}\left(\frac{\sigma_B - \sigma_A}{\epsilon_B^{\mathrm{etr}} - \epsilon_A^{\mathrm{etr}}}\right)\left(\boldsymbol{m}^{(AB)} \otimes \boldsymbol{m}^{(AB)} + \boldsymbol{m}^{(AB)} \otimes \boldsymbol{m}^{(BA)}\right), \tag{56}$$

where $\boldsymbol{m}^{(AB)} = \boldsymbol{n}^{(A)} \otimes \boldsymbol{n}^{(B)}$ the matrix of elastic moduli in principal axes is given as:

$$a_{AB} \coloneqq \frac{\partial \sigma_A}{\partial \epsilon_B^{\mathrm{etr}}} = \sum_{C=1}^{3}\left(\frac{\partial^2 \widehat{\psi}^{\mathrm{e}}}{\partial \epsilon_A^{\mathrm{e}} \partial \epsilon_C^{\mathrm{e}}}\right)\frac{\partial \epsilon_C^{\mathrm{e}}}{\partial \epsilon_B^{\mathrm{etr}}}. \tag{57}$$

Utilizing Tensorflow and Autograd, the return mapping algorithm is fully generalized for any isotropic hyperelastic and yield function data-driven constitutive laws. It also allows for quick implementation of any parametrization of the neural network architectures. In future work, the framework can be extended to accommodate anisotropic responses, as well as architectures with complex internal variables and higher descriptive power.

4. Alternative comparison models for control experiments

In this section, we will briefly review some simple black-box neural network architectures that can be employed to predict the path-dependent plasticity behaviors. The predictive capabilities of these behaviors will be compared to our neural network elastoplasticity framework in Section 5.4. Three different architectures will be designed for comparison with our framework: a multi-step feed-forward network, a recurrent GRU network, and a 1-D convolutional network. All of these networks demonstrate the ability to capture path-dependent behavior utilizing different memory mechanisms.

The first architecture is a feed-forward network that takes the current strain as well as strain and stress from the previous time-step to predict the stress at the current time step. The feed-forward architecture consists of fully-connected Dense layers that have the following formulation in matrix form:

$$\boldsymbol{h}_{\mathrm{dense}}^{(l+1)} = a\left(\boldsymbol{h}^{(l)}\boldsymbol{W}^{(l)} + \boldsymbol{b}^{(l)}\right), \tag{58}$$

where $\boldsymbol{h}_{\mathrm{dense}}^{(l+1)}$ is the output of the Dense layer, $\boldsymbol{h}^{(l)}$ is the output of the previous layer l, a is an activation function, $\boldsymbol{W}^{(l)}$, $\boldsymbol{b}^{(l)}$ are the trainable weight matrix and bias vector of the layer respectively. It is noted that the layer formulation itself cannot hold any memory information. The memory of path-dependence in this architecture is derived from the input of the neural network. The input is the full strain tensor $\boldsymbol{\epsilon}_n$ at time step n and the full stress tensor $\boldsymbol{\sigma}_{n-1}$ at the previous time step $(n-1)$, both in Voigt notation. The output prediction of the network is the stress tensor $\boldsymbol{\sigma}_n$ at time step n. The network attempts to infer the path-dependent behavior by associating the previous stress state with the current one. The architecture consists of three Dense hidden layers (100 neurons each) with ReLU activation functions and the output Dense layer with a Linear activation function.

The second architecture is a recurrent network that learns the path-dependent behavior in the form of time series. The architecture utilizes the Gated Recurrent Unit (GRU) layer formulation, a recurrent architecture introduced in [51] — a variation of the popular Long Short Term Memory (LSTM) recurrent architecture [52]. The GRU cell controls memory information by utilizing three gates (an update gate, a reset gate, and a current memory gate). This architecture takes a time series of strain as the input with a history length of ℓ. The variable ℓ is a network hyperparameter that is fine-tuned for optimal results and it signifies the amount of information from the previous time steps that are taken into consideration to make a prediction for the current time step. Thus, a GRU network

Table 1
Summary of black-box neural network architectures used for control experiments.

Model	Description
$\mathcal{M}_{\text{stepDense}}$	**Dense** (100 neurons/ReLU) → **Dense** (100 neurons/ReLU) → **Dense** (100 neurons/ReLU) → Output **Dense** (Linear)
$\mathcal{M}_{\text{GRU}}$	**GRU** (32 units/tanh) → **GRU** (32 units/tanh) → **Dense** (100 neurons/ReLU) → **Dense** (100 neurons/ReLU) → Output **Dense** (Linear)
$\mathcal{M}_{\text{Conv1D}}$	**Conv1D** (32 filters/ReLU) → **Conv1D** (64 filters/ReLU) → **Conv1D** (128 filters/ReLU) → **Flatten** → **Dense** (100 neurons/ReLU) → **Dense** (100 neurons/ReLU) → Output **Dense** (Linear)

sample for the time step n has input the series of strain tensors $[\boldsymbol{\epsilon}_{n-\ell}, \ldots, \boldsymbol{\epsilon}_{n-1}, \boldsymbol{\epsilon}_n]$ and output the stress tensor for the current step $\boldsymbol{\sigma}_n$. The architecture used in this work consists of two GRU hidden layers (32 recurrent units each) with a ReLU activation function, followed by two Dense layers (100 neurons) with ReLU activations and a Dense output layer with a Linear activation function. The history variable was set to $\ell = 20$.

The last architecture employed in the control experiments learns the path-dependent information from time series by extracting features through a 1-D convolution filter. The convolution filter extracts higher-order features from time series of fixed length and has been used for time-series predictions [53] and audio processing [54]. The input of this architecture is the series of strain tensors $[\boldsymbol{\epsilon}_{n-\ell}, \ldots, \boldsymbol{\epsilon}_{n-1}, \boldsymbol{\epsilon}_n]$ and output the stress tensor for the current step in Voigt notation $\boldsymbol{\sigma}_n$. The 1D convolutional filter processes segments of the path-dependent time series data in a rolling window manner and has a length equal to ℓ. The architecture consists of three 1D convolution layers (32, 64, and 128 filters respectively) with ReLU activation functions. The output features of the last convolutional layer are flattened and then fed into two consecutive Dense layers (100 neurons) with ReLU activations, followed by a Dense output layer with a Linear activation function.

All the architectures were trained for 500 epochs with the Nadam optimizer with a batch size of 64. They were trained on different data sets to illustrate the comparisons with our elastoplasticity framework — the data sets are described in the context of the numerical experiments in Section 5.4. The hyperparameters of these architectures were fine-tuned through trial and error in an effort to provide optimal results and a fair comparison with our elastoplasticity framework to the best of our knowledge. The three black-box architectures are summarized in Table 1.

5. Numerical experiments

In this section, we report the results of the numerical experiments that we conducted to verify the implementation and evaluate the predictive capacity of the presented elastoplasticity ANN framework. For brevity, some background materials and simple verification exercises are placed in the Appendices. In Section 5.1, we demonstrate how the training of the hyperelastic energy functional approximator can benefit from the use of higher-order activation functions and higher-order Sobolev constraints. In Section 5.2, we demonstrate the training of the yield function level set neural networks and their approximation of the evolving yield functions. In Section 5.4, we compare the three recurrent architectures of Section 4 to our elastoplasticity framework as surrogate models for a polycrystal microstructure. Finally, in Section 5.5, we demonstrate the ability of our framework to integrate into a finite element simulation by fully replacing the elastic and plastic constitutive models with their data-driven counterparts.

5.1. Benchmark study 1: Higher-order Sobolev training of hyperelastic energy functional

In this numerical experiment, we compare the learning capacity of feed-forward neural networks trained on hyperelastic energy functional data. The training loss curves and the predictive capacity of the neural networks with different configurations of Multiply layers and different order norms as loss functions are compared. The generation of the data sets is discussed in Appendix A.

The neural network models in this work are trained on two energy functionals data sets for linear elasticity and non-linear elasticity (Eq. (59)) of 2500 sample points each. The points are sampled in a uniform grid of the

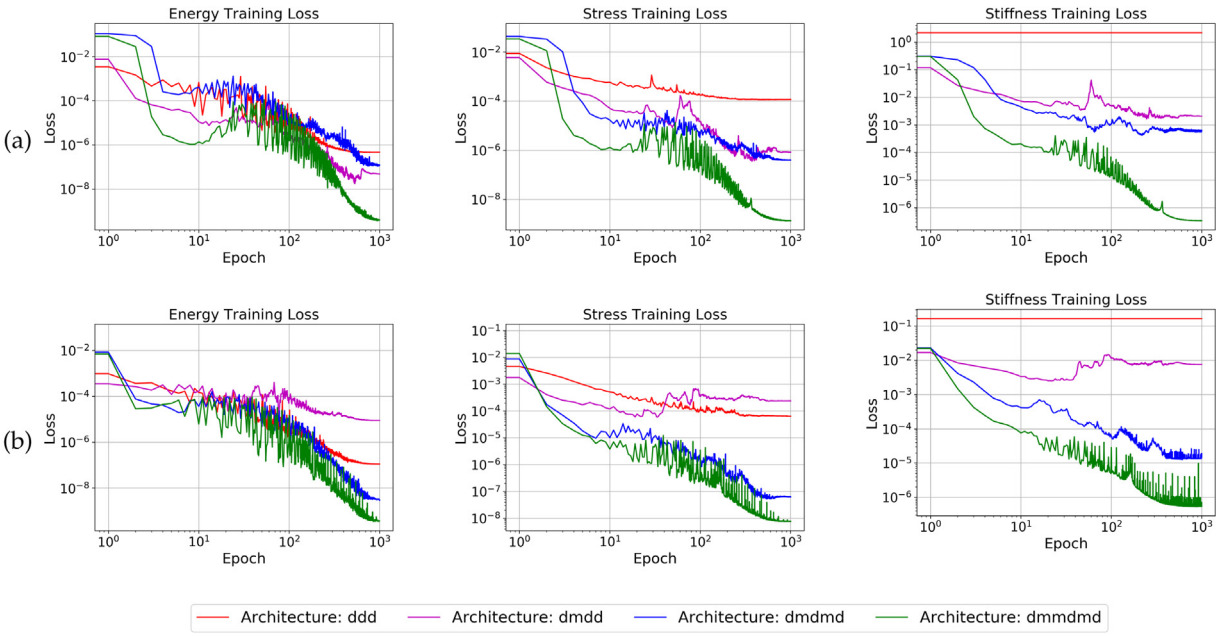


Fig. 7. Training loss comparison of feed-forward architectures with a progressively larger number of Multiply Layers with an H^2 training objective for (a) linear elasticity and (b) Modified Cam-Clay hyperelastic law [55]. As a higher degree of continuity is introduced in the network architecture, the stiffness accuracy prediction increases — more control is allowed for the H^2 terms of the training objective.

strain invariant space $(\epsilon_v^{\mathrm{e}}, \epsilon_s^{\mathrm{e}})$. In the first part of this numerical experiment, we investigate the capability of the different feed-forward architectures to fulfill the higher-order Sobolev constraints. The layers' kernel weight matrix was initialized with a Glorot uniform distribution and the bias vector with a zero distribution — the repeatability of the training process is demonstrated in Fig. 28 of Appendix D. Other than the number of intermediate Multiply layers, all the other hyperparameters are identical among all the architectures tested. The training objective is formulated according to Eq (3). All the models were trained for 1000 epochs with a batch size of 32 using the Nadam optimizer [56], set with default values in the Keras library.

The performance of different neural network architectures (see Fig. 1) is compared and the results are shown in Fig. 7. Architecture dmmdmd consistently exhibits the best learning capacity and achieves the lowest loss in energy, stress and tangential stiffness calculation. As expected, Architecture ddd fails in predicting the tangential stiffness due to the insufficient degree of continuity of the activation functions.

In the second numerical experiment, we investigate the predictive accuracy of the dmmdmd architecture (as shown in Fig. 1) trained via L^2, H^1, and H^2 norms. In all three cases, the neural network architecture and the training hyperparameters are identical to the ones used in the first numerical experiment. The results of these three training experiments can be seen in Fig. 8. The predictive capability of the model increases when higher-order Sobolev training is utilized with the best overall scores procured for H^2 norm-based training. Czarnecki et al. [42] had observed that constraining the H^1 terms in the loss function improves the function value prediction accuracy. Our results indicate that by constraining the H^2 terms, we are improving the prediction of the function values along with the first-order and the second-order derivatives of the function (see Fig. 9).

5.2. Benchmark study 2: Training of yield function as a level set

In this numerical experiment, we demonstrate how to train the neural network to generate a yield function whose evolution is driven by signed distance function data interpreted from experiments. The yield function neural networks have a feed-forward architecture of a hidden Dense layer (100 neurons/ReLU), followed by two Multiply layers, then another hidden Dense layer (100 neurons/ReLU) and an output Dense layer (Linear). The layers' kernel weight matrix was initialized with a Glorot uniform distribution and the bias vector with a zero distribution — the

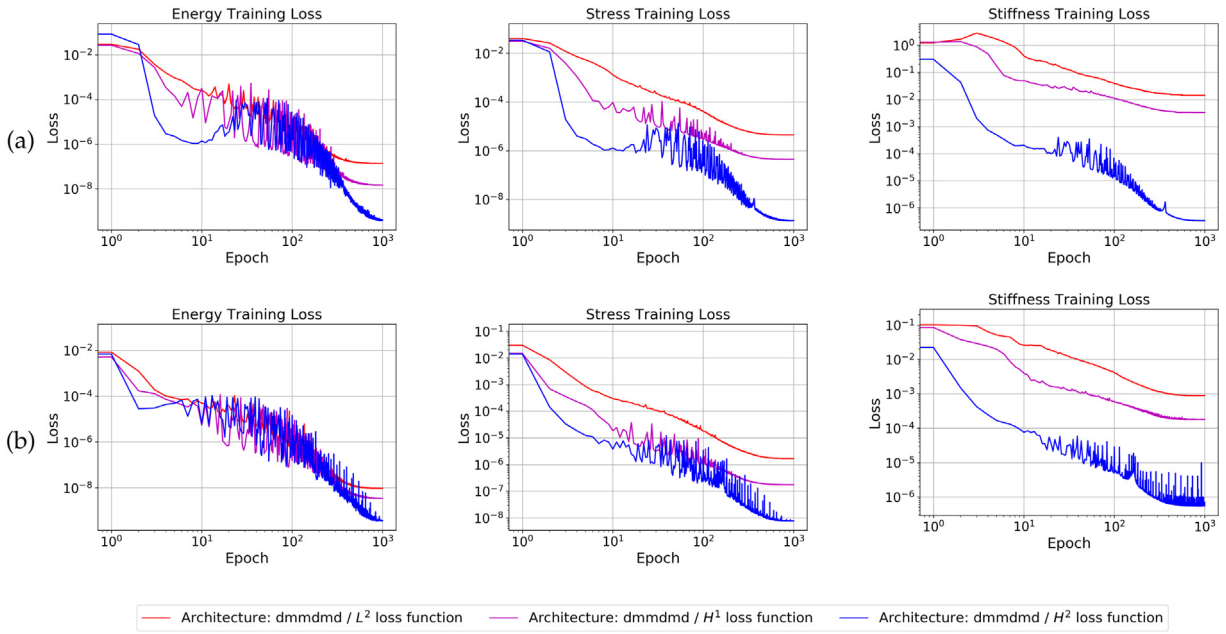


Fig. 8. Training loss comparison for L^2, H^1, and H^2 training objectives of an architecture with three Multiply layers (dmmdmd) for (a) linear elasticity and (b) Modified Cam-Clay hyperelastic law [55]. The H^2 training objective procures more accurate results than the L^2 and H^1 objectives for all of the energy, stress, and stiffness fields.

repeatability of the training process is demonstrated in Fig. 28 of Appendix D. All the models were trained for 2000 epochs with a batch size of 128 using the Nadam optimizer, set with default values. The neural networks were trained on a data set of J2 plasticity as well as data sets for 4 different polycrystal RVEs as described in Appendix B. The training loss curves for this experiment with an L^2 training objective are shown in Fig. 10.

The ability to capture a yield surface directly from the data becomes crucial in materials such as the polycrystal microstructures — where complex constitutive responses may manifest from spatial heterogeneity and grain boundary interactions. In Fig. 11, it is shown that a polycrystal RVE of the same size with different crystal orientations can have distinctive initial yield surfaces. Anticipating the geometry of the yield surface in the stress space and then handcrafting it with mathematical expressions would be a great undertaking and possibly futile — a change in the crystal properties would require deducing a geometric shape design from scratch.

This numerical test indicates that the proposed framework may automate the discovery of new yield surfaces while bypassing the time-consuming hand-crafting process. This framework may also be extended to capture the plastic behavior of anisotropic materials if the six-dimensional stress space is used. This will be considered in future work by expanding the stress invariant input space to include orientations and possibly more descriptive plastic internal variables that are derived from the topology of the microstructures.

5.2.1. Smoothed approximation of the non-smooth yield surfaces

Another useful feature of the proposed machine learning approach is that the Sobolev training can be used to generate a smoothed approximation of a multi-yield-surface system (e.g. Mohr–Coulomb, Tresca, crystal plasticity with multiple slip systems) on the π-plane. Classical non-smooth and multi-yield surface models often lead to sharp tips and corners of the yield surface that makes the stress gradient of the yield function bifurcated. This is not only an issue for stability but also requires specialized algorithmic designs for the return mapping algorithm to function (cf. [57]). As a result, there have been decades of efforts to hand-derive implicit functions that are smoothed approximations of well-known multi-yield surface models [58,59]). As shown in Fig. 11, the proposed Sobolev framework may automate this time-consuming process by simply using a combination of data points, activation functions, and loss functions to regularize the non-smooth yield surface. The resultant smoothed yield surface does not only avoid the bifurcated stress gradient at the corners but also enables us to use the standard

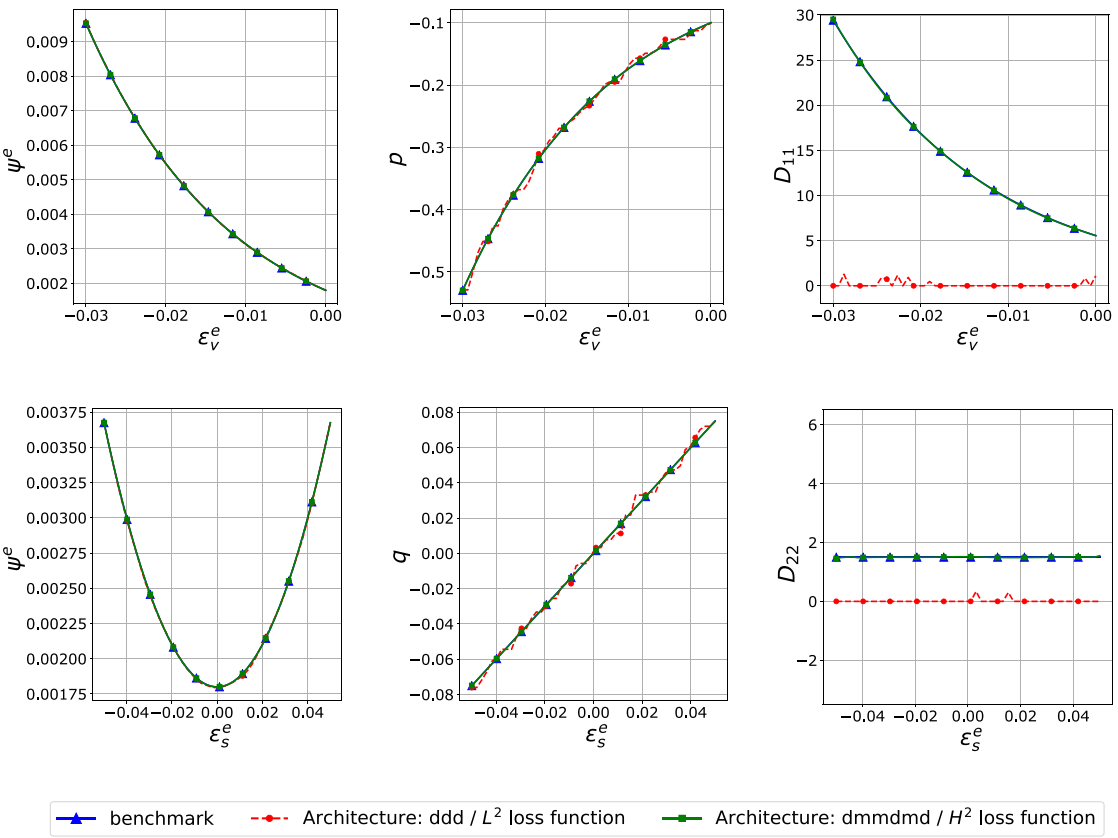


Fig. 9. Comparison of the predictions of an L^2 trained ddd network with and an H^2 trained dmmdmd network for the energy functional, stress, and stiffness of the Modified Cam-Clay hyperelastic law [55]. The ddd architecture (piece-wise linear activation functions) can only predict local second-order derivatives (D_{11}, D_{22}) to be equal to 0. The dmmdmd architecture, modified with Multiply layers, can capture these higher-order derivatives. The stress and stiffness are in MPa.

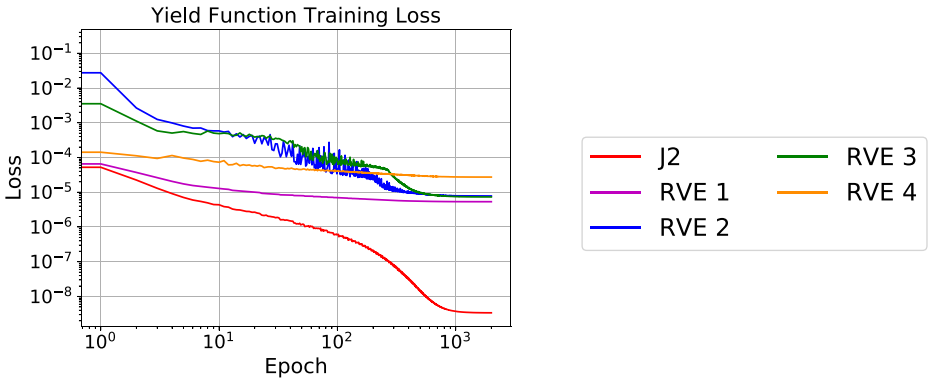


Fig. 10. Training loss curves for the J2 plasticity and 4 different polycrystal RVEs' yield function level sets.

return mapping algorithm for implicit stress integrations without requiring any additional numerical treatment to handle the tips and corners of the yield surface.

Furthermore, the ML-derived evolution of the yield function is also capable of replicating complex hardening mechanisms not known *a priori*. In Fig. 12, we demonstrate how the neural network can predict a hardening mechanism that has not yet been discovered in the literature. In particular, this yield surface does not only change

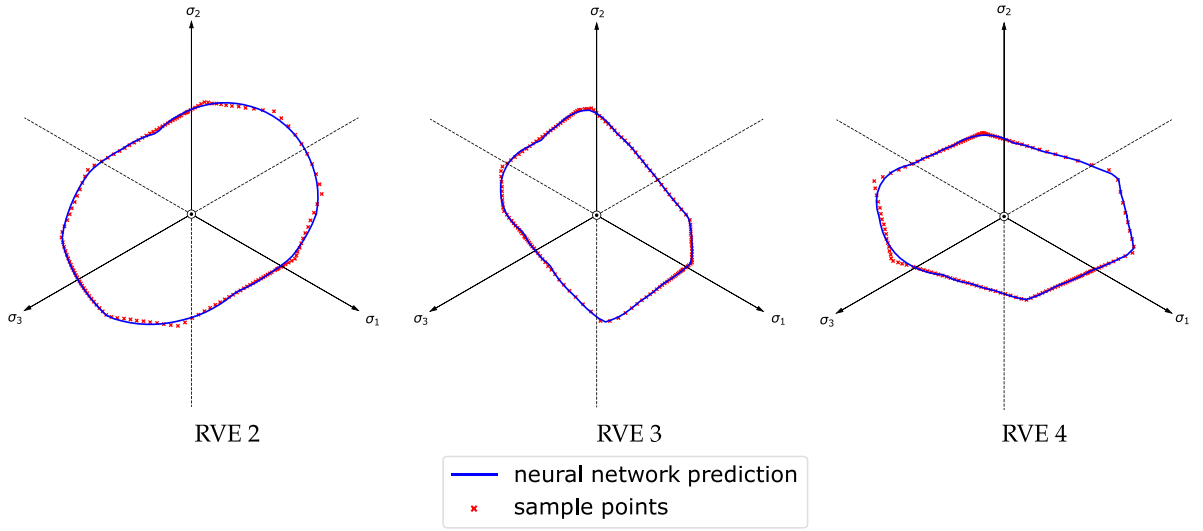


Fig. 11. Yield surface neural network predictions for three polycrystal RVEs with different crystal orientations.

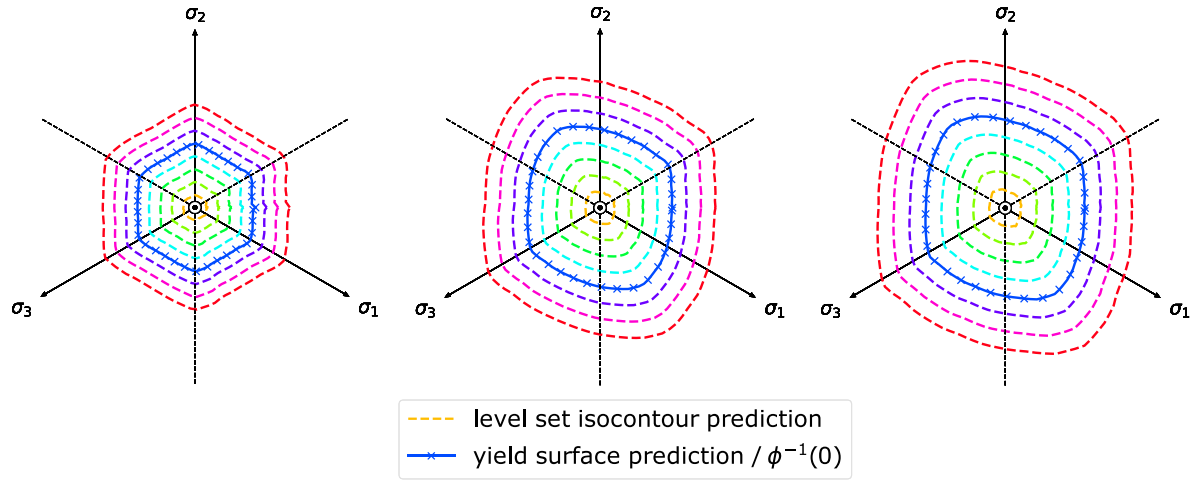


Fig. 12. Predicted signed distance function isocontours for three evolving yield surfaces of RVE 1 of a specimen undergoing increasing axial compression (left to right).

in size upon yielding but also **deforms** on the π-plane. Anticipating this mechanism and then deducing the corresponding mathematical expression to handcraft a hardening law is not trivial. Our framework can interpret experimental data and deduce the optimal shapes and forms of a yield surface that evolves with the strain history without the aforementioned burdens. This is not only important for making more accurate predictions but also enables us to derive more precise plasticity models tailored to specific specimens or data sets beyond parameter calibrations.

5.3. Benchmark study 3: Yield function training with higher-order constraints

In this numerical experiment, we demonstrate how to use the proposed training framework to enforce thermodynamic constraints and other desirable properties of the yield function via geometrical interpretation in the stress space. Our experimental data obtained from direct numerical simulations have been pre-processed into signed distance functions that evolve according to the accumulated plastic strain. The constraints we are interested to enforce here are the unit gradient $|\nabla\phi| = 1$ and the non-negativity of the plastic dissipation.

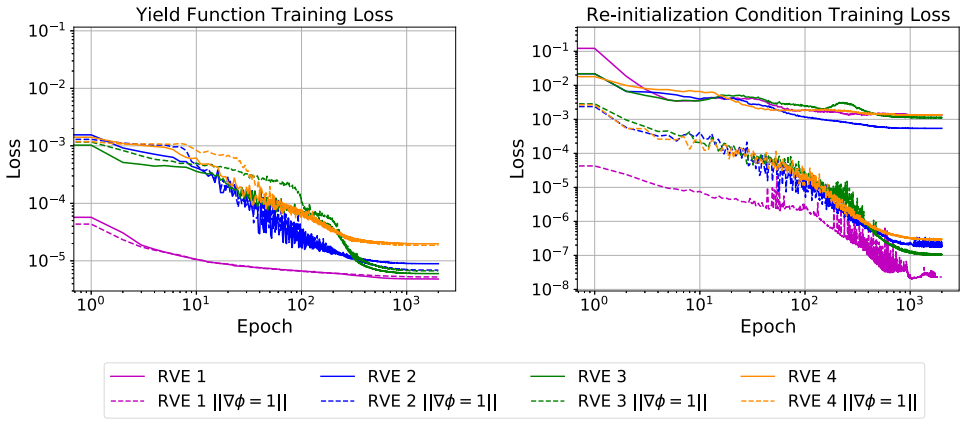


Fig. 13. Training loss (LEFT) and re-initialization condition loss (RIGHT) comparison for four polycrystal RVEs' yield functions with and without enforcing the Eikonal equation unit gradient constraint.

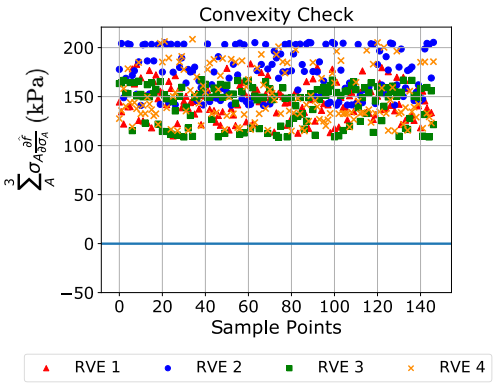


Fig. 14. Convexity check for randomly sampled stress points from the polycrystal RVE data set.

5.3.1. Unit gradient constraint

The unit gradient constraint has been directly applied as an additional term in the loss function and the results of the training for 4 different RVEs with and without unit gradient $|\nabla\phi| = 1$ constraints are compared in Fig. 13. These two sets of yield function training show that the additional unit gradient constraint does not affect the learning capacity and the accuracy of the resultant yield function. Meanwhile, the training that explicitly enforces the unit gradient property is also at least 3 orders more successful at fulfilling the unit gradient constraint than the counterpart that does not do so.

5.3.2. Convexity of the yield function

To enforce the non-negative plastic dissipation, we enforce the convexity of the yield function by enforcing an inequality constraint via Eq. (46). The additional term in the loss function only activates when the thermodynamic inequality is violated. By increasing the value of the loss function, it penalizes predictions that violate the convexity conditions during training. During the training phase of the numerical experiments presented in this paper, the penalty term did not activate. Nevertheless, the penalty term in the loss function is still employed as a safeguard to prevent the violation of the thermodynamic constraints.

We expect that this safeguard will be helpful in the future work where this framework will be extended to handle experimental data with noise and for anisotropic materials of which the visual inspection of the convexity in the principal stress or π-plane is no longer sufficient. Verification of the convexity is performed and the results are shown in Fig. 14 where material states were randomly sampled from the polycrystal RVE database to test whether Inequality (46) is violated.

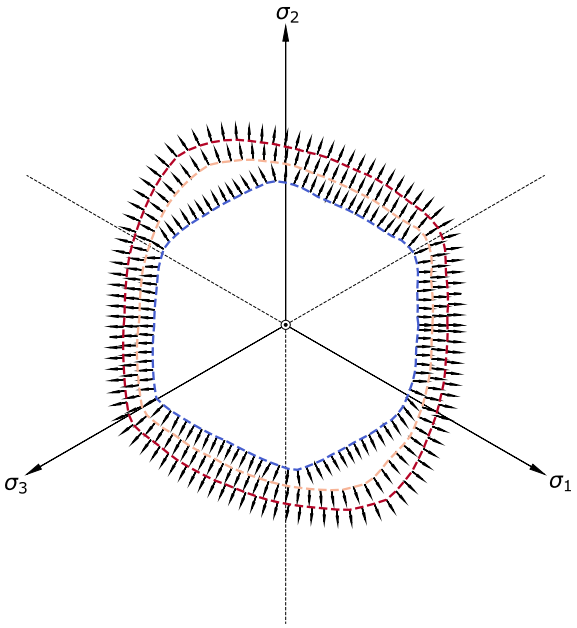


Fig. 15. The ML-predicted plastic flow of the polycrystal RVE for increasing accumulated plastic strain ($\bar{\epsilon}_p$ equal to 0.01, 0.03, and 0.08).

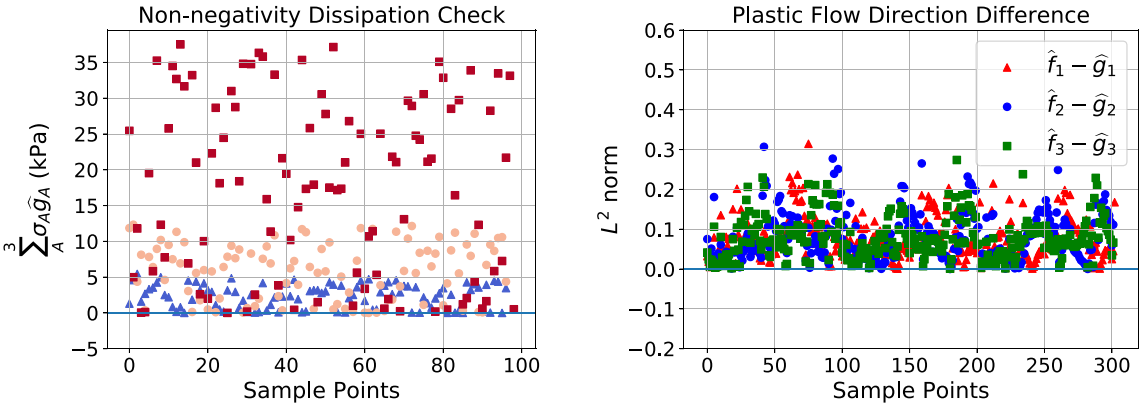


Fig. 16. (a) The plastic flow rule is checked for non-negative dissipation (the points correspond to the different accumulated plastic strain levels of Fig. 15). (b) L^2 norm comparison for the predicted plastic flow direction between yield function neural network ($\widehat{f_A}$) and plastic flow neural network ($\widehat{g_A}$).

5.3.3. Non-negative plastic dissipation for non-associative plasticity

In the case where the plastic flow is non-associative, we enforce the additional rule expressed in Eq. (52) to ensure the non-negativity of the plastic dissipation for isotropic materials. The remaining training parameters of the network are identical to the ones used for the yield function learning in Section 5.2. To verify that convexity is preserved, we perform a random check by sampling stress points on the π-plane and compute the plastic dissipation (see Fig. 16(a)). In this case, having the safeguard incorporated into the loss function ensures that plastic dissipation is either zero or positive. Furthermore, there are noticeable differences between the optimal plastic flow and the stress gradient of the yield function, which indicates the need to introduce non-associative plastic flow in this prediction task (see Fig. 16(b)).

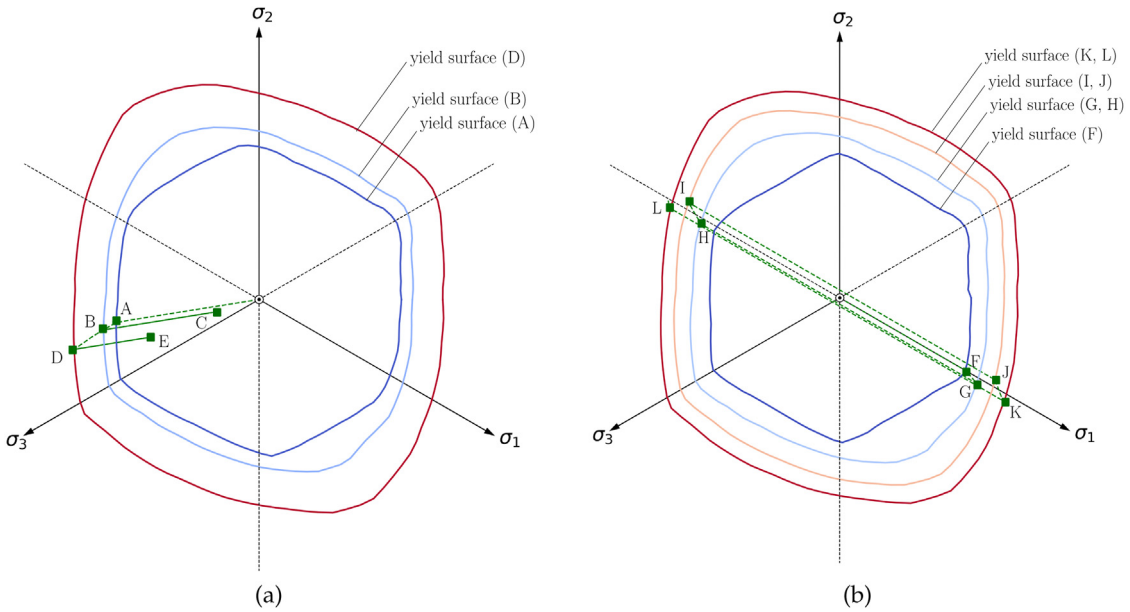


Fig. 17. Stress path in the π-plane for (a) a loading–unloading pattern and (b) a cyclic loading path. The yield surface neural network predicts the consecutive yield surfaces for different levels of hardening. (a) The points A, B, C, D, and E correspond to the strain–stress curve of Fig. 19. (b) The points F, G, H, I, J, K, and L correspond to the strain–stress curve of Fig. 20.

5.4. Application 1: Surrogate model comparisons for polycrystal RVEs

In this example, we compare the performance of the elastoplastic model with Hamilton–Jacobi hardening (introduced in Section 2) with the trained black-box models obtained from the other commonly used recurrent neural network architectures introduced in Section 4. Our goal is to create a model to predict the upscaled elasto-plastic responses of a polycrystal RVE undertaking unseen loading paths. The data set generation for the neural network approximator $\widehat{f}$ is described in Appendix B.

To conduct a fair and systematic comparison, we have trained, tested, and compared with the recurrent models with different amounts of data generated from loading paths of various types of complexity. It is noted that the database used for the training of $\hat{f}$ will not be extended further than the 140 cases of monolithic loading cases described in Appendix B.

5.4.1. Predicting cyclic responses from monotonic data

Initially, we train all the recurrent architectures with the 140 cases of monolithic loading paths (with 200 to 400 deformation states per case), sampled radially from the π-plane. The same set of data is also used to train the approximator $\hat{f}$. All models are expected to perform adequately well in blind predictions for monolithic testing cases as they have been trained for these simple patterns as shown in Fig. 18(a). However, the black-box recurrent networks fail to recover even a single unloading and reloading path, as seen in Fig. 18(b) whereas our proposed interpretable model is able to recover loading and unloading patterns quite well even though it was only trained on monolithic loading paths. This success is due to the existence of a yield surface that enables our proposed model to detect unloading and, hence, trigger the right elastic unloading responses, even in the absence of unloading data (see Fig. 17). While the lack of unloading data may still affect the accuracy of the ML-generated hardening mechanism, the prediction of the proposed ML plasticity model is still more robust than the black-box counterpart.

5.4.2. Predicting cyclic responses from cyclic data

In the second numerical experiment, we increase the complexity of the database that the recurrent neural networks are trained on. Following the same loading path angles as a basis on the π-plane, we generate cases that now include complex unloading and reloading paths. We, thus, allow the recurrent architectures to be exposed to previously

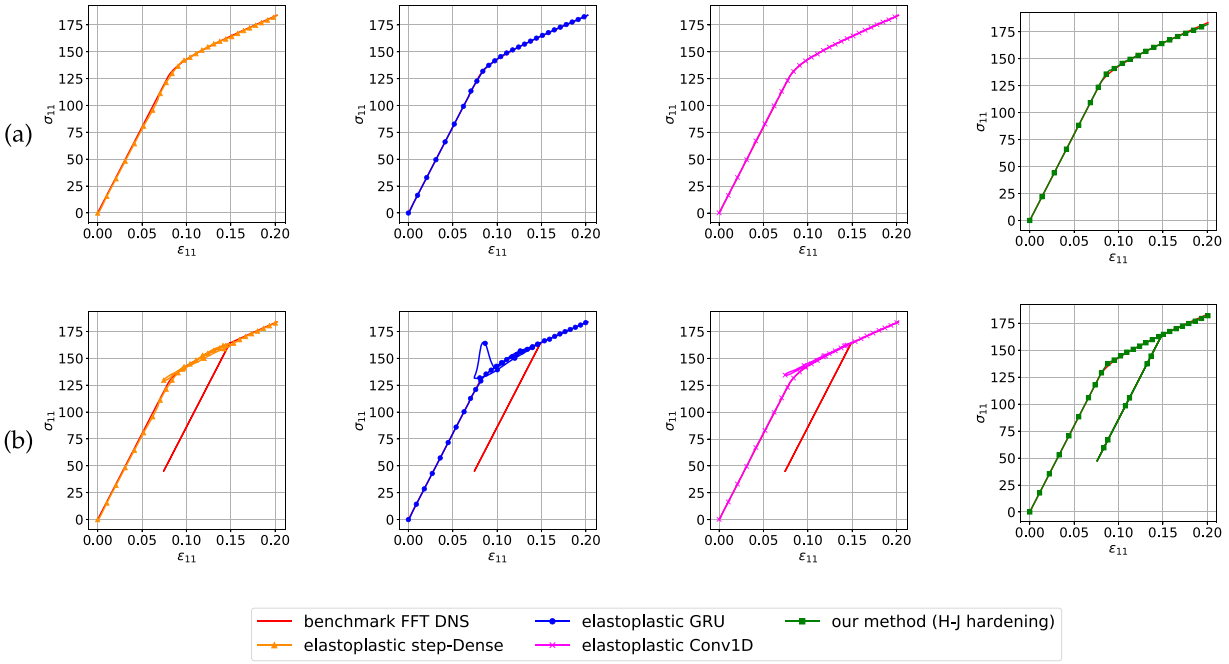


Fig. 18. Comparison of black-box neural network architectures trained on monolithic data with our Hamilton–Jacobi hardening elastoplastic framework (introduced in Section 2). The black-box models can capture the monolithic loading path (a) but cannot capture any unloading paths (b). Our framework can capture both even though it has only seen monolithic data. The stress measure is in kPa.

missing elastic unloading paths. We randomly assign the unloading and loading paths for every loading direction. At every direction, we randomly assign from 1 to 3 unloading and reloading paths with the unloading target strain also randomly chosen each time. Using this method, we double the number of sampling points of the initial cases by adding random unloading and reloading patterns to retrain the recurrent architectures. The performances of all these models are again compared. The results for three testing cases with cyclic loadings are shown in Fig. 19.

In the third comparison experiment, we examine the accuracy and robustness of the predictions of constitutive responses for the polycrystal specimen subjected to cyclic loading and unloading paths that span both tensile and compressive directions. The results for the cyclic testing can be seen in Fig. 20. As expected, the black-box models – even with an extended training data set – fail to capture the cyclic behaviors. The proposed ML elasto-plasticity model — even when only trained with monolithic data, exhibits very good accuracy and robustness of predictions on the cyclic behaviors for the polycrystal plasticity.

5.5. Application 2: Finite element simulations with machine learning-derived polycrystal plasticity models

The return mapping algorithm of our elastoplastic neural network framework, described in Section 3, is implemented in a series of benchmark finite element simulations. The goal of these computational examples is to demonstrate the framework's ability to be integrated into multi-scale simulations. By predicting the homogenized elastoplastic response of the microstructure, we can enable offline hierarchical multi-scale predictions that are much faster than an FE^2 approach without compromising the accuracy and robustness.

We perform the finite element quasi-static simulation of macroscopic monotonic uniaxial displacement of a bar depicted in Fig. 21. The domain is symmetric along the horizontal and vertical axes and the elasticity and plasticity model used are isotropic, thus, we are modeling one quarter of the domain to predict the symmetric behavior. The domain is meshed with 3800 triangular elements with an average side length of 6.75×10^{-4} m. The displacement u is applied at the boundaries as shown in Fig. 21 in increments of $\Delta u = 5 \times 10^{-5}$ m. The microscopic elastoplastic behavior of every material point in the mesh is predicted by an elastic energy functional neural network and a yield function signed distance function neural network, integrated by the return mapping algorithm of Section 3.

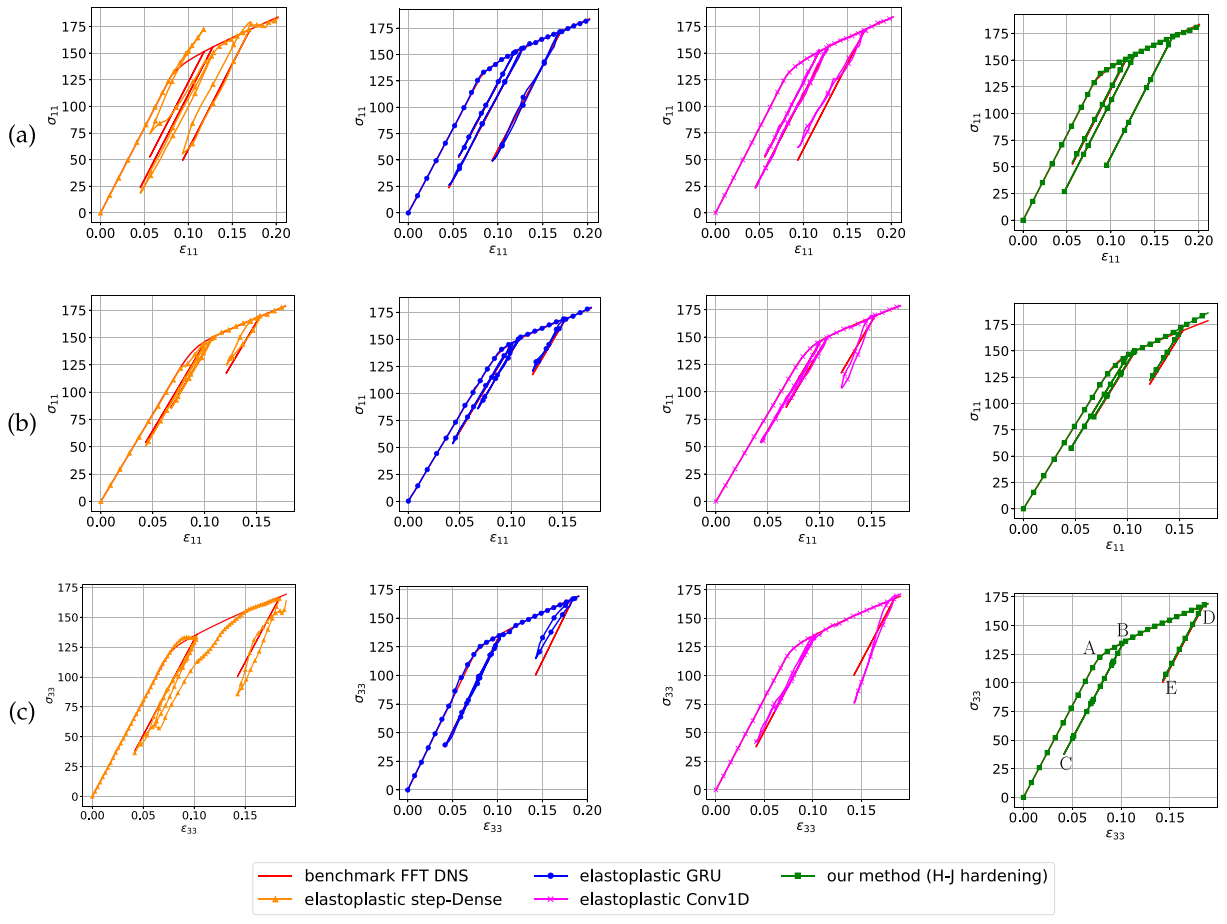


Fig. 19. Comparison of black-box neural network architectures trained on random loading–unloading data with our Hamilton–Jacobi hardening elastoplastic framework. Three different cases of loading–unloading are demonstrated (a, b, and c). The black-box models can capture loading–unloading behaviors better than the monolithic data trained ones (Fig. 18) but still may show difficulty capturing some unseen unloading paths. Our framework appears to be more robust in loading–unloading path predictions — even though it is only trained on monolithic data. The stress measure is in kPa.

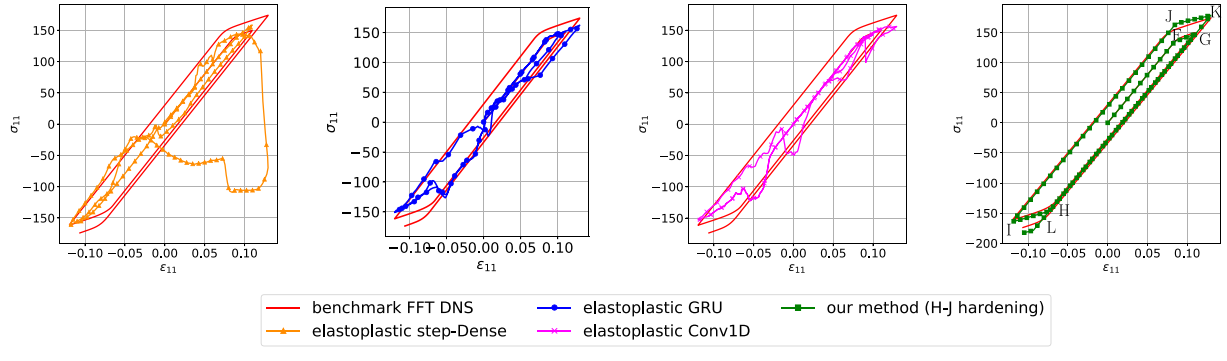


Fig. 20. Comparison of black-box neural network architectures trained on random loading–unloading data with our Hamilton–Jacobi hardening elastoplastic framework for a cyclic loading path. The stress measure is in kPa.

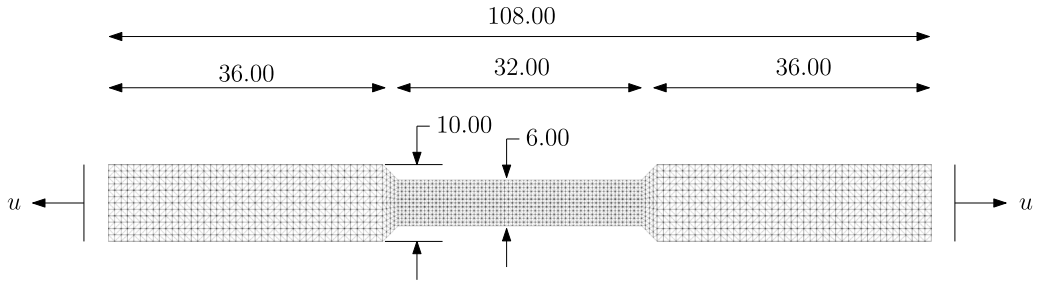


Fig. 21. Macroscopic structure and boundary conditions used in the finite element simulations. The domain is symmetric along the horizontal and vertical axes so only one quarter of the domain is modeled. The units are in mm.

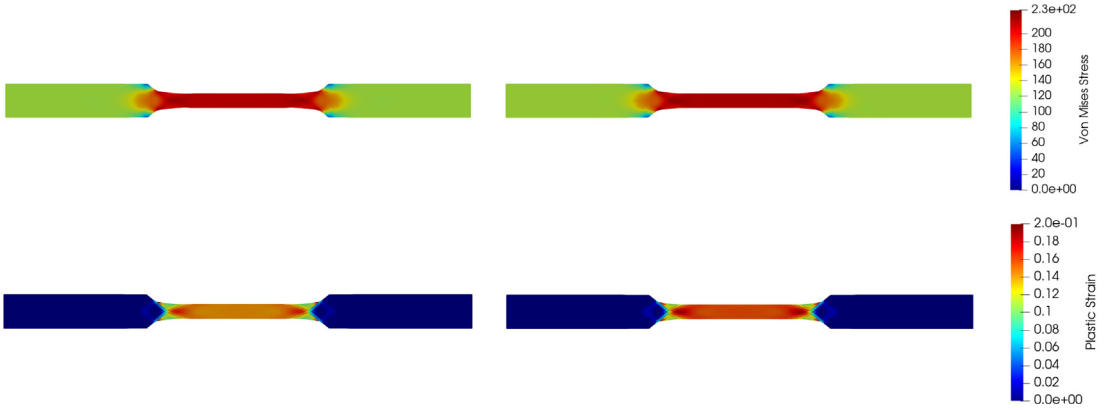


Fig. 22. Von Mises stress (TOP) and accumulated plastic strain (BOTTOM) for the benchmark J2 plasticity (LEFT) simulation and neural network J2 yield function (RIGHT) FEM simulations. The stress measure is in kPa.

As the first numerical verification exercise, we combine a quadratic energy functional of linear elasticity and a J2 plasticity yield function with isotropic hardening. The neural networks and their training for the elastic response have been described in Section 5.1 and, for the plastic response, in Section 5.2 and Appendix D. The goal displacement for the uniaxial loading simulation is $u_{\text{goal}} = 5.5 \times 10^{-3}$ m. The results at the goal displacement for the benchmark solution and our elastoplastic Hamilton–Jacobi hardening framework are demonstrated in Fig. 22 and appear to be in close agreement.

In the second numerical experiment, we simulate the behavior under uniaxial loading of the domain in which the material points represent a polycrystalline microstructure. The elastoplastic framework in this simulation consists of a quadratic hyperelastic energy functional neural network and a polycrystal yield function the training procedures of which are described in Sections 5.1 and 5.2 respectively. Both networks were trained on FFT simulation data as described in Appendix B to predict the homogenized elastoplastic behavior of the polycrystal.

Previously, the multiscale polycrystal constitutive behavior has been captured through a coupling of the FFT and FEM method (e.g. [60,61]). However, the efficiency of these methods depends on the heterogeneity of the polycrystal as it can affect the computational cost of the simulations for a large number of crystals and the stability when there are sharp material property differences. In the current work, there is no need for online FFT simulations to be run in parallel with the FEM simulations. The neural network training database for the elasticity and the plasticity are built separately offline for a discrete number of FFT simulations and the trained networks will be interpolating the behaviors and making blind predictions during the FEM simulation. The results of the simulation for the Von Mises stress and the accumulated plastic strain for the simulation at the displacement goal of $u_{\text{goal}} = 6.5 \times 10^{-3}$ m are demonstrated in Fig. 23. The stress curves and stress paths on the π-plane for two points of the domain are also demonstrated in Fig. 24.

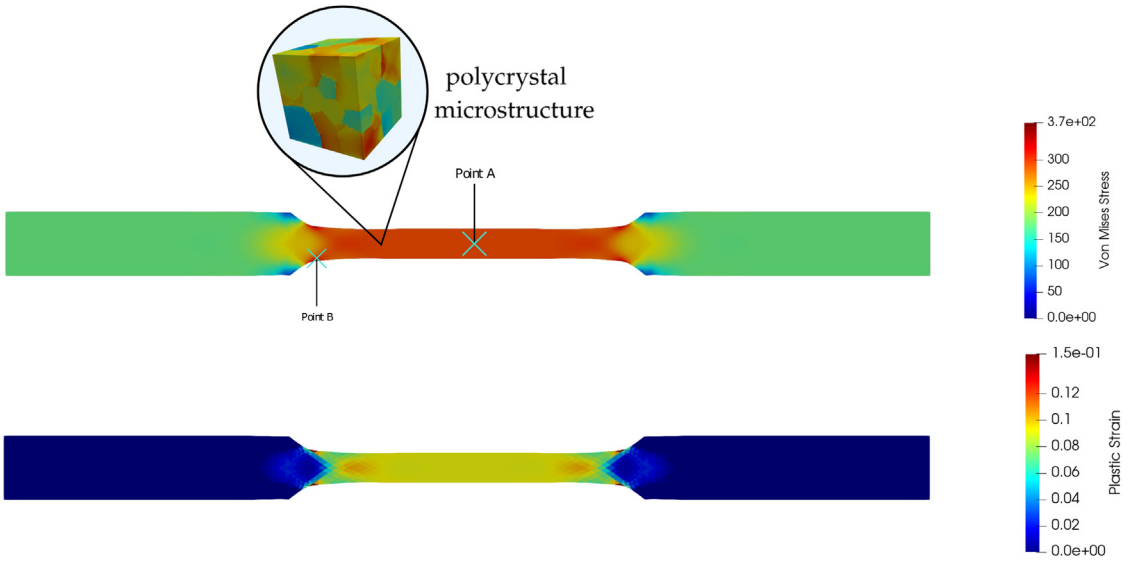


Fig. 23. Von Mises stress (TOP) and accumulated plastic strain (BOTTOM) for the neural network polycrystal yield function FEM simulations. The stress measure is in kPa.

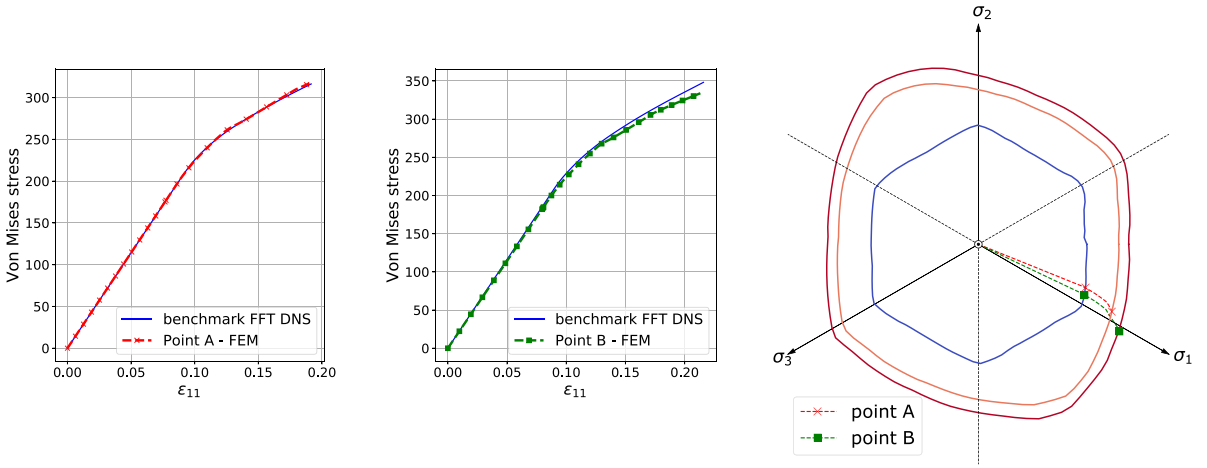


Fig. 24. Von Mises stress curves (LEFT) and stress paths on the π-plane (RIGHT) for Points A and B of the domain of Fig. 23. The stress measure is in kPa.

6. Conclusions

The history of plasticity theory is influenced by the geometrical interpretations of mechanics concepts in different parametric spaces [8,9,46,62]. Fromulating materials models in an Euclidean space spanned by invariants or eigenvalues as orthogonal bases had helped us interpreting the yielding, the subsequent hardening and softening and the plastic flow mechanisms. However, new plastic deformation mechanisms may still take decades to be discovered and adopted into constitutive laws by the mechanics community. In this work, our contributions are two-fold. First, we leverage the geometrical interpretation of plasticity theory to establish a connection between the elastoplasticity and the level set theories. Second, we introduce a deep machine learning algorithm designed to generate plasticity models with sufficient smoothness. By using higher-order training to regularize the continuity and smoothness of the elastic energy functional, the yield function, the flow rules, and the hardening mechanisms, we create a framework that is simple and easy to interpret and yet sufficiently sophisticated to formulate new yield functions, plastic flow, and hardening mechanisms yet to be discovered in the literature. Furthermore, thermodynamic constraints can

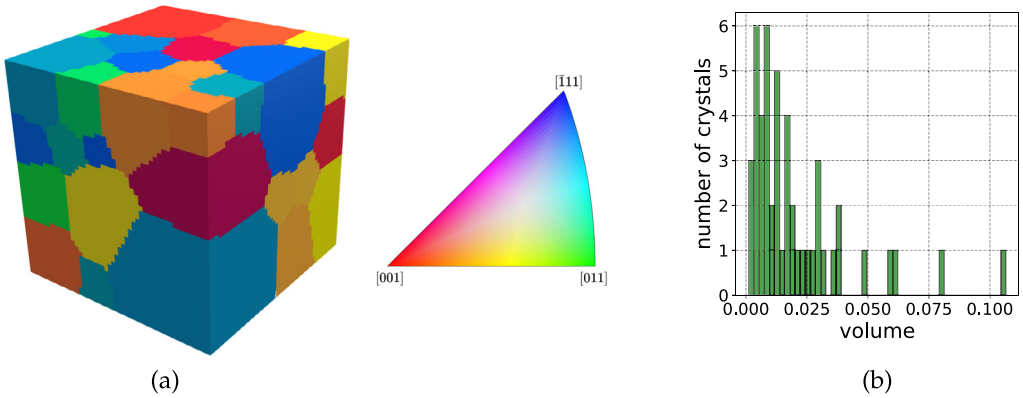


Fig. 25. (a) Crystal orientations and (b) crystal volume distribution of the polycrystal RVE used for the elastoplasticity database generation.

be easily checked and introduced, as the machine learning generated models are now geometrically interpretable. Finally, the most significant part of this research is that it provides a generalized framework where the yield function may form in any arbitrary shape and evolve in any way that optimizes the quality of the predictions. Our numerical experiments provides evidence that the level set framework may manifest many classical plasticity models when given the corresponding data as well as formulating new yield surfaces and hardening laws with geometries that are difficult to anticipate and hand-craft. Our findings indicate that our proposed framework may yield more accurate, robust, and interpretable predictions than those forecasted by the black-box neural networks.

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.

Data availability

The data that support the findings of this study are available from the corresponding author upon request.

Acknowledgments

The authors would like to thank Dr. Ran Ma for providing the implementation of the polycrystal microstructure generation, the FFT solver, and the information for Fig. 25. The authors are supported by the NSF CAREER grant from Mechanics of Materials and Structures program at National Science Foundation, United States of America under grant contracts CMMI-1846875 and OAC-1940203, the Dynamic Materials and Interactions Program from the Air Force Office of Scientific Research, United States of America under grant contracts FA9550-17-1-0169 and FA9550-19-1-0318. These supports are gratefully acknowledged. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the sponsors, including the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Appendix A. Data generation for the hyperelasticity benchmark

In this work, the numerical experiments (Section 5) are performed on synthetic data sets generated for two small strain hyperelastic laws. One of them is isotropic linear elasticity. The second is a small-strain hyperelastic law designed for the Modified Cam-Clay plasticity model [55,63,64]. The hyperelastic energy functional allows full coupling between the elastic volumetric and deviatoric responses and is described as:

$$\psi\left(\epsilon_v^{\mathrm{e}}, \epsilon_s^{\mathrm{e}}\right) = -p_0 c_r \exp\left(\frac{\epsilon_{v0} - \epsilon_v^{\mathrm{e}}}{\xi}\right) - \frac{3}{2} c_\mu p_0 \exp\left(\frac{\epsilon_{v0} - \epsilon_v^{\mathrm{e}}}{\xi}\right) \left(\epsilon_s^{\mathrm{e}}\right)^2, \tag{59}$$

where ϵ_{v0} is the initial volumetric strain, p_0 is the initial mean pressure when $\epsilon_v = \epsilon_{v0}$, $\xi > 0$ is the elastic compressibility index, and $c_\mu > 0$ is a constant. The hyperelastic energy functional is designed to describe an elastic compression law where the equivalent elastic bulk modulus and the equivalent shear modulus vary linearly with $-p$, while the mean pressure p varies exponentially with the change of the volumetric strain $\Delta\epsilon_v = \epsilon_{v0} - \epsilon_v$. The specifics and the utility of this hyperelastic law is outside the scope of this current work and will be omitted. The numerical parameters of this model were chosen as $\epsilon_{v0} = 0$, $p_0 = -100$ kPa, $c_\mu = 5.4$, and $\xi = 0.018$. Taking the partial derivatives of the energy functional with respect to the strain invariants, the stress invariants are derived as:

$$p = \frac{\partial \psi}{\partial \epsilon_v^{\mathrm{e}}} = p_0 \left(1 + \frac{3c_\mu}{2\xi} \left(\epsilon_s^{\mathrm{e}}\right)^2\right) \exp\left(\frac{\epsilon_{v0} - \epsilon_v^{\mathrm{e}}}{\xi}\right), \tag{60}$$

$$q = \frac{\partial \psi}{\partial \epsilon_s^{\mathrm{e}}} = -3c_\mu p_0 \exp\left(\frac{\epsilon_{v0} - \epsilon_v^{\mathrm{e}}}{\xi}\right) \epsilon_s^{\mathrm{e}}. \tag{61}$$

The components of the symmetric stiffness Hessian matrix $\boldsymbol{D}^e$ are derived by taking the second-order partial derivative of the energy functional with respect to the two strain invariants:

$$\begin{aligned} D_{11}^{\mathrm{e}} &= \frac{\partial^2 \psi}{\partial \epsilon_v^{\mathrm{e}\,2}} = -\frac{p_0}{c_r} \left(1 + \frac{3c_\mu}{2c_r} \left(\epsilon_s^{\mathrm{e}}\right)^2\right) \exp\left(\frac{\epsilon_{v0} - \epsilon_v^{\mathrm{e}}}{c_r}\right), \\ D_{22}^{\mathrm{e}} &= \frac{\partial^2 \psi}{\partial \epsilon_s^{\mathrm{e}\,2}} = -3c_\mu p_0 \exp\left(\frac{\epsilon_{v0} - \epsilon_v^{\mathrm{e}}}{c_r}\right), \\ D_{12}^{\mathrm{e}} &= D_{21}^{\mathrm{e}} = \frac{\partial^2 \psi}{\partial \epsilon_v^{\mathrm{e}} \partial \epsilon_s^{\mathrm{e}}} = \frac{3p_0 c_\mu \epsilon_s^{\mathrm{e}}}{c_r} \exp\left(\frac{\epsilon_{v0} - \epsilon_v^{\mathrm{e}}}{c_r}\right). \end{aligned} \tag{62}$$

Appendix B. Data generation for polycrystal yield function

Here we provide a brief account on the direct numerical simulations that generates the data set for the ML-generated plasticity model in Section 5. The numerical specimen is a polycrystal assembly consisting of 49 face centered cubic crystal grains. The crystal orientations are randomly generated using the open source software MTEX [65]. The crystal orientation distribution is demonstrated in Fig. 25 along with the crystal volume distribution. Directed numerical simulations are performed on this numerical specimen by solving the Lippman–Schwinger equation using the FFT spectral method with periodic boundary condition [66,67]. The resultant stress field and plastic deformation are homogenized and these homogenized responses constitute the material database used for training of the machine learning plasticity model.

The elasticity model of the polycrystals is linear elasticity with a Young's Modulus of $E = 2.0799$ MPa and a Poisson ratio of $\nu = 0.3$. The material's plastic behavior was calculated using the ultimate algorithm for crystal plasticity [68]. The model has 12 linearly independent slip systems with a yield stress of 100 kPa and a hardening modulus of 100 kPa. An FFT elastoplastic simulation is performed radially for each of 140 different Lode's angles spanning the π-plane.

Each data point generated by the FFT simulations is stored in a cylindrical coordinate system with positions specified by a radius ρ, an angle θ, and an accumulated plastic strain $\bar{\epsilon}_p$. For every generated sample point (ρ_o, θ_o) on the π-plane, we construct 14 signed distance function training points using a signed distance function, distributed uniformly on the radial direction with a distance range of $\pm\rho_o$ from point (ρ_o, θ_o). The size of the constructed signed distance function corresponds to parameters $L_{\text{levels}} = 15$ and $\zeta = 2$ in Algorithm 2.

After generating the points of the signed distance function, every point has a corresponding output value equal to the signed distance function $\phi(\rho_o, \theta_o, \bar{\epsilon}_{p,o})$ for that point. In this way, all the signed distance function points on an isocontour will have the same output value. This has proven to be an obstacle in the back-propagation during the neural network training — many input combinations correspond to the same output value. To increase the variation of the output values of each sample during training, we introduce a helper transformation function $\zeta(\rho, \theta)$ of the output values in the data pre-processing step. Thus, during training, every signed distance function input sample point $(\rho_o, \theta_o, \bar{\epsilon}_{p,o})$ is mapped to an output value:

$$\phi_\zeta(\rho_o, \theta_o, \bar{\epsilon}_{p,o}) = \phi(\rho_o, \theta_o, \bar{\epsilon}_{p,o}) + \zeta(\rho_o, \theta_o). \tag{63}$$

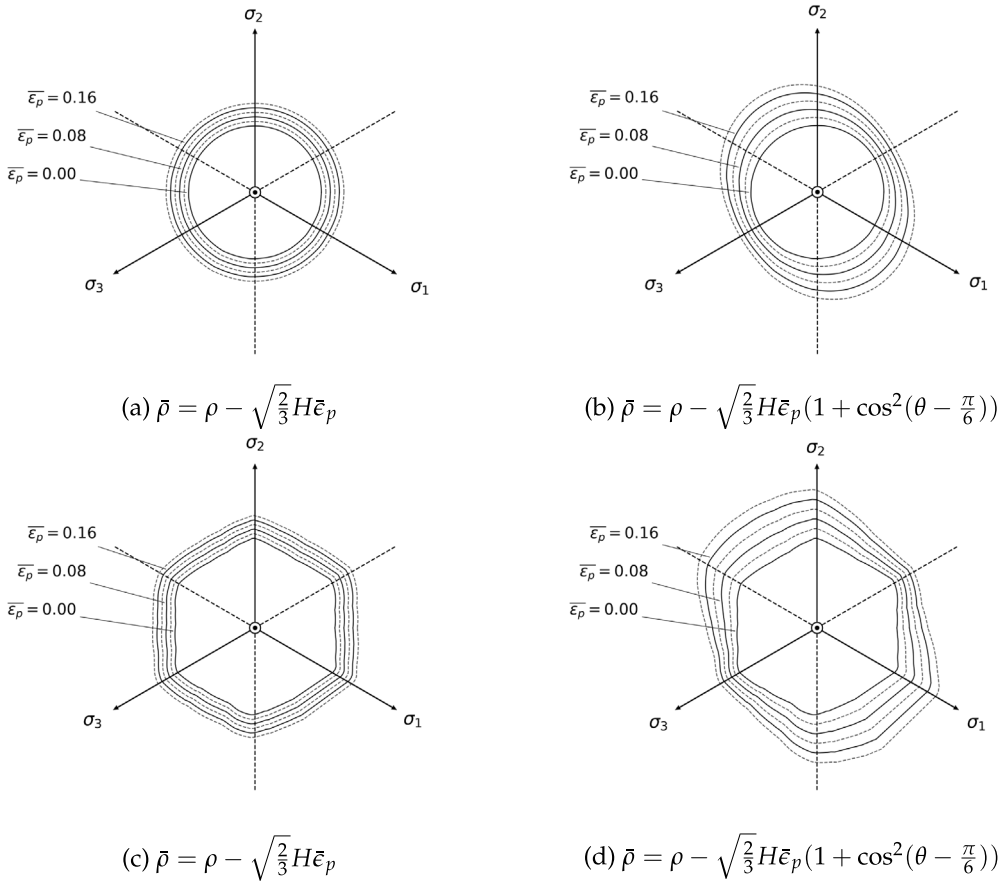


Fig. 26. Custom hardening transformations of initial neural network yield surfaces. Transformations (a) and (c) emulate simple isotropic hardening (dilation of yield surface). Transformations (b) and (d) emulate a mixed mode hardening mechanism (dilation and change of shape). The transformations are implemented by modifying the neural network input radius ρ.

During the prediction step, the true value of the signed distance function can recovered by subtracting the know value of $\zeta(\rho_o, \theta_o)$ from the prediction output. The helper function in this work was chosen as $\zeta(\rho, \theta) = 2\bar{\rho}\cos(\theta/3)$, where $\bar{\rho}$ is the mean value of the radii in the yield function data set.

Appendix C. Verification exercise with custom hardening

The plasticity components of the neural network elastoplasticity framework can further be decomposed by separating the initial yield surface and its evolution — the hardening law. We are introducing a method to apply custom hardening laws to the neural network approximated yield functions. The initial yield surface is controlled by a neural network of the form $\tilde{f}(\rho, \theta)$ with only the Lode's coordinates as inputs. The hardening is handled by a separate hardening law. In plasticity literature, hardening is usually implemented by transforming the yield surface — changing the yield stress value. However, in the case of our neural network yield function approximation, the yield stress in not explicitly defined and cannot be immediately modified. To overcome this obstacle, we define the desired hardening laws to the neural network input instead of the assumed yield stress. Specifically, we define a hardening law as transformation $\boldsymbol{L}$ of the original Lode's coordinates ρ and θ, such that:

$$\boldsymbol{L}(\rho, \theta, \xi) = \langle L_\rho(\rho, \theta, \xi), L_\theta(\rho, \theta, \xi) \rangle = \langle \rho_L, \theta_L \rangle, \tag{64}$$

where $L_r(\rho, \theta, \xi)$ and $L_\theta(\rho, \theta, \xi)$ are the parametric equations that transform ρ and θ into the input variables ρ_L, θ_L respectively after hardening, and ξ is an internal hardening variable.

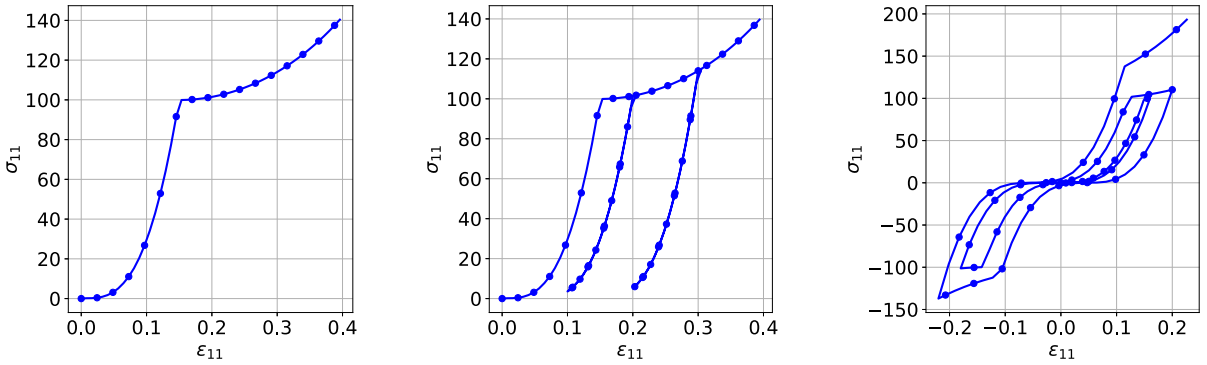


Fig. 27. Monolithic loading (LEFT), loading–unloading (MIDDLE), and cyclic loading (RIGHT) path predictions for a fictitious non-linear energy functional and hardening elastoplasticity neural network framework. The initial yield surface is predicted by the Von Mises yield surface neural network. The ANN elastoplastic framework can handle highly non-linear hyperelastic energy functionals and custom hardening laws. The stress measure is in kPa.

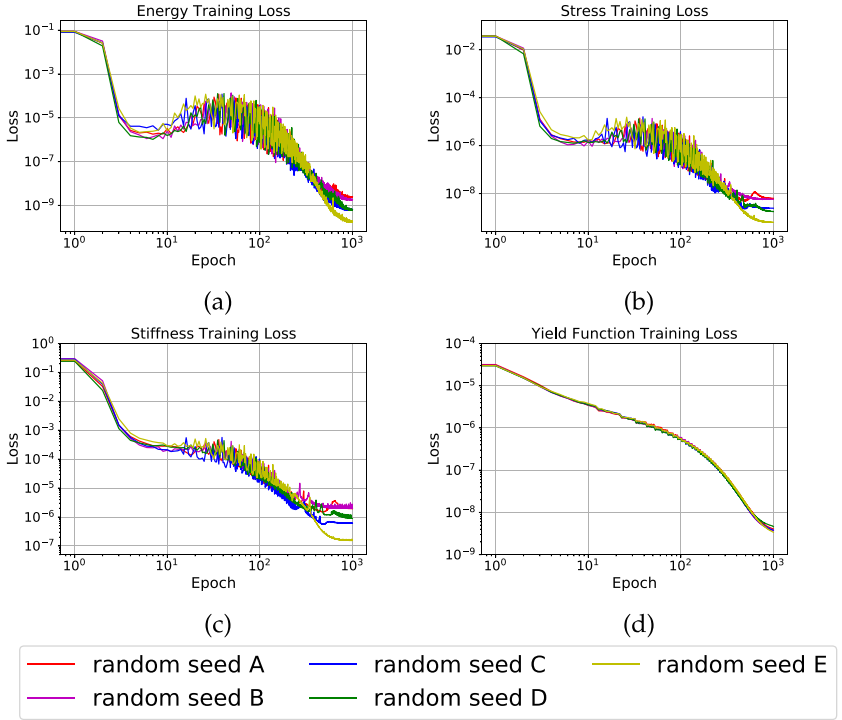


Fig. 28. Training loss comparison (a) the energy, (b) the stress, (c) the stiffness of an H^2 training objective of a dmmdmd architecture for linear elasticity, and (d) the yield function of a J2 plasticity neural network with 5 different random seeds.

Common literature hardening laws can be translated into input transformations of this type and applied to the neural network yield functions through geometric interpretation. For example, in the simple case of isotropic hardening of the Von Mises plasticity model, hardening in the π-plane can be interpreted as the dilation of the circular yield surface — i.e. increase of the radius ρ_y where there is yielding. In the case of a neural network approximating the Von Mises yield function, the value of the current r_y would not be readily available to modify. For that reason, instead of increasing the yield radius ρ_y, we opt for decreasing the input radius ρ of the neural

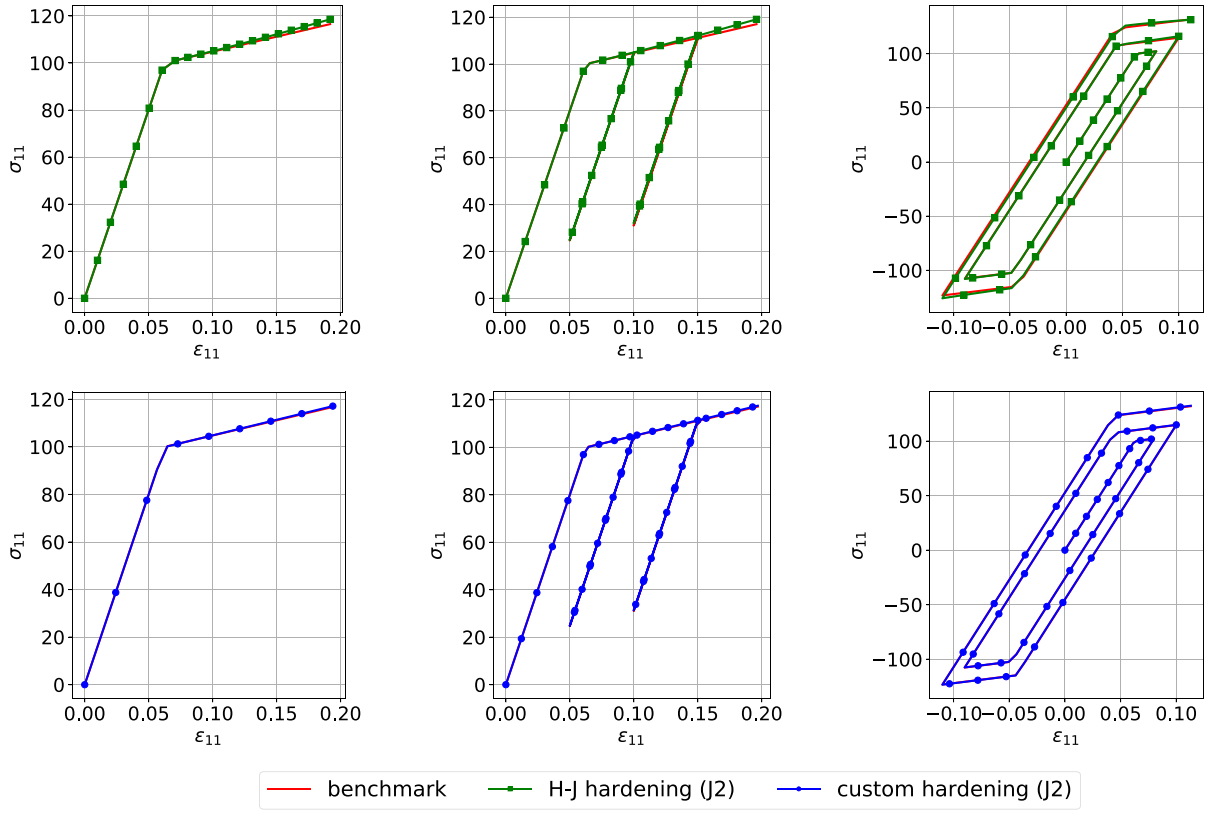


Fig. 29. Comparison of the neural network elastoplastic framework (linear elasticity and Von Mises plasticity) with benchmark simulation data for monolithic (LEFT), loading–unloading (MIDDLE), and cyclic loading (RIGHT) paths. TOP: The yield function NN replaces the yield function and the hardening law. BOTTOM: The yield function NN predicts only the initial yield surface and a custom identified hardening law is applied. The stress measure is in kPa.

network an equivalent amount. The transformed radius $\bar{\rho}$ is defined as:

$$\bar{\rho} = \bar{L}_{\rho}(\rho, \bar{\epsilon}_p) = \rho - \sqrt{\frac{2}{3}} H \bar{\epsilon}_p, \tag{65}$$

where H is the material's identified hardening modulus. Any custom hardening model can be applied with the right conversion to an input transformation. This enables for even more flexibility when assembling the theoretical components of the elastoplastic framework system. The hyperelastic energy functional, the initial yield surface, and the hardening law are independent of each other and can separately replaced. Furthermore, being able to assign a hardening law as a separate process in the data-driven yield function could prove valuable when only information of the initial yield surface is available in the data.

A few different cases of custom hardening transformations are demonstrated in Fig. 26. The initial yield surfaces are predicted from a neural network approximator — all the points approximated have an accumulated plastic strain $\bar{\epsilon}_p = 0$. Fig. 26(a) and (c) showcase a simple isotropic hardening case emulated by reducing the neural network input radius ρ uniformly for all the Lode's angles θ on the π-plane. The hardening mechanism can be geometrically interpreted as a dilation of the initial yield surface. Fig. 26(b) and (d) showcase two modes of hardening acting simultaneously — a dilation and an elongation towards a preferred direction of the initial yield surface.

In the current formulation, the neural network elastoplastic framework can consist of any isotropic hyperelastic energy functional and isotropic yield function. To demonstrate the framework's capability to capture non-linear behaviors, we have implemented a fictitious highly non-linear and a fictitious non-linear custom hardening law. The energy functional neural network is trained on data set based a modification on the linear elastic energy functional

with the shear part replaced with a highly non-linear term:

$$\breve{\psi}(\epsilon_v^{\mathrm{e}}, \epsilon_s^{\mathrm{e}}) = \frac{1}{2} K \epsilon_v^{\mathrm{e}\,2} + \frac{3}{2} G \epsilon_s^{\mathrm{e}\,4}. \tag{66}$$

The non-linear hardening law is implemented by applying a transformation on the Lode's radius input of the Von Mises yield function neural network. The hardening law $\breve{L}$ provides a transformed radius:

$$\breve{\rho} = \breve{\rho}(\rho, \bar{\epsilon}_p) = \rho \cdot (1 - \bar{\epsilon}_p^2)^6. \tag{67}$$

The prediction of the framework is demonstrated in Fig. 27. The framework provides great flexibility to decompose the material behavior for the elasticity, yield surface and hardening law — all of which can be individually replaced. This also allows for a combination of data-driven and handcrafted laws that can be tuned to closely replicate observed material behaviors.

Appendix D. Verification exercise on learning classical J2 plasticity with isotropic hardening

As a part of the verification exercise, we also test whether the proposed framework is able to deduce an elasto-plasticity model with linear elasticity and Von Mises plasticity with isotropic hardening. The elastoplastic ANN framework consists of a neural network approximating the linear elastic energy functional and a yield function neural network that approximates a Von Mises yield surface. The hardening law of the system is implemented in two different ways to demonstrate the flexibility of the framework — similar to Section 5 or following the custom hardening method of Appendix C (Eq. (65)). The material has a Young's Modulus of $E = 2.0799$ MPa, a Poisson ratio of $\nu = 0.3$, an initial yield stress of 100 kPa, and a hardening modulus of $H = 0.1E$.

To demonstrate the repeatability of the training process, we perform the training of the neural networks that represent a linear elasticity energy functional and a J2 yield function with 5 different random seeds. The training loss functions for these training experiments are demonstrated in Fig. 28. The neural network initialization appeared to have minimal effects on the training process results.

The comparison of the neural network elastoplastic framework with three benchmark simulations is shown in Fig. 29. The framework is tested against a monolithic loading path, a loading path with multiple unloading patterns, and a cyclic loading path. The framework can adequately capture loading and unloading patterns it has not been explicitly trained on.

References

[1] M. Pastor, O.C. Zienkiewicz, A.H.C. Chan, Generalized plasticity and the modelling of soil behaviour, Int. J. Numer. Anal. Methods Geomech. 14 (3) (1990) 151–190.

[2] Olgierd C. Zienkiewicz, A.H.C. Chan, M. Pastor, B.A. Schrefler, T. Shiomi, Computational Geomechanics, Vol. 613, Citeseer, 1999.

[3] Kun Wang, WaiChing Sun, Qiang Du, A cooperative game for automated learning of elasto-plasticity knowledge graphs and models with ai-guided experimentation, Comput. Mech. 64 (2) (2019) 467–499.

[4] Kun Wang, WaiChing Sun, Qiang Du, A non-cooperative meta-modeling game for automated third-party calibrating, validating, and falsifying constitutive laws with parallelized adversarial attacks, 2020, arXiv preprint arXiv:2004.09392.

[5] Yannis F. Dafalias, Bounding surface plasticity. i: Mathematical foundation and hypoplasticity, J. Eng. Mech. 112 (9) (1986) 966–987.

[6] D.I.H.D. Kolymbas, An outline of hypoplasticity, Arch. Appl. Mech. 61 (3) (1991) 143–151.

[7] Kun Wang, WaiChing Sun, Simon Salager, SeonHong Na, Ghonwa Khaddour, Identifying material parameters for a micro-polar plasticity model via X-ray micro-computed tomographic (CT) images: lessons learned from the curve-fitting exercises, Int. J. Multiscale Comput. Eng. 14 (4) (2016).

[8] James R. Rice, Inelastic constitutive relations for solids: an internal-variable theory and its application to metal plasticity, J. Mech. Phys. Solids 19 (6) (1971) 433–455.

[9] Rodney Hill, The Mathematical Theory of Plasticity, Vol. 11, Oxford university press, 1998.

[10] WaiChing Sun, A unified method to predict diffuse and localized instabilities in sands, Geomech. Geoengin. 8 (2) (2013) 65–75.

[11] Eric C. Bryant, WaiChing Sun, A micromorphically regularized cam-clay model for capturing size-dependent anisotropy of geomaterials, Comput. Methods Appl. Mech. Engrg. 354 (2019) 56–95.

[12] J. Ghaboussi, J.H. Garrett Jr., Xiping Wu, Knowledge-based modeling of material behavior with neural networks, J. Eng. Mech. 117 (1) (1991) 132–153.

[13] Tomonari Furukawa, Genki Yagawa, Implicit constitutive modelling for viscoplasticity using neural networks, Internat. J. Numer. Methods Engrg. 43 (2) (1998) 195–219.

[14] Stéphane Pernot, C.-H. Lamarque, Application of neural networks to the modelling of some constitutive laws, Neural Netw. 12 (2) (1999) 371–392.

[15] M. Lefik, D.P. Boso, B.A. Schrefler, Artificial neural networks in numerical modelling of composites, Comput. Methods Appl. Mech. Engrg. 198 (21–26) (2009) 1785–1804.
[16] Kun Wang, WaiChing Sun, A multiscale multi-permeability poroplasticity model linked by recursive homogenizations and deep learning, Comput. Methods Appl. Mech. Engrg. 334 (2018) 337–380.
[17] Yousef Heider, Kun Wang, WaiChing Sun, So (3)-invariance of informed-graph-based deep neural network for anisotropic elastoplastic materials, Comput. Methods Appl. Mech. Engrg. 363 (2020) 112875.
[18] Nikolaos Vlassis, Ran Ma, WaiChing Sun, Geometric deep learning for computational mechanics part i: Anisotropic hyperelasticity, Comput. Methods Appl. Mech. Engrg. 371 (2020).
[19] Daniele Versino, Alberto Tonda, Curt A. Bronkhorst, Data driven modeling of plastic deformation, Comput. Methods Appl. Mech. Engrg. 318 (2017) 981–1004.
[20] R.v. Mises, Mechanik der festen körper im plastisch-deformablen zustand, Nachr. Ges. Wiss. Göttingen, Math.-Phys. Kl. 1913 (1913) 582–592.
[21] William Prager, The theory of plasticity: a survey of recent achievements, Proc. Inst. Mech. Eng. 169 (1) (1955) 41–57.
[22] K.J. William, E.P. Warnke, Constitutive model for the triaxial behaviour of concrete (paper iii-l), in: Proc. Seminar on Concrete Structures Subjected to Triaxial Stresses, 1974.
[23] Daniel Charles Drucker, Some implications of work hardening and ideal plasticity, Quart. Appl. Math. 7 (4) (1950) 411–418.
[24] Ronaldo I. Borja, Alexander P. Amies, Multiaxial cyclic plasticity model for clays, J. Geotech. Eng. 120 (6) (1994) 1051–1070.
[25] Mahdi Taiebat, Yannis F. Dafalias, Sanisand: Simple anisotropic sand plasticity model, Int. J. Numer. Anal. Methods Geomech. 32 (8) (2008) 915–948.
[26] Kim Lau Nielsen, Viggo Tvergaard, Ductile shear failure or plug failure of spot welds modelled by modified gurson model, Eng. Fract. Mech. 77 (7) (2010) 1031–1047.
[27] C.D. Foster, R.A. Regueiro, Arlo F. Fossum, Ronaldo I. Borja, Implicit numerical integration of a three-invariant, isotropic/kinematic hardening cap plasticity model for geomaterials, Comput. Methods Appl. Mech. Engrg. 194 (50–52) (2005) 5109–5138.
[28] WaiChing Sun, Qiushi Chen, Jakob T. Ostien, Modeling the hydro-mechanical responses of strip and circular punch loadings on water-saturated collapsible geomaterials, Acta Geotech. 9 (5) (2014) 903–934.
[29] C. Miehe, N. Apel, M. Lambrecht, Anisotropic additive plasticity in the logarithmic strain space: modular kinematic formulation and implementation based on incremental minimization principles for standard materials, Comput. Methods Appl. Mech. Engrg. (ISSN: 00457825) 191 (47–48) (2002) 5383–5425, http://dx.doi.org/10.1016/S0045-7825(02)00438-3, URL http://linkinghub.elsevier.com/retrieve/pii/S0045782502004383.
[30] Ronaldo I. Borja, Plasticity. Modeling and Computation, Springer, Berlin Heidelberg, Berlin, Heidelberg, ISBN: 978-3-642-38546-9, 2013, http://dx.doi.org/10.1007/978-3-642-38547-6.
[31] Kailai Xu, Daniel Z. Huang, Eric Darve, Learning constitutive relations using symmetric positive definite neural networks, 2020, arXiv preprint arXiv:2004.00265.
[32] Marek Lefik, Bernhard A. Schrefler, Artificial neural network as an incremental non-linear constitutive model for a finite element code, Comput. Methods Appl. Mech. Engrg. 192 (28–30) (2003) 3265–3283.
[33] Ruiyang Zhang, Yang Liu, Hao Sun, Physics-informed multi-lstm networks for metamodeling of nonlinear structures, 2020, arXiv preprint arXiv:2002.10253.
[34] M.A. Bessa, R. Bostanabad, Zeliang Liu, A. Hu, Daniel W. Apley, C. Brinson, Wei Chen, Wing Kam Liu, A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality, Comput. Methods Appl. Mech. Engrg. 320 (2017) 633–667.
[35] M. Mozaffar, R. Bostanabad, W. Chen, K. Ehmann, Jian Cao, M.A. Bessa, Deep learning predicts path-dependent plasticity, Proc. Natl. Acad. Sci. 116 (52) (2019) 26414–26420.
[36] Annan Zhang, Dirk Mohr, Using neural networks to represent von mises plasticity with isotropic hardening, Int. J. Plast. (2020) 102732.
[37] Y.M.A. Hashash, S. Jung, J. Ghaboussi, Numerical implementation of a neural network based material model in finite element analysis, Internat. J. Numer. Methods Engrg. 59 (7) (2004) 989–1005.
[38] Xin Wang, Yi Qin, Yi Wang, Sheng Xiang, Haizhou Chen, Reltanh: An activation function with vanishing gradient resistance for sae-based dnns and its application to rotating machinery fault diagnosis, Neurocomputing 363 (2019) 88–98.
[39] Matías Roodschild, Jorge Gotay Sardiñas, Adrián Will, A new approach for the vanishing gradient problem on sigmoid activation, Prog. Artif. Intell. 9 (4) (2020) 351–360.
[40] James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, David D. Cox, Hyperopt: a python library for model selection and hyperparameter optimization, Comput. Sci. Discov. 8 (1) (2015) 014008.
[41] Brent Komer, James Bergstra, Chris Eliasmith, Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn, in: ICML Workshop on AutoML, Vol. 9, Citeseer, 2014, p. 50.
[42] Wojciech M. Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, Razvan Pascanu, Sobolev training for neural networks, in: Advances in Neural Information Processing Systems, 2017, pp. 4278–4287.
[43] W.R. Wawersik, L.W. Carlson, D.J. Holcomb, R.J. Williams, New method for true-triaxial rock testing, Int. J. Rock Mech. Min. Sci. 34 (3–4) (1997) 330–e1.
[44] Bezalel Haimson, John W. Rudnicki, The effect of the intermediate principal stress on fault formation and fault angle in siltstone, J. Struct. Geol. 32 (11) (2010) 1701–1711.
[45] Christopher M. Bishop, et al., Neural Networks for Pattern Recognition, Oxford university press, 1995.
[46] W. Lode, Versuche über den einfluß der mittleren hauptspannung auf das fließen der metalle eisen, kupfer und nickel, Z. Phys. 36 (11–12) (1926) 913–939.

[47] J.H. Argyris, G. Faust, J. Szimmat, E.P. Warnke, K.J. Willam, Recent developments in the finite element analysis of prestressed concrete reactor vessels, Nucl. Eng. Des. 28 (1) (1974) 42–75.
[48] François Chollet, et al., Keras, 2015, https://keras.io.
[49] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al., Tensorflow: A system for large-scale machine learning, in: 12th {$USENIX$} Symposium on Operating Systems Design and Implementation ({$OSDI$} 16), 2016, pp. 265–283.
[50] Dougal Maclaurin, David Duvenaud, Ryan P. Adams, Autograd: Effortless gradients in numpy, in: ICML 2015 AutoML Workshop, Vol. 238, 2015, p. 5.
[51] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio, Learning phrase representations using rnn encoder–decoder for statistical machine translation, 2014, arXiv preprint arXiv:1406.1078.
[52] Felix A. Gers, Jürgen Schmidhuber, Fred Cummins, Learning to forget: Continual prediction with lstm, 1999.
[53] Yann LeCun, Yoshua Bengio, et al., Convolutional networks for images, speech, and time series, Handb. Brain Theory Neural Netw. 3361 (10) (1995) 1995.
[54] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, Wavenet: A generative model for raw audio, 2016, arXiv preprint arXiv:1609.03499.
[55] Ronaldo I. Borja, Chao-Hua Lin, Francisco J. Montáns, Cam-clay plasticity, part iv: Implicit integration of anisotropic bounding surface model with nonlinear hyperelasticity and ellipsoidal loading function, Comput. Methods Appl. Mech. Engrg. 190 (26–27) (2001) 3293–3323.
[56] Timothy Dozat, Incorporating nesterov momentum into adam, 2016.
[57] Eduardo A. de Souza Neto, Djordje Peric, David R.J. Owen, Computational Methods for Plasticity: Theory and Applications, John Wiley & Sons, 2011.
[58] Hajime Matsuoka, Teruo Nakai, Relationship among tresca, mises, mohr-coulomb and matsuoka-nakai failure criteria, Soils Found. 25 (4) (1985) 123–128.
[59] A.J. Abbo, S.W. Sloan, A smooth hyperbolic approximation to the mohr-coulomb yield criterion, Comput. Struct. 54 (3) (1995) 427–441.
[60] Julian Kochmann, Stephan Wulfinghoff, Stefanie Reese, Jaber Rezaei Mianroodi, Bob Svendsen, Two-scale fe–fft-and phase-field-based computational modeling of bulk microstructural evolution and macroscopic material behavior, Comput. Methods Appl. Mech. Engrg. 305 (2016) 89–110.
[61] Julian Kochmann, Lisa Ehle, Stephan Wulfinghoff, Joachim Mayer, Bob Svendsen, Stefanie Reese, Efficient multiscale fe-fft-based modeling and simulation of macroscopic deformation processes with non-linear heterogeneous microstructures, in: Multiscale Modeling of Heterogeneous Structures, Springer, 2018, pp. 129–146.
[62] Barré D.E. Saint Venant, Memoire sur l'etablissement des equations differentielles des mouvements interieurs operes dans les corps ductiles au dela des limites ou le elasticite pourtrait les ramener a leur premier etat, C. R. Acad. Sci., Paris 70 (1870) 473–480.
[63] K_H. Roscoe, J.B. Burland, On the generalized stress–strain behaviour of wet clay, 1968.
[64] G.T. Houlsby, The use of a variable shear modulus in elastic–plastic models for clays, Comput. Geotech. 1 (1) (1985) 3–13.
[65] F. Bachmann, Ralf Hielscher, Helmut Schaeben, Texture analysis with MTEX – Free and open source software toolbox, 2010, http://dx.doi.org/10.4028/www.scientific.net/SSP.160.63.
[66] Ran Ma, WaiChing Sun, Fft-based solver for higher-order and multi-phase-field fracture models applied to strongly anisotropic brittle materials and poly-crystals, Comput. Methods Appl. Mech. Engrg. (2019) tentatively accepted.
[67] Ran Ma, WaiChing Sun, Computational thermomechanics for crystalline rock. part ii: Chemo-damage-plasticity and healing in strongly anisotropic polycrystals, Comput. Methods Appl. Mech. Engrg. 369 (2020) 113184.
[68] Ronaldo I. Borja, Jon R. Wren, Discrete micromechanics of elastoplastic crystals, Internat. J. Numer. Methods Engrg. 36 (22) (1993) 3815–3840.