Physically recurrent neural networks for path-dependent heterogeneous materials
Embedding constitutive models in a data-driven surrogate

Maia, M. A.; Rocha, I. B.C.M.; Kerfriden, P.; van der Meer, F. P.

DOI
10.1016/j.cma.2023.115934

Publication date
2023

Document Version
Final published version

Published in
Computer Methods in Applied Mechanics and Engineering

Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 407 (2023) 115934
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Physically recurrent neural networks for path-dependent heterogeneous materials: Embedding constitutive models in a data-driven surrogate

M.A. Maia[a,*], I.B.C.M. Rocha[a], P. Kerfriden[b], F.P. van der Meer[a]

[a] *Delft University of Technology, Department of Civil Engineering and Geosciences, P.O. Box 5048, 2600GA, Delft, The Netherlands*
[b] *Mines Paris, PSL University, Centre des matériaux, 63-65 Rue Henri-Auguste Desbruères BP87, F-91003, Évry, France*

Received 19 September 2022; received in revised form 14 January 2023; accepted 31 January 2023
Available online xxxx

Abstract

Driven by the need to accelerate numerical simulations, the use of machine learning techniques is rapidly growing in the field of computational solid mechanics. Their application is especially advantageous in concurrent multiscale finite element analysis (FE^2) due to the exceedingly high computational costs often associated with it and the high number of similar micromechanical analyses involved. To tackle the issue, using surrogate models to approximate the microscopic behavior and accelerate the simulations is a promising and increasingly popular strategy. However, several challenges related to their data-driven nature compromise the reliability of surrogate models in material modeling. The alternative explored in this work is to reintroduce some of the physics-based knowledge of classical constitutive modeling into a neural network by employing the actual material models used in the full-order micromodel to introduce non-linearity. Thus, path-dependency arises naturally since every material model in the layer keeps track of its own internal variables. For the numerical examples, a composite Representative Volume Element with elastic fibers and elasto-plastic matrix material is used as the microscopic model. The network is tested in a series of challenging scenarios and its performance is compared to that of a state-of-the-art Recurrent Neural Network (RNN). A remarkable outcome of the novel framework is the ability to naturally predict unloading/reloading behavior without ever seeing it during training, a stark contrast with popular but data-hungry models such as RNNs. Finally, the proposed network is applied to FE^2 examples to assess its robustness for application in nonlinear finite element analysis.

Keywords: Artificial Neural Networks (ANNs); Multiscale; Heterogeneous materials; Path-dependency

1. Introduction

Driven by the need to accelerate numerical simulations, machine learning (ML) techniques are rapidly growing in the field of computational solid mechanics. The use of surrogate models in particular is especially advantageous in concurrent multiscale finite element analysis (FE^2). In this approach, each integration point at the macroscale is linked to a microscopic model, allowing complex materials such as composite materials to be explicitly modeled using relatively simple constitutive models. Although appealing, the computational cost associated with

* Corresponding author.
E-mail address: m.alvesmaia@tudelft.nl (M.A. Maia).

https://doi.org/10.1016/j.cma.2023.115934

the concurrent Finite Element (FE) simulations is often prohibitive, hindering its widespread use in practical engineering-scale applications. In that sense, another layer of approximation, for instance based on ML techniques, that help accelerate FE^2 simulations is key.

However, despite the successful use of this type of technique in several fields (*e.g.*, speech recognition, language processing, and biomedical sciences), the use of surrogate models in material modeling is still filled with challenges and open issues. One of them relates to the data scarcity that exposes how data-driven models do not perform as well in extrapolation as they do within their training range. In general, another shortcoming of black-box models is their lack of interpretability. Although not a limiting aspect by itself, the limited interpretability these models offer can make it difficult to understand whether the fitted model is capable of yielding physically consistent solutions for loading paths different than those seen during training.

In this scenario, the definition of a Design of Experiments (DoE) strategy plays an important role in the generalization capabilities of the surrogate model. The designer needs to take into account the computational (or experimental) budget available to collect the data and what type of loading scenarios are expected to be experienced by the model. The former is known *a priori*, but the latter is not known exactly until the finite element simulation itself is run. As a consequence, large training and testing datasets are usually employed in an attempt to cover all different loading scenarios for a given strain/stress range.

When path-dependency is present, conceiving an efficient DoE becomes even more complex since stresses now depend on the history of the material, leading to potentially infinite parameter space. One way to bypass the computationally intensive approach is to run and incorporate new load paths on-the-fly, updating the surrogate model as necessary as in the works of Goury et al. [1] and Rocha et al. [2]. In addition to the definition of the DoE, the physics-based assumptions about the material should also be wisely taken into account when choosing the modeling approach. For instance, conventional Artificial Neural Networks (ANNs) with strains and stresses taken as input and output, respectively, will fail to capture phenomena such as elastic unloading. This scenario is illustrated by Vlassis and Sun [3] and occurs due to the unique mapping between inputs and outputs that does not reflect two different stress states for the same strain. One way to differentiate the different paths is by augmenting the feature space of the ANN with extra variables that carry partial information about the history of stresses and/or strains [4,5].

Another highly popular strategy to handle path-dependency is to use Recurrent Neural Networks (RNNs), a variation of ANNs capable of learning from sequential data. This is done by incorporating the previous state of the network when making predictions for the current state. However, this approach often suffers from short-term memory when dealing with long sequences. To address that issue, more complex architectures with more model parameters and fine control of the flow of information retained through a sequence of information were proposed (*e.g.*, GRU and LSTM). These networks gained popularity and have been used to model a wide variety of materials that show history-dependency [6–11].

However, RNNs are still severely limited by the curse of dimensionality associated with sampling arbitrarily long strain paths. The overview on the potential of RNNs for modeling path-dependent plasticity presented by Gorji et al. [12] illustrates the multiple ways in which one can define a DoE and argue that GRUs, in particular, can be used to model the plastic response of materials provided a rich enough training dataset is used. However, the authors do not investigate further how these networks would perform in scenarios slightly different than those seen in training or their efficiency in an FE^2 framework.

Recently, a new class of ANNs called Physics-Informed Neural Networks (PINNs) was proposed by [13] to solve any given laws of physics described by general nonlinear partial differential equations. For training PINNs, in addition to the data used to model the governing equation, the loss function is augmented with information on the imposed physical constraints (*e.g.* initial and boundary conditions). In this approach, automatic differentiation plays a key role in allowing the outputs of the network to be differentiated with respect to the input.

Applications of PINNs as surrogate constitutive models are still relatively new, but a few works showcase their potential in modeling elastoplastic [14,15] and elastic–viscoplastic behaviors [16]. In [14,16], the loss function is also incremented with terms related to the physics constraints assumed by the material models considered (*e.g.* the Karush–Kuhn–Tucker conditions on the yield function). While both works approximate the displacements and the stress based on the position of the material point at the macroscale and on the material properties, Haghighat et al. [14] further investigates the applicability of their surrogate model in a sensitivity analysis study. Finally, Eghbalian et al. [15] proposed an architecture where the nonlinear incremental elasticity and the strain decomposition are hardwired into the networks's architecture and the loss function. Thus leading to improved capability in predicting unseen loading scenarios over conventional ANNs.

Inspired by PINNs, Masi et al. [17] tailored a network architecture capable of ensuring thermodynamically consistent predictions by evaluating the numerical derivatives of the network with respect to its inputs. Later, Masi and Stefanou [18] proposed a key extension to the approach, in which the identification of the internal variables is no longer user-dependent. For that purpose, an encoder–decoder architecture is incorporated to identify the minimum number of internal variables from the set of internal coordinates of the system (*e.g.*, displacement fields, internal forces, etc.) in an unsupervised manner. The feature allows the recovery of the full-field information of the microstructure by the decoder. Another strategy that takes into account the derivatives of the approximated functions to impose thermodynamically consistency is proposed by Vlassis and Sun [3]. The authors treat the yield function as a signed distance function level set and formulate a supervised learning task that deduces the learned evolving yield function against a monotonically increasing accumulated plastic strain. As a result, cyclic loading paths can be predicted based only on monotonic data. For a recent and comprehensive review on the state-of-the-art literature of ANNs in the constitutive modeling of composite materials, the reader is referred to Liu et al. [19].

Different from regular ANNs or RNNs, the Deep Material Network (DMN) proposed by Liu et al. [20] assigns physical meaning to the hyperparameters. The DMN learns the hidden topology representation of the micromodel (RVE) based on the stiffness matrices and residual stresses of the different material phases. A major advantage of DMNs is the ability to extrapolate to nonlinear behavior based only in elastic snapshots. Several follow-up publications and improvements on DMN can be found in the literature. The author's latest work [21] is dedicated to tackling multiscale failure analysis, a far less explored avenue by the community. Other successful applications in this field can be found in the works of Kerfriden et al. [22], Bessa et al. [23] and Oliver et al. [24], where different reduced-order modeling strategies were employed.

In an alternative approach that incorporates some level of physics in the surrogate model, Fuhg et al. [25] combines a physical model that accounts partially for the constitutive behavior of the material and a Gaussian Process (GP) correction to locally improve the baseline physical model. Although this boosts accuracy inside the training space, in practice, it does not translate to benefits in terms of extrapolation. Another alternative that retains some of the physics embedded in the full-order solution is model order reduction. These methods are frequently employed to alleviate the computational cost of FE^2 [26,27] and usually offer better generalization properties to unseen points than common surrogate models. The drawback is that they are inherently slower [28]. Such techniques can also be coupled with RNNs [29], ANNs [5] and GPs [30] to quickly infer the coefficients of the reduced basis approximation for arbitrary parameter values.

Although a large body of literature has been devoted to applying ML techniques in the solid mechanics field, the gaps and issues left by data-driven surrogate models are still an open issue, compromising their reliable and widespread use in practical applications. In this work, a new network design is proposed specifically for the modeling of path-dependent materials and to accelerate concurrent finite element simulations. In Section 2 the FE^2 method is presented, while in Section 3 one of the most popular approaches to tackle the computational bottleneck originated from it is briefly discussed. In Section 4, the main features of the novel neural network are described. In Section 5, the Design of Experiments and methodology adopted for the comparative study shown in Section 6 is described. In this study, the performance of the proposed network is compared to a RNN for a single-scale problem. In Section 7 the novel approach is integrated into an FE^2 framework and tested in two applications for robustness and accuracy. In Section 8, the network is tested for other combinations of material models to illustrate its flexibility. Finally, conclusions are presented in Section 9.

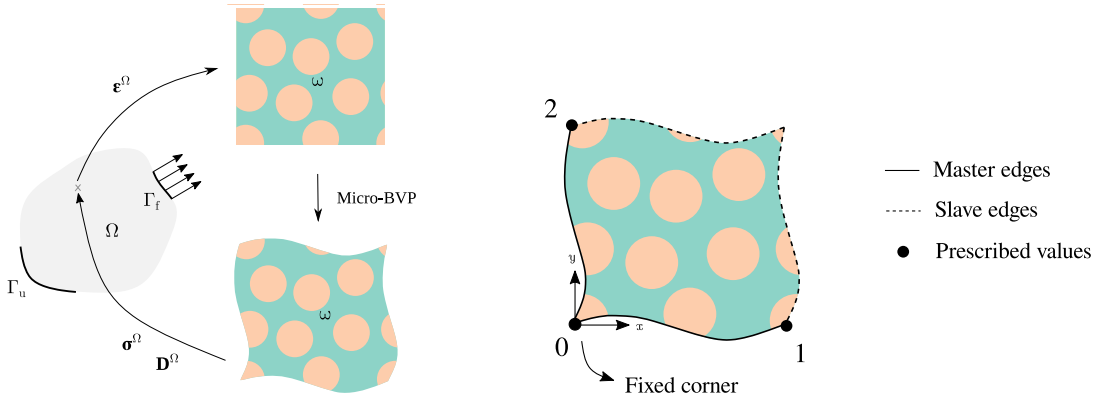
2. Concurrent multiscale analysis

Let Ω define the macroscopic domain being modeled. To find the internal stresses and displacement field of such body in absence of body forces, a boundary value problem that satisfies the following equilibrium equations is defined:

$$\nabla \boldsymbol{\sigma}^{\Omega} = \mathbf{0} \tag{1}$$

where ∇ is the divergence operator and $\boldsymbol{\sigma}^{\Omega}$ is the macroscopic stress, which depends on the macroscopic displacement field $\mathbf{u}^{\Omega}$ (for simplicity, this dependence is omitted). The governing equations are subjected to the boundary conditions:

$$\boldsymbol{\sigma}^{\Omega}\,\mathbf{n} = \mathbf{t}^{\Gamma_f} \quad \text{on } \Gamma_f \qquad \mathbf{u}^{\Omega} = \mathbf{u}^{\Gamma_u} \quad \text{on } \Gamma_u \tag{2}$$



(a) Schematic representation of FE^2 (b) Controlling nodes of RVE

Fig. 1. Scheme of FE^2 framework and definition of the boundary value problem on RVE.

where $\mathbf{n}$ is the normal to the surface Γ_f and $\mathbf{u}^{\Gamma_u}$ and $\mathbf{t}^{\Gamma_f}$ represent a set of Dirichlet and Neumann boundary conditions acting on the body surface such that $\Gamma_u \cap \Gamma_f = \emptyset$ as illustrated in Fig. 1(a). To relate strains and stresses, a constitutive model $\mathcal{D}$ is required:

$$\boldsymbol{\sigma}^{\Omega} = \mathcal{D}^{\Omega}\ (\boldsymbol{\varepsilon}^{\Omega}, \boldsymbol{\alpha}^{\Omega}) \tag{3}$$

where $\boldsymbol{\alpha}^{\Omega}$ are history variables that account for path-dependency and $\boldsymbol{\varepsilon}^{\Omega}$ is the macroscopic strain defined under small displacement assumptions as:

$$\boldsymbol{\varepsilon}^{\Omega} = \frac{1}{2}\left(\nabla \mathbf{u}^{\Omega} + (\nabla \mathbf{u}^{\Omega})^{\mathrm{T}}\right) \tag{4}$$

In the concurrent multiscale approach, the model $\mathcal{D}^{\Omega}$ is not directly formulated but is instead obtained by nesting a lower scale finite element model to each integration point. In that scale, the microscopic structure of complex materials can be explicitly modeled using simpler constitutive models for each of the components. Further discussion on how to solve the microscopic problem and link both scales is shown in Sections 2.1 and 2.2.

To solve the boundary value problem at the macroscale, the Finite Element (FE) method is employed to discretize the domain Ω into a number of elements connected by nodes with N degrees of freedom. The global equilibrium is solved iteratively in its discretized weak form:

$$\mathbf{r} = \mathbf{f}^{\Gamma} - \mathbf{f}^{\Omega}(\mathbf{u}^{\Omega}) = \mathbf{0} \tag{5}$$

where $\mathbf{r} \in \mathbb{R}^N$ is a residual vector that goes to zero when equilibrium is reached, $\mathbf{f}^{\Gamma} \in \mathbb{R}^N$ is the global external vector that represents the Neumann boundary conditions and $\mathbf{f}^{\Omega} \in \mathbb{R}^N$ is the global internal force vector given by a volume integral:

$$\mathbf{f}^{\Omega} = \mathop{\mathrm{A}}_{e=1}^{n_e} \int_{\Omega_e} \mathbf{B}_e^{\mathrm{T}}\ \boldsymbol{\sigma}^{\Omega}(\mathbf{u}_e^{\Omega})\ \mathrm{d}\Omega \tag{6}$$

where A is an assembly operator that takes into account the connectivities between the elements and the global system and $\mathbf{B}$ is a matrix with the spatial derivatives of the shape functions used to interpolate nodal displacements. Finally, an iterative procedure is adopted to solve Eq. (5) for the macroscopic displacement field:

$$\Delta \mathbf{u}^{\Omega} = \mathbf{u}_n^{\Omega} - \mathbf{u}_o^{\Omega} = -\mathbf{K}_o^{-1}\mathbf{r}_o \tag{7}$$

where the subscripts o and n refer to old and new analysis increments, respectively and $\mathbf{K} \in \mathbb{R}^{N\times N}$ is the global tangent stiffness matrix given by:

$$\mathbf{K} = \mathop{\mathrm{A}}_{e=1}^{n_e} \int_{\Omega_e} \mathbf{B}_e^{\mathrm{T}}\ \mathbf{D}_e^{\Omega}(\mathbf{u}_e^{\Omega})\ \mathbf{B}_e\ \mathrm{d}\Omega \tag{8}$$

and $\mathbf{D}^{\Omega}$ is the constitutive tangent matrix, discussed in Section 2.2.

The key difference to a classical FE simulation lies in the embedding of another FE model in the macroscopic integration points. Here, to obtain the internal forces in Eq. (6) and the tangent stiffness matrix in Eq. (8) for a single integration point of the macroscale, one needs to run an entire FE model instead of a single evaluation of a homogeneous material model. This is the most computationally expensive part of the framework and is where the proposed network aims to tackle. The approach here is to replace the solution to the microscopic problem (discussed in Section 2.1) with a surrogate model, specifically a neural network. The homogenization procedure required to upscale the responses to the macroscale is discussed in Section 2.2.

2.1. Microscopic scale

Let ω be a Representative Volume Element (RVE) of the microscopic material features whose behavior is to be upscaled. Assuming that the principle of separation of scales (*i.e.* $\Omega \gg \omega$) holds, the two scales can be linked by enforcing:

$$\mathbf{u}^{\omega} = \boldsymbol{\varepsilon}^{\Omega}\mathbf{x}^{\omega} + \tilde{\mathbf{u}} \tag{9}$$

where the linear displacement field is the result of the imposed macroscopic strains $\boldsymbol{\varepsilon}^{\Omega}$ and the fluctuation field $\tilde{\mathbf{u}}$ is the result of microscopic inhomogeneities. The principle of separation of scales implies the stain averaging theorem that states that the macroscopic strains are considered uniform over the RVE domain:

$$\boldsymbol{\varepsilon}^{\Omega}(\mathbf{x}^{\Omega}) = \frac{1}{\|\omega\|}\int_{\omega} \boldsymbol{\varepsilon}^{\omega}(\mathbf{x}_{\omega})\, \mathrm{d}\omega \tag{10}$$

where $\boldsymbol{\varepsilon}^{\omega}$ is the microscopic strain tensor. Therefore, the microscopic displacement field in Eq. (9) can only satisfy (10) if the fluctuation displacement field vanishes at the RVE boundary when upscaling quantities. An additional requirement on the fluctuation field having zero resultant work at the boundaries arises from the Hill–Mandel principle. Both requirements are met using Periodic Boundary Conditions (PBC) to represent the behavior of a macroscopic bulk material point. Fig. 1(b) illustrates the node groups and boundary edges needed to implement the PBC. In Section 5, the generation of $\boldsymbol{\varepsilon}$-$\boldsymbol{\sigma}$ paths for the training of the surrogate models is obtained by setting a user-defined function to set the prescribed displacements $\mathbf{u}_1$ and $\mathbf{u}_2$.

Finally, keeping the hypothesis of small strains, the stress equilibrium problem is described as:

$$\nabla\boldsymbol{\sigma}^{\omega} = \mathbf{0} \qquad \boldsymbol{\sigma}^{\omega} = \mathcal{D}^{\omega}\left(\boldsymbol{\varepsilon}^{\omega}, \boldsymbol{\alpha}^{\omega}\right) \qquad \boldsymbol{\varepsilon}^{\omega} = \frac{1}{2}\left(\nabla\mathbf{u}^{\omega} + (\nabla\mathbf{u}^{\omega})^{\mathrm{T}}\right) \tag{11}$$

where $\mathbf{u}^{\omega}$ is the microscopic displacement field and $\boldsymbol{\sigma}^{\omega}$ and $\boldsymbol{\varepsilon}^{\omega}$ are the microscopic stress and strain tensors, respectively. An analogous procedure to the one detailed in Section 2 is used to find the microscopic displacement field (subjected to the periodic boundary conditions). Note that at this scale, regular physics-based material models $\mathcal{D}^{\omega}$ (*e.g.* elastoplasticity, viscoelasticity, etc.) are employed to represent the constitutive behavior of the homogeneous material of the discretized elements.

2.2. Homogenization procedure

After convergence of the microscopic displacement field $\mathbf{u}^{\omega}$, the upscaling procedure is performed based on the Hill–Mandel principle. The principle postulates that the macroscopic stress power must equal the volume average of the microscopic power over the RVE. Considering the definition in Eq. (9), an expression similar to Eq. (10) is obtained for the homogenized stresses:

$$\boldsymbol{\sigma}^{\Omega} = \frac{1}{\|\omega\|}\int_{\omega} \boldsymbol{\sigma}^{\omega}\mathrm{d}\omega \tag{12}$$

As for the macroscopic constitutive tangent stiffness $\mathbf{D}^{\Omega}$, a probing operator $\mathcal{P}$ is applied on the global microscopic tangent stiffness matrix $\mathbf{K}^{\omega}$ without the need to invert it as proposed by Nguyen et al. [31].

3. Recurrent neural networks

In this section, a brief overview of the working mechanisms of Recurrent Neural Networks is presented. Although part of the paper is dedicated to comparing them with the novel approach, here the idea is to use well-known concepts from ANNs and RNNs to illustrate features of the proposed network in the following sections.

As a starting point, consider a conventional feed-forward neural network to surrogate the nonlinear constitutive relationship of a path-independent material given by the following parametric regression model:

$$\widehat{\boldsymbol{\sigma}}^{\Omega} = \mathcal{NN}\left(\boldsymbol{\varepsilon}^{\Omega}, \mathbf{W}, \mathbf{b}\right) \tag{13}$$

where $\mathbf{W}$ and $\mathbf{b}$ are weights and biases calibrated through a fitting procedure based on observations of the actual microscopic model. During training, the strains are fed to the first neural layer (input layer) and values are propagated until the final layer (output layer) to give the predicted stresses $\widehat{\boldsymbol{\sigma}}^{\Omega}$. These are in turn compared to the ground truth value according to a loss function. Based on that, the model parameters are adjusted so that the error between the predicted stresses and the actual stresses is minimized:

$$\mathbf{W}, \mathbf{b} = \underset{\overline{\mathbf{W}}\ \overline{\mathbf{b}}}{\operatorname{argmin}} \sum_{i\ \in\ \mathbf{X}} \parallel \boldsymbol{\sigma}_i(\boldsymbol{\varepsilon}_i^{\Omega}) - \widehat{\boldsymbol{\sigma}}_i(\boldsymbol{\varepsilon}_i^{\Omega}, \overline{\mathbf{W}}, \overline{\mathbf{b}}) \parallel^2 \tag{14}$$

where $\mathbf{X} \in \mathbb{R}^{n_\varepsilon \times N}$ is a snapshot matrix with N pairs of $\boldsymbol{\varepsilon}^{\Omega} - \boldsymbol{\sigma}^{\Omega}$ obtained from microscopic simulations. This setting is the most straight-forward way to map pairs of macroscopic strains and stresses but does not offer good generalization properties once path-dependency is introduced. In that case, one way to overcome the lack of history information is to extend their feature space with *e.g.* previous (incremental) strains and/or stresses [4,5].

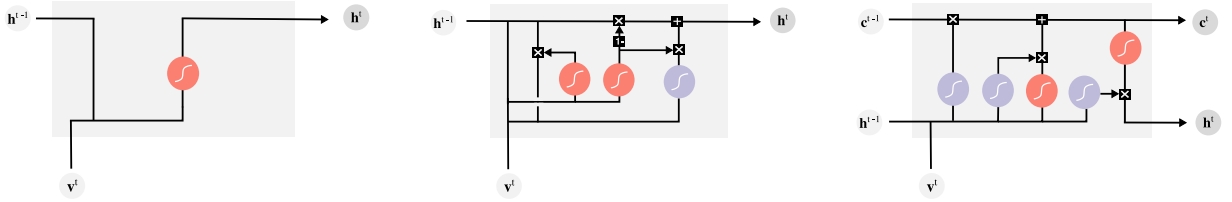
As an alternative, RNNs offer additional parameters (*i.e.* the hidden state) and mechanisms (*i.e.* the gates that control the flow of information being propagated) to learn history information from sequential data in an implicit way. These parameters describe the evolution of the so-called hidden state and can encapsulate information from previous iterations without the need to introduce history variables in the feature space. In a regular RNN, the outputs and hidden state are given by:

$$\begin{aligned} \mathbf{h}^t &= \phi(\mathbf{W}_1\mathbf{v}^t + \mathbf{W}_s\mathbf{h}^{t-1} + \mathbf{b}_s) \\ \widehat{\boldsymbol{\sigma}}^t &= \phi(\mathbf{W}_2\mathbf{h}^t + \mathbf{b}_2) \end{aligned} \tag{15}$$

where $\phi(\cdot)$ is an activation function, $\mathbf{W}_s$ and $\mathbf{b}_s$ are the additional model parameters (compared to conventional feed-forward neural networks), $\mathbf{v}^t$ are the current neuron values coming from the last layer and $\mathbf{h}^t$ and $\mathbf{h}^{t-1}$ are current and previous states, respectively. This arrangement allows the network to learn how stress evolves for a sequence of strains instead of building a regression model from independent stress–strain pairs and is illustrated in Fig. 2(a). However, in practice, the efficiency of RNNs are impeded by vanishing gradient problems and are not suitable for long-term history dependent problems.

To overcome that, more sophisticated architectures (more popularly known as cells) have been proposed. Among the most popular ones are the Gated Recurrent Unit (GRU) and the Long-Short Term Memory (LSTM), illustrated in Figs. 2(b) and 2(c), respectively. The internal mechanisms, also known as gates, used to control the flow of information passing from one state to another are represented by the colored circles. For each gate, additional parameters need to be learned by the network in a way that the element-wise application of the sigmoid (red circles) or tanh (purple circles) functions can wisely retain what should be preserved and what can be forgotten in a long sequence.

Despite best efforts, these architectures are still vulnerable to overfitting, compromising their ability to generalize well to new data. Potential solutions to prevent this phenomenon include regularization techniques such as L2 penalty, early stopping and dropout. In this work, a special type of dropout proposed by Kingma et al. [32] is used in combination with a GRU architecture is considered to perform the comparison with the proposed network. In this Bayesian GRU, the regular dropout with continuous noise (*i.e.* Gaussian dropout) is reinterpreted as a variational method that allows optimal dropout rates to be inferred from the data as opposed to it being fixed and defined in advance as usual. This circumvents the need for a validation set during model selection. For more details, the interested reader is referred to [32].



(a) Classical Recurrent cell (b) Gated Recurrent Unit (GRU) cell (c) Long-Short Term Memory (LSTM) cell

Fig. 2. Different architectures for recurrent cells: red circles correspond to the element-wise application of the sigmoid function, while purple circles correspond to the tanh function. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

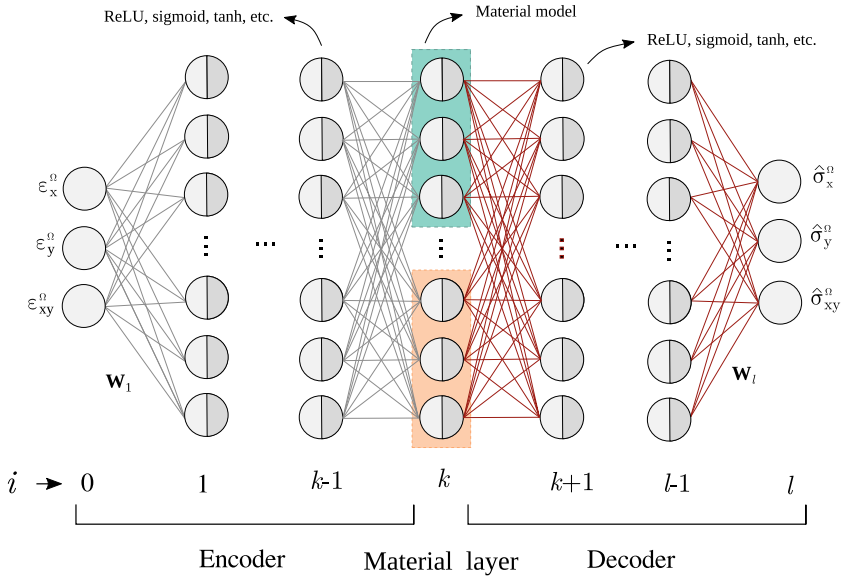


Fig. 3. The proposed architecture.

4. Physically recurrent neural network

This section presents the neural network proposed to capture path-dependent behavior of heterogeneous microscopic models. The core task of the network is to learn how the macroscopic strain $\boldsymbol{\varepsilon}^{\Omega}$ can be dehomogenized into a small set of representative material points and how their responses can be combined to obtain the homogenized macroscopic stresses $\boldsymbol{\sigma}^{\Omega}$. For this, the parametric regression model $\mathcal{R}$ illustrated in Fig. 3 is proposed: a combination of a data-driven encoder, a material layer with embedded physics-based material models and a data-driven decoder. Each of these components are discussed in detail in Sections 4.1, 4.2 and 4.3, respectively. Finally, the training process is described in Section 4.4 and the use of this network as constitutive model in FE^2 frameworks is discussed in Section 4.5.

4.1. Encoder

The encoder consists of all parameters that convert the macroscopic strain from the input layer to the values used as input of the material layer, which corresponds to the gray lines in Fig. 3. Since these values are the inputs of actual material models that are later combined by the decoder into the prediction of the macroscopic stresses, we interpret the role of the encoder as being the microscopic periodic boundary-value problem (BVP) solved with FE, only at a much lower computational cost. On the other hand, no information on the displacement field of the micromodel is retrieved by the network as the encoder is learned based exclusively on snapshots of macroscopic

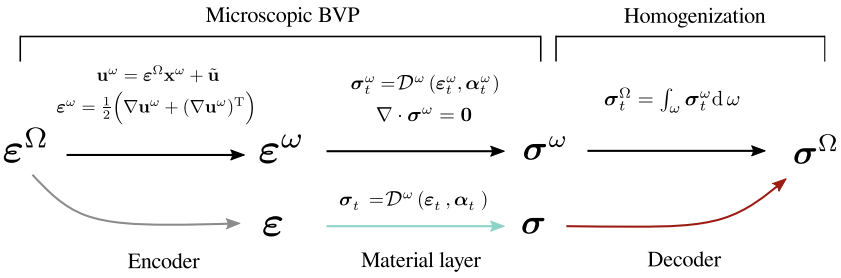


Fig. 4. Interpretation of the proposed network with respect to a full-order solution.

stresses. This understanding is depicted in Fig. 4 by the gray curved line linking the strains from the macroscopic scale and the fictitious strains seen by the material points in the network.

As for the architecture, an arbitrary number of layers and units (with conventional activation functions as illustrated in Fig. 3) can be used. In case regular dense layers are employed, the neuron states ($\mathbf{a}_{i-1}$) from the previous layer $i-1$ are propagated to the following layer i according to:

$$\mathbf{v}_i = \mathbf{W}_i \mathbf{a}_{i-1} + \mathbf{b}_i \qquad \mathbf{a}_i = \phi(\mathbf{v}_i) \tag{16}$$

where $\mathbf{W}_i \in \mathbb{R}^{n_i \times n_{i-1}}$ is a weight matrix and $\mathbf{b}$ is a bias term with n_i being the number of neurons of layer i, and ϕ is an activation function applied in an element-wise manner to the neuron values of i to introduce nonlinearity into the network. In the particular case where the dense layer is either the input or the output layers, no activation function is applied and $\mathbf{v}_0$ is set to $\boldsymbol{\varepsilon}^\Omega$. This results in $\mathbf{a}_0 = \mathbf{v}_0 = \boldsymbol{\varepsilon}^\Omega$. Popular activation functions include the sigmoid, tanh and ReLU. In the present investigation, the architecture of the network consists in three layers: input, material and output layers. This setting results in a linear relationship between macroscopic strains and local strains that are used as input for the material layer.

It is worth stressing that this architecture does not yield path-dependent local strain paths, in contrast to the actual RVE where the strain distribution will generally be path-dependent. However, the homogenized response is path-dependent, through the history variables in the material layer. The effect of introducing path-dependency to the encoder is discussed in Appendix A.

4.2. Material layer

The material layer is responsible for introducing explicitly the same physics-based material models used in the RVE that the network will be a surrogate for. To properly incorporate them and take full advantage of its outputs, important changes on how neurons are evaluated compared to regular dense layers are proposed. First, instead of introducing nonlinearity in a element-wise manner with a scalar-to-scalar activation function, neurons are grouped in m sets of the size of the input layer (see colored boxes in Fig. 3) and then evaluated as a subgroup. Each subgroup is referred to as a *fictitious material point* and its size is equal to the length of the strain vector (*i.e.* length 3 for the present investigation in two dimensions). In this arrangement, each neuron of the subgroup j represents one component of the strain vector $\boldsymbol{\varepsilon}_j$, as illustrated in Fig. 5(a).

Supposing the micromodel contains n material models $\mathcal{D}_1^\omega, \ldots \mathcal{D}_n^\omega$, several combinations of them can be employed in the material layer. The choice on which material models should be used to evaluate the fictitious material points depends on the types of non-linearity embedded in each of these models. For simplicity, we choose to illustrate the network with a general material model $\mathcal{D}^\omega$ for all fictitious material points. Such model can take the form of any of the n material models $\mathcal{D}_i^\omega$ with known material properties used in the micromodel. Here, $\mathcal{D}^\omega$ takes as input the current strain $\boldsymbol{\varepsilon}^t \in \mathbb{R}^{n_\varepsilon}$ and the internal variables from previous time step $\boldsymbol{\alpha}^{t-1} \in \mathbb{R}^{n_{IntVar}}$, where n_{IntVar} is the number of internal variables of the material model. With that, the model is used to evaluate current stress state $\boldsymbol{\sigma}^t \in \mathbb{R}^{n_\epsilon}$ and updated internal variables $\boldsymbol{\alpha}^t$. These quantities motivated the tailor-made architecture of the proposed layer.

To store the internal variables used as input/output of the material model, an auxiliary vector $\mathbf{h}_j \in \mathbb{R}^{n_{IntVar}}$ referred as history vector is defined. In the particular case of a subgroup with a material model with no internal

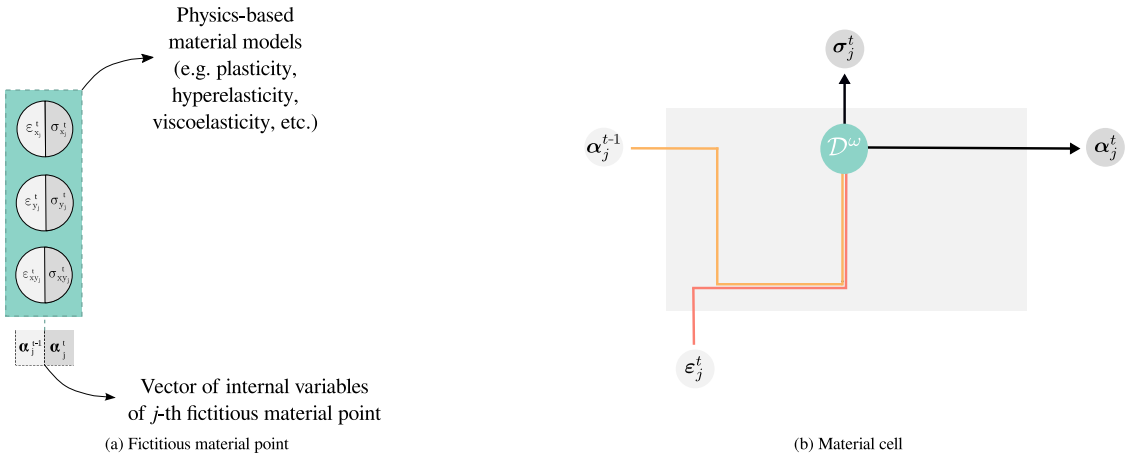


Fig. 5. Schemes of (a) fictitious material point and (b) material layer as a cell.

variables (*e.g.* linear elastic model), $\mathbf{h}_j$ does not exist since $n_{IntVar} = 0$. For the first time step, the history vector is initialized as zero for all m subgroups. As information reaches the material layer and the material model is called, three outputs are made available: the stresses $\boldsymbol{\sigma}_j^t$, the updated internal variables $\boldsymbol{\alpha}_j^t$ and the tangent stiffness matrix $\mathbf{D}_j^t \in \mathbb{R}^{n_\varepsilon \times n_\varepsilon}$. In this layer, only the stresses are propagated forward. To do this, each stress component is associated to a unit of the subgroup, as illustrated in Fig. 5(a). Then, the updated internal variables $\boldsymbol{\alpha}^t$ are stored in $\mathbf{h}_j^t$ so that when new strains $\boldsymbol{\varepsilon}_j^{t+1}$ are fed to the fictitious material point, the material model is aware of its own history so far, making the $\boldsymbol{\varepsilon}$-$\boldsymbol{\sigma}$ path of each subgroup unique. This architecture is illustrated in Fig. 5(b). For the sake of notation clarity, from now on we omit the time index t when referring to current values.

Note that $\mathbf{h}_j$ is not learned through a set of parameters, but obtained as an automatic output of the (path-dependent) material model employed in subgroup j. This works as the physical memory of the network as it stores the history variables that describe a given fictitious material point. Using internal variables obtained directly from the material models is where the proposed approach crucially differs from purely data-driven RNNs. This is further discussed in Section 4.6, where we assess how this network compares to other methods in the literature. It is also worth mentioning that since no data from the microscale has been collected and imposed in the network, the paths seen by the fictitious material points do not need to hold any similarity with actual integration points of the microscopic model.

Using standard machine learning notation, the material layer propagates previous neuron states ($\mathbf{a}_{k-1}$) and applies the material model $\mathcal{D}^\omega$ as follows:

$$\mathbf{v}_k = \mathbf{W}_k \mathbf{a}_{k-1} + \mathbf{b}_k \Rightarrow \mathbf{a}_k, \mathbf{h} = \mathcal{D}^\omega(\mathbf{v}_k, \mathbf{h}^{t-1}) \tag{17}$$

where $\mathbf{W}_k \in \mathbb{R}^{n_k \times n_{k-1}}$ is the weight matrix connecting layers $k-1$ and k, $\mathbf{b}_k \in \mathbb{R}^{n_k}$ is a bias term. In addition, $\mathbf{v}_k$ are the neuron values (correspond to the concatenated vector of all microscopic strains $\boldsymbol{\varepsilon}_j$), $\mathbf{h}^{t-1}$ and $\mathbf{h}$ are history-related term (correspond to concatenated vector of all internal variables $\boldsymbol{\alpha}_j^{t-1}$) resulting from the material models with path-dependent behavior from past and current time step and $\mathbf{a}_k$ are the current neuron states (correspond to the concatenated vector of all microscopic stresses $\boldsymbol{\sigma}_j$).

4.2.1. Choice of constitutive model

A general guideline on how to select the constitutive models used for evaluating the fictitious material points is to employ all different sources of nonlinearity with their respective known material properties in the material layer. To illustrate that, consider the micromodel used in Sections 6 and 7, a composite microstructure with two material models: a linear elastic model $\mathcal{D}_1^\omega$ to describe the fibers and an elastoplastic model with isotropic hardening $\mathcal{D}_2^\omega$ to describe the matrix. The latter starts as linear-elastic and evolves into the plastic regime once the yield stress is reached. Motivated by that, only model $\mathcal{D}_2^\omega$ is employed in all fictitious material points since the network still can make any of the subgroups to behave linear elastically by passing small strains to the material model and

subsequently scaling the stresses to give a significant elastic contribution through the decoder. This is illustrated in Section 7.1 in a numerical example, where the response of all fictitious material points are shown for a single macroscopic point.

However, if instead of a linear elastic model, a nonlinear elastic model was used to describe the fibers, the network would not perform optimally. In that case, although the elastoplastic model does introduce nonlinearity to the network, the (nonlinear) contribution from the fibers is no longer embedded in that model. Furthermore, if the nonlinear elastic model was the one chosen to evaluate all subgroups, the network would essentially become a feed-forward one with no history information taken into account (explicitly or implicitly), losing the ability to predict elastic unloading. For such a micromodel, both material models would need to be considered.

Another interesting case is that of a micromodel with two elastoplastic phases with different material properties. This time, depending on the contrast of the material properties, a single material model with a fixed set of properties coming from one of the two phases might be enough to reproduce the homogenized response of the micromodel. While both cases are illustrated in Section 8, naturally, far more complex arrangements than the ones discussed here are found in practice. This is also true for the potential extensions to the current approach. In the last scenario, for instance, making the material properties of each fictitious material point a trainable feature might be advantageous. This could also be a valuable feature when dealing with experimental data or with a micromodel with continuously varying material properties. Addressing these extensions is object of ongoing research.

4.3. Decoder

The decoder consists of all parameters that convert the outputs from the material layer to the predicted macroscopic stress $\widehat{\boldsymbol{\sigma}}^{\Omega}$ in the output layer, which corresponds to the brown lines in Fig. 3. Similar to the encoder, an arbitrary number of conventional layers and units can be employed. In the full-order solution, after convergence of the microscopic BVP, the macroscopic stresses are obtained by the volume average of microscopic stresses over the entire RVE. In the network, since the solution of the microscopic BVP is replaced by the encoder and the microscopic material points in the RVE are replaced by the few fictitious material points, the decoder is then analogous to the homogenization operator that transforms local stresses to macroscopic stresses, as illustrated by the brown curved line in Fig. 4.

In the present work, we use a single dense layer (output) with linear activation and physics-motivated modifications to perform the task. With this, all the nonlinearity of the network arises from the models in the material layer. As discussed previously, the decoder can be understood as the averaging operator in a multiscale approach and with the chosen architecture (dense-material-dense), the weights of the output layer can be seen as the relative contribution of each fictitious material point to the macroscopic stress. Based on that, a constraint on the positivity of the weights of the output layer is considered. For that, a softplus function $\rho(\cdot)$ is applied element-wise on the weights matrix before computing the neuron values of the last layer:

$$\mathbf{v}_l = \rho(\mathbf{W}_l)\ \mathbf{a}_{l-1} + \mathbf{b}_l \tag{18}$$

where $\mathbf{b}_l$ is set to zero and $\mathbf{a}_{l-1}$ corresponds to the stresses coming from the material layer. This procedure guarantees that, after the transformation, weights will always be positive.

4.4. Training

The goal of the training phase is to minimize a loss function given by:

$$L = \frac{1}{N}\ \sum_{i=1}^{N} \frac{1}{2}\ \| \ \boldsymbol{\sigma}^{\Omega}(\boldsymbol{\varepsilon}_i^{\Omega}) - \widehat{\boldsymbol{\sigma}}^{\Omega}(\boldsymbol{\varepsilon}_i^{\Omega}) \ \|^2 \tag{19}$$

where N is the number of snapshots. Based on it, a Stochastic Gradient Descent (SGD) optimization algorithm is used to update the trainable parameters $\mathbf{W}$ and $\mathbf{b}$:

$$\begin{aligned} \mathbf{W}^n &= \mathbf{W}^o - \mathcal{A}\Big(\frac{1}{B}\sum_{i=1}^{B}\frac{\partial L_i}{\partial \mathbf{W}}\Big) \\ \mathbf{b}^n &= \mathbf{b}^o - \mathcal{A}\Big(\frac{1}{B}\sum_{i=1}^{B}\frac{\partial L_i}{\partial \mathbf{b}}\Big) \end{aligned} \tag{20}$$

where L_i is the loss of the ith sample, o indicates current values, n indicates updated values and B is the size of the sample mini-batch used in the update. Finally the $\mathcal{A}$ operator depends on the solver. In this work, the Adam optimizer [33] is used.

To compute the gradients appearing in Eq. (20), backpropagation in time is employed in a similar fashion as done to RNNs: based on the network state ($\mathbf{v}$ and $\mathbf{a}$) after computing each training curve with n pairs of $\boldsymbol{\sigma} - \boldsymbol{\varepsilon}$, the chain rule is used to propagate the derivative of the loss functions starting from the output layer and progressively moving back through the network and through time. Commonly, this process is dealt with by automatic differentiation, but we present the expressions to allow for integrating the network directly into existing FE software. For this, two auxiliary quantities are defined. The first is defined for each layer and helps propagating the error through the network $\mathbf{d}_i \in \mathbb{R}^{n_i}$. Starting from the output layer l, it is defined as:

$$\mathbf{d}_l = \frac{\partial L}{\partial \mathbf{a}_l} = \widehat{\boldsymbol{\sigma}}^{\Omega} - \boldsymbol{\sigma}^{\Omega} \tag{21}$$

Next, the effect of the activation function is taken into account as:

$$\bar{\mathbf{d}}_i = \mathbf{d}_i \odot \frac{\partial \phi(\mathbf{v}_i)}{\partial \mathbf{v}} \tag{22}$$

where $\odot$ represents the Hadamard product. After that, it is possible to compute the gradients of the trainable parameters:

$$\frac{\partial L}{\partial \mathbf{W}_i} = \bar{\mathbf{d}}_i \mathbf{a}_{i-1}^{\mathrm{T}} \qquad \frac{\partial L}{\partial \mathbf{b}_i} = \bar{\mathbf{d}}_i \tag{23}$$

Finally, the values $\mathbf{d}$ of the previous layer (the next layer to be backpropagated) can be computed as:

$$\mathbf{d}_{i-1} = \mathbf{W}_i^{\mathrm{T}} \bar{\mathbf{d}}_i \tag{24}$$

and the algorithm moves to Eq. (22) for layer $i-1$.

When reaching the material layer, recall that the internal variables are stored in $\mathbf{h}$ and used for keeping track of the evolution of the internal variable through time. For that reason, a second auxiliary quantity is introduced and Eq. (22) is replaced by:

$$\bar{\mathbf{d}}_i = \mathbf{d}_i \odot \frac{\partial \mathbf{a}_i}{\partial \mathbf{v}_i} + \mathbf{d}_h^{t+1} \odot \frac{\partial \mathbf{h}}{\partial \mathbf{v}_i} \tag{25}$$

where the first term concerns the derivatives of stresses with respect to strains, the second term concerns the derivatives of the current internal variables with respect to strains and $\mathbf{d}_h \in \mathbb{R}^{n_i}$ is given by:

$$\mathbf{d}_h = \mathbf{d}_i \odot \frac{\partial \mathbf{a}_i}{\partial \mathbf{h}} + \mathbf{d}_h^{t+1} \odot \frac{\partial \mathbf{h}}{\partial \mathbf{h}^{t-1}} \tag{26}$$

Note that the derivatives of the stresses with respect to the strains of material point j are an output of the material model: the tangent stiffness matrix $\mathbf{D}_j$. The remaining derivatives in Eqs. (25) and (26) are evaluated using central finite differences. Naturally, computing gradients with other methods would also be possible. For instance, if the material model used in the network supports automatic differentiation, storing the internal variables in $\mathbf{h}$ for backpropagation can be bypassed as the derivatives are automatically obtained in this approach.

Finally, to obtain the gradients of the trainable parameters including the history-dependence coming from the material layer and compute the values $\mathbf{d}$ of the previous layer, we consider Eq. (25) instead of (22) in the expressions shown in Eqs. (23) and (24), respectively.

4.5. Use as constitutive model

To make new stress predictions, the macroscopic strain $\boldsymbol{\varepsilon}^{\Omega}$ is fed to the input layer and a complete forward pass is performed. The final activated neuron values of the output layer give the predicted stress. To obtain the macroscopic consistent tangent stiffness matrix $\mathbf{D}^{\Omega}$, a complete backward pass is required:

$$\mathbf{D}^{\Omega} = \frac{\partial \widehat{\boldsymbol{\sigma}}^{\Omega}}{\partial \boldsymbol{\varepsilon}^{\Omega}} = \frac{\partial \mathbf{a}_l}{\partial \mathbf{v}_0} = \mathbf{J} \tag{27}$$

which is obtained with a backward pass through the network:

$$\mathbf{J}_i = \mathbf{J}_{i+1}\mathbf{I}_i^{\phi}\mathbf{W}_i \quad \text{with} \quad \mathbf{J}_{l+1} = \mathbf{I} \tag{28}$$

where $\mathbf{I}_i^{\phi}$ is a matrix whose diagonal contains the derivatives of the activation function with respect to the neuron values $\mathbf{v}$:

$$\mathbf{I}_i^{\phi} = \text{diag}(\frac{\partial \phi(\mathbf{v}_i)}{\partial v}) \tag{29}$$

except for the material layer. In that case, such matrix is full and consists of the concatenation of the tangent stiffness matrix of all fictitious material points. It is worth mentioning that despite the linear dependency on the tangent stiffness matrices of the material models, the Jacobian matrix of the network does not inherit their spectral properties.

4.6. Analogies to other methods

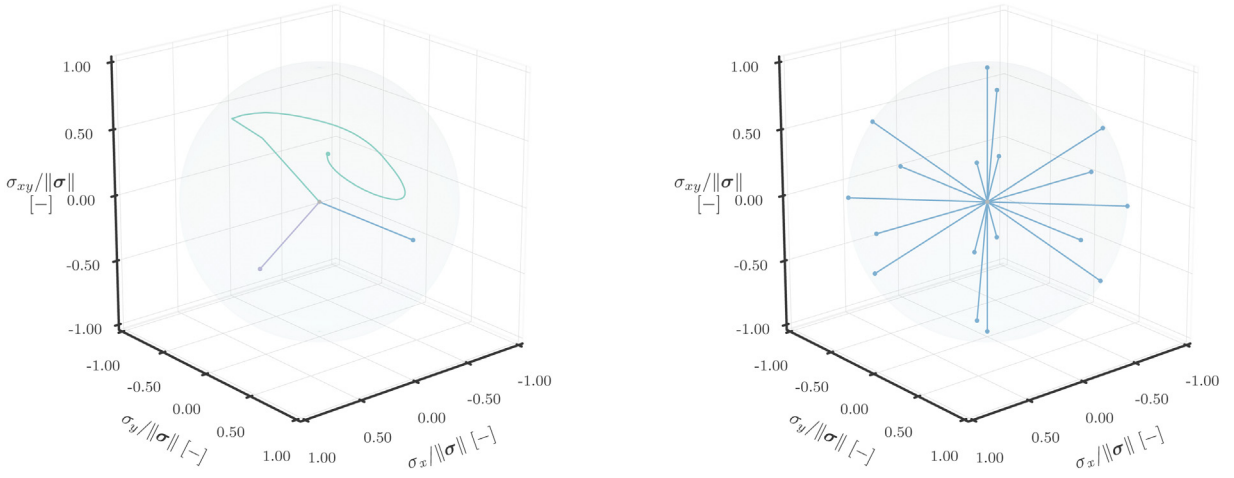
In this section, the parallels between features of the proposed network and related works in the literature are briefly discussed. One possible analogy comes from hyper-reduced-order models [34]. With the architecture chosen for the present investigation, both methods work on a reduced number of material points with modified (integration) weights. However, in the network, these points are only fictitious and learned by the encoder based on snapshots of the homogenized stresses. Moreover, each stress component is associated with a different weight. By contrast, the material points in the hyper-reduction approach exist in the microscopic model and a single modified integration weight of each material point selected is used to compute all its stress/internal force components.

Following the discussion on the encoder, it is worth highlighting how this feature would be framed with respect to asymptotic homogenization schemes such as Mori–Tanaka [35]. In this type of solution, the microscopic problem is also not solved explicitly and only average fields are calculated. Relying on the equivalent inclusion idea and on Eshelby's solution [36], the strain concentration tensor is obtained analytically and yields the full solution of the microscopic model as it correlates the average field of the phases in the micromodel with its average field. In our network, although the macroscopic stresses are also obtained by relating macroscopic and (fictitious) microscopic strains through an encoder, here no average field is calculated for each of the phases. Indeed, not every phase needs to be included in the material layer and multiple strain paths for the same phase are considered. Furthermore, while Mori–Tanaka is accurate for moderate volume fractions of the inclusions, such restriction is not present in our method.

Compared to PINNs, in which physical constraints are explicitly included in the loss function, here, most physical constraints are naturally taken care of by the physics-based material models directly embedded in the material layer. The proposed approach is also more general as it can be directly used for arbitrary material models and is not particularly tailored to a single type of model (*e.g.* elastoplastic behavior [14,15]). As an added benefit, our model selection procedure makes physical sense: we add more material points or material models to the network.

Another noteworthy strategy with relevant analogies to our method is the DMN [20]. In this approach, the contribution of a few material points evaluated using the classical constitutive models in the RVE is also employed to make predictions in the online phase. On the same reasoning as discussed in Section 4.2, since the inputs come directly from actual material models, path-dependency is captured naturally. However, the main concept and architecture of DMNs are different from the ones explored here. In the offline phase, the goal of the DMN is to find a topological representation of the RVE with fewer degrees of freedom (*i.e.* material points) based only on the elastic stiffness matrices of the different material phases that compose the original micromodel. For the online phase, the feature space is increased to include residual stresses of the micromodel components, and an iterative procedure is implemented. The authors compare the incremental strains of the material points at the beginning of the iteration with the one obtained by a de-homogenization process that backpropagates the macroscopic incremental strain from the output layer to the bottom layer (*i.e.* input layer). Upon convergence, the set of internal variables of each material point at the bottom layer is therefore updated.

In the present work, the feature space is the same in both phases and no iterative procedure is employed in the forward pass, which simplifies implementation and reduces even further the number of material model calls. Here, the strain path each fictitious material point follows is simply described by the encoder and not all phases need to be included in the network. The homogenized stresses and tangent stiffness are obtained in a single forward



(a) Types of loading used in this study

(b) Proportional loading with *a priori* known directions

Fig. 6. Different design of experiments strategies. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and backward pass, respectively. Furthermore, the backpropagation in our approach is considerably simpler than the DMN. Although the use of homogenization (and de-homogenization) operations in the DMN assigns physical interpretation to the model, it also makes training a rather intricate process.

Finally, to draw a parallel with LSTMs, one might understand **h** as the cell state **c**, but instead of using bijective and smooth functions such as the sigmoid and tanh functions to describe the evolution of the material response, the material model itself is directly employed. This bypasses the need to learn new parameters to regulate the flow of information kept or forgotten throughout time (see Fig. 2(b)) and has important implications for the training process. The most important one is the ability to mirror physical behaviors such as elastic unloading/reloading without ever seeing the pattern during training, a stark contrast with LSTMs and GRUs that usually require extensive training sets with multiple cycles of loading and reloading at different strain levels with different step sizes. The physical interpretation of the nonlinearity is directly embedded in the network. In the numerical examples of this work, the nonlinearity is due to plasticity, but other effects such as hyperelasticity, visco-plasticity, stiffness degradation, or any combination thereof, could be embedded by adapting the constitutive model that is used in the material layer.

5. Design of experiments

One critical aspect of the training and testing of surrogate models is the formulation of a sampling plan. Typically, a uniform distribution of the sampling points is desirable, but that task becomes more complex when path-dependent behavior is present. In this case, pairs of strains and stresses are collected and processed as sequences, which leads to potentially infinite-dimensional parameter spaces.

In this work, two strategies are considered. In the first approach, proportional loading paths are generated, which means that the stress ratio between the components is constant. Here, the sequence of strains is created based on two features: the loading function $\lambda(\Delta\varepsilon, t)$ and the loading direction given by the unit vector $\mathbf{n}$, where $\Delta\varepsilon$ is the step size and t is the current time step. For each time step, $\mathbf{n}$ is multiplied by the scalar-valued loading function λ creating a new set of strains, which is in turn applied at the controlling nodes of the microscopic model.

For monotonic loading, the loading function is as depicted in Fig. 7(a). The values in the unit vector can come from prior knowledge of the material as illustrated in Fig. 6(b), in which only fundamental cases such as uniaxial strain, pure shear, and biaxial cases are considered, or from random distributions as represented by the purple line Fig. 6(a). In the present work, the random directions are obtained by sampling values from n_ε independent Gaussian distributions ($\mathrm{X} \sim \mathcal{N}(0, 1)$) and subsequently normalizing the vectors.

Despite the simplicity in creating such paths, RNNs trained exclusively on monotonic cannot predict cyclic responses. Thus, to create non-monotonic sequences, a linear piecewise function as the one depicted in Fig. 7(b) is

used. Note that even though the loading function is changed, the unit vector is kept constant for the entire strain sequence, yielding proportional loading. However, to cover the entire space of possible cyclic responses, a large (and *a priori* unknown) number of curves comprehending different unloading points with different duration of unloading/reloading and step sizes is necessary. In this work, that matter is first handled in a simplified way by only sampling two different cycles of unloading/reloading.

Finally, in a more general approach, a second strategy to create the $\boldsymbol{\varepsilon}$-$\boldsymbol{\sigma}$ paths is considered: the *random walks*. These are typically defined by sampling random strain increments with random loading directions for each time step, resulting in non-proportional loading. In this work, random walks are created by associating the prescribed strains to independent Gaussian Processes (GPs) with $\mathrm{X} \sim \mathcal{N}(\mu,\, \sigma^2)$ and covariance function given by:

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp\Big(-\frac{1}{2\ell^2}\|\mathbf{x}_p - \mathbf{x}_q\|^2\Big) \tag{30}$$

where $\mathbf{x}_p$ and $\mathbf{x}_q$ are the time step indices of the strain sequence being sampled, σ_f^2 is the variance and ℓ is a lengthscale. In this setting, the lengthscale controls the smoothness of the strain path and the variance controls how large the step size can be for each prescribed degree of freedom of the controlling nodes. Similar approaches were employed by Mozaffar et al. [8] and Logarzo et al. [11].

Algorithm 1: Generation of random loading path using GPs

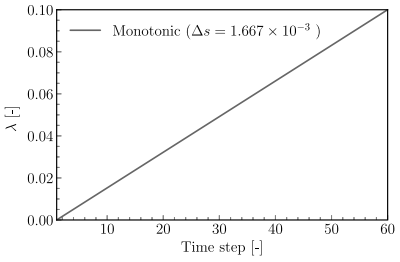
Input : lengthscale ℓ, variance σ_f^2, number of strain components $N_{\text{components}}$, number of time steps N_{steps}
Output: macroscopic strains $\mathfrak{D}_\varepsilon$ and macroscopic stresses $\mathfrak{D}_\sigma$

initialize datasets: $\mathfrak{D}_\varepsilon \leftarrow \varnothing$, $\mathfrak{D}_\sigma \leftarrow \varnothing$
for $i \in [1, 2, ..., N_{components}]$ **do**
  initialize input and output datasets for GP_i: $\mathbf{X}_{\mathrm{GP}_i} \leftarrow 0$, $\mathbf{Y}_{\mathrm{GP}_i} \leftarrow 0$
  initialize GP_i: $\mathrm{GP}_i \leftarrow$ `initGP(` $\mathbf{X}_{\mathrm{GP}_i}, \mathbf{Y}_{\mathrm{GP}_i}, \ell, \sigma_f^2$)
for $t \in [1, 2, ..., N_{steps}]$ **do**
  initialize current macroscopic strain: $\boldsymbol{\varepsilon}_{\text{current}} \leftarrow \varnothing$
  for $i \in [1, 2, ..., N_{components}]$ **do**
    sample from posterior distribution: $\varepsilon_i \leftarrow \mathrm{GP}_i$`::samplePosterior (` t `)`
    add value to strain vector: $\boldsymbol{\varepsilon}_{\text{current}} \leftarrow \boldsymbol{\varepsilon}_{\text{current}} \cup \varepsilon_i$
  solve micromechanical BVP: $\boldsymbol{\sigma}_{\text{current}} \leftarrow$ `fullModel::materialUpdate (` $\boldsymbol{\varepsilon}_{\text{current}}$ `)`
  if *convergence* **then**
    store equilibrium solution of micromodel: `fullModel::storeSolution( )`
    add macroscopic strains and stresses to dataset: $\mathfrak{D}_\varepsilon \leftarrow \mathfrak{D}_\varepsilon \cup \boldsymbol{\varepsilon}_{\text{current}}$, $\mathfrak{D}_\sigma \leftarrow \mathfrak{D}_\sigma \cup \boldsymbol{\sigma}_{\text{current}}$
    for $i \in [1, 2, ..., N_{components}]$ **do**
      add time step and current strain to dataset of GP_i: $\mathbf{X}_{\mathrm{GP}_i} \leftarrow \mathbf{X}_{\mathrm{GP}_i} \cup t$, $\mathbf{Y}_{\mathrm{GP}_i} \leftarrow \mathbf{Y}_{\mathrm{GP}_i} \cup \varepsilon_i$
      update GP_i with new data: GP_i`::update (` $\mathbf{X}_{\mathrm{GP}_i}, \mathbf{Y}_{\mathrm{GP}_i}$ `)`
return $(\mathfrak{D}_\varepsilon, \mathfrak{D}_\sigma)$

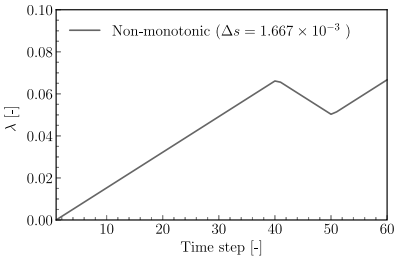
The details of the present implementation are given in Algorithm 1. Note that instead of drawing the entire strain sequence for a given component, we sample it step by step and update the GP dataset before sampling again. This strategy results in the same strain sequence given a fixed random seed throughout the steps, but in this way the GPs can also be used in applications where the number of loading steps is changed on-the-fly. Following the work of Logarzo et al. [11], we define the mean of all GPs to be zero and include $t = 0$ and $\varepsilon_i = 0$ as a prior. In addition to that, references to *fullModel* (*i.e.* the full-order microscopic model) in Algorithm 1 are kept as minimal and general as possible. One example of loading path resulting from Algorithm 1 is illustrated in Fig. 6(a).

In this paper, both strategies generate $\boldsymbol{\varepsilon}$-$\boldsymbol{\sigma}$ curves containing 60 time steps, unless stated otherwise. To summarize the types of loading studied in the following sections:

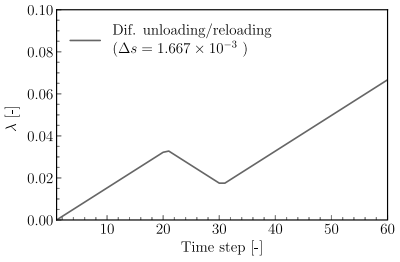
- Type I: monotonic and proportional loading paths with *a priori* known directions. The 18 directions used to train the proposed network are illustrated in Fig. 6(b) and include uniaxial strains, pure shear, biaxial cases and biaxial with shear cases.
- Type II: monotonic and proportional loading paths randomly spread across the design space. The loading directions are generated randomly and the loading function is as shown in Fig. 7(a).



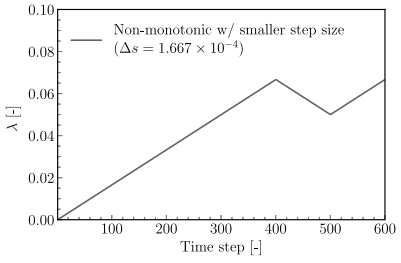
(a) Monotonic loading (Types I and II)



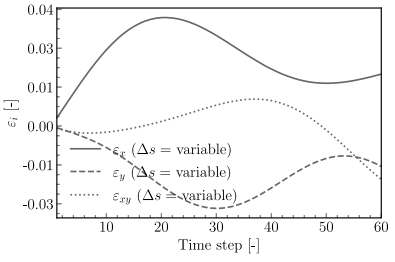
(b) Non-monotonic loading (Type III)



(c) Different unloading/reloading (Type IVa)



(d) Non-monotonic with smaller step size (Type IVb)



(e) Random non-proportional and non-monotonic with $\ell = 20$ (Type V)

Fig. 7. Proportional and non-proportional loading functions.

- Type III: non-monotonic and proportional loading paths randomly spread across the design space. Again, the loading directions are random, but the loading function is now given by Fig. 7(b) and includes one cycle of unloading;
- Type IV: Variations to Type III:
  - Type IVa: same loading directions as the test set of Type III, but unloading/reloading takes place at a different point in time as shown in Fig. 7(c);
  - Type IVb: same loading directions as the test set of Type III, but time step is 10 × smaller. Thus, to reach the same norm as the original curve in Type II, 600 time steps are evaluated, as depicted in Fig. 7(d);
- Type V: non-monotonic and non-proportional loading paths randomly spread across the design space. A GP-based path described by Eq. (30) is illustrated in Fig. 6(a). Fig. 7(e) illustrates the strain paths of each component using this approach with lengthscale $\ell = 20$ and $\sigma_f = 1.0 \times 10^{-3}$.

6. Assessing network performance

In this section, the performance of the proposed network is compared to a state-of-the-art RNN trained on different training dataset sizes and methods to sample the design space. The comparison is done for a single micromodel. Specifically, four scenarios are investigated: (i) predicting unloading/reloading behavior from monotonic data, (ii) predicting unloading/reloading behavior from non-monotonic data, (iii) predicting unseen patterns from non-monotonic data, and (iv) training with non-monotonic and non-proportional loading paths. In the first three scenarios, our network is trained exclusively on the fundamental loading cases of Type I (18 curves), while the training of the RNN is an open question to be addressed in the following sections.

From now on, the network presented in this work will be referred to as Physically Recurrent Neural Network, or simply PRNN. The PRNN was trained for 80 000 epochs, while the RNN was trained for 60 000 epochs with an early stopping criterion, which consists of interrupting training if the best training loss so far is not improved over a given period (in this work, 5000 epochs). The Adam optimizer is used in all cases with batch size of 9 and default parameters suggested by Kingma and Ba [33], with the exception of the learning rate of 0.01 for the RNNs. The

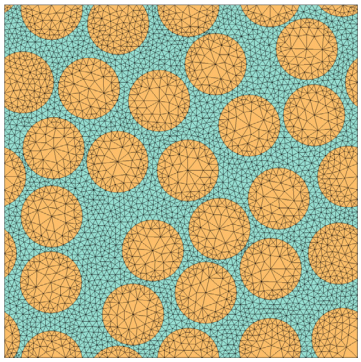


Fig. 8. Geometry and mesh discretization of microscopic model adopted in this work.

layer sizes are chosen through model selection to provide optimal results and fair comparison with our approach to the best of our knowledge. The methodology adopted is briefly described in Section 6.1.

The microscopic model consists of an RVE with 36 elastic fibers (volume fraction = 0.6) with properties E = 74 000 MPa and ν = 0.2 embedded in an elastoplastic matrix with isotropic hardening. The geometry and the mesh with 7048 elements are depicted in Fig. 8. The elastoplastic matrix is modeled using the von Mises yield criterion with properties E = 3130 MPa, ν = 0.3 and yield stress given by:

$$\sigma_y = 64.8 - 33.6\exp^{-\varepsilon^p_{eq}/0.0003407} \tag{31}$$

where ε^p_{eq} is the equivalent plastic strain defined as:

$$\varepsilon^p_{eq} = \sqrt{\frac{2}{3}\,\boldsymbol{\varepsilon}^p : \boldsymbol{\varepsilon}^p} \tag{32}$$

and $\boldsymbol{\varepsilon}^p$ is the plastic strain. Plane stress conditions are assumed.

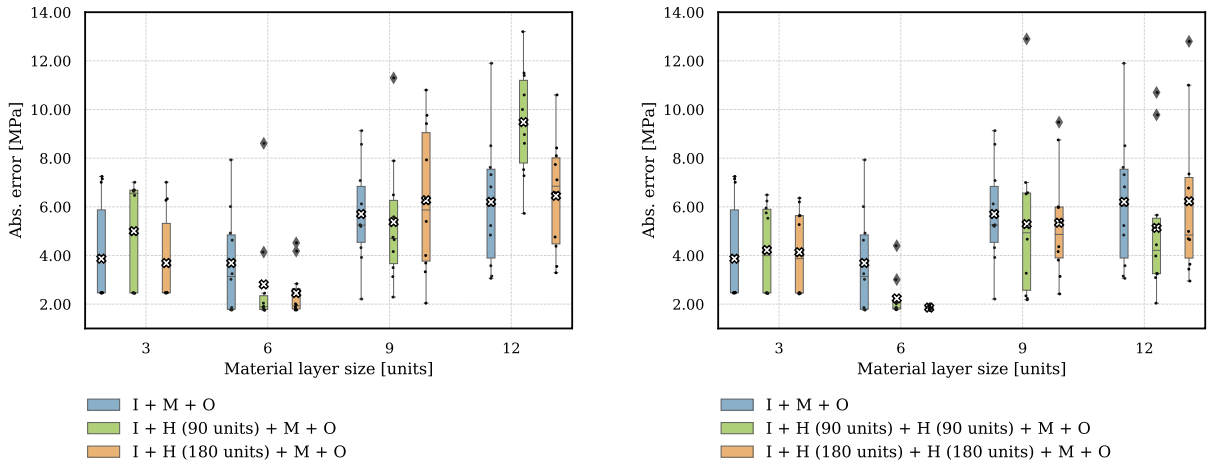
In the following section, a grid-search strategy is employed to choose the best architecture for the networks. The goal is to find the optimum architecture before heading to the testing sections. For that, different architectures and weight initializations are considered to mitigate the effect of randomness.

6.1. Model selection

For the PRNN, three types of architectures are considered. First, networks with input, material and output layers are considered. Then, models with a hidden layer activated with the tanh function before the material layer are investigated. Finally, networks with two hidden layers in the encoder are considered. Note that this is not an exhaustive search as many other architectures are possible and suitable for both the encoder and decoder. In this study, the size of the hidden layers in the encoder are pre-defined (either 90 or 180 units) and the size of the material layer is variable.

Another important remark is that, in this case, only the elastoplastic model is used in the network so that the size of the material layer is the only variable in the model selection. Recall that the network still can make a subgroup to behave elastically by passing small strains to the material model. The training and the validation sets, $\mathfrak{D}_{\text{PRNN}}$ and $\mathcal{V}_{\text{PRNN}}$, consist of 18 Type I curves and 54 Type II curves, respectively. Fig. 9 shows the boxplots with the average validation error of each run alongside the mean error value over different architectures.

In all cases, the networks with 6 units (*i.e.* two fictitious material points) performed best. Furthermore, despite the larger variance in models without hidden layers other than the material layer itself, the best networks of this type are as accurate as the ones with one or two hidden layers with significantly fewer parameters (see lower bounds in the boxplots). Another concern in using overly complex models in this particular case is the difficulty to assess the accuracy of the tangent stiffness matrix for use in FE^2 applications without probing the model in numerical applications or explicitly including it in the formulation (*e.g.* through the loss function). In that light, we opt for the most parsimonious architecture – the one with a single material layer between the input and output layers – for



(a) Effect of including one hidden layer in the encoder and material layer of variable size

(b) Effect of including two hidden layers in the encoder and material layer of variable size

Fig. 9. Absolute error of PRNNs trained on $\mathfrak{D}_{\text{PRNN}} = \{18 \text{ Type I curves}\}$ over validation set $\mathcal{V}_{\text{PRNN}} = \{54 \text{ Type II curves}\}$. Letters I, H, M and O refer to the input, hidden, material and output layers, respectively.

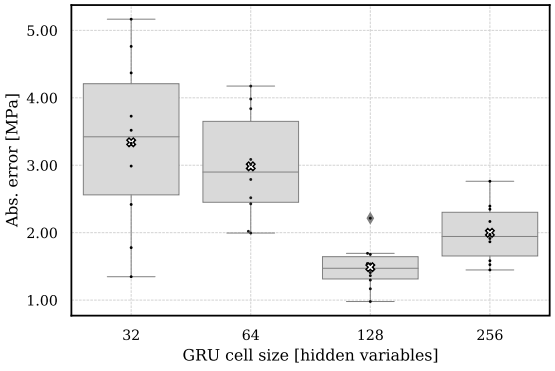


Fig. 10. Training error for RNNs trained on $\mathfrak{D}_{\text{RNN}} = \{18 \text{ Type I curves, } 90 \text{ Type II and } 90 \text{ Type III curves}\}$.

the rest of the paper as it led to more robust stresses and tangent stiffness matrix predictions in both single-scale tests and multiscale applications.

For simplicity, the architecture of the Bayesian RNN is composed of an input layer, a single GRU cell and an output (dense) layer. Again, weights and biases are randomized in each initialization, the training set ($\mathfrak{D}_{\text{RNN}}$) consists of the fixed set of 18 Type I curves, 90 Type II curves randomly chosen from a pool of 1800 curves, and 90 Type III curves also randomly chosen from a pool of 1800 curves, amounting to 198 loading paths. That way, all types of curves used for training in the following sections are covered. Fig. 10 shows the boxplot with the average training error of each run alongside the mean error value of all runs represented by the x marker. In this study, it is found that the GRU with 128 hidden variables performs best. In this case, no validation set is needed to determine the best dropout rate as the type of RNN used in this investigation infers it from the training data by default (see Section 3).

6.2. Predicting unloading/reloading from monotonic data

In this section, the training process of the RNNs on Type II curves (*i.e.* without unloading) is reported. The first test set consists of 100 Type II curves. Fig. 11 shows the average error of the RNNs over the test set compared to the best (blue triangle), worst (upside-down blue triangle), and average error (blue circle) found by the PRNNs.

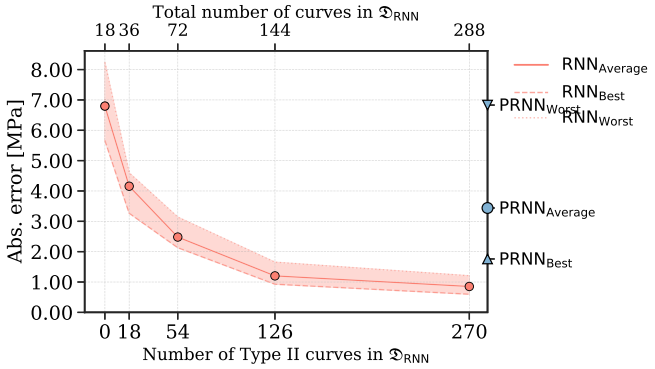
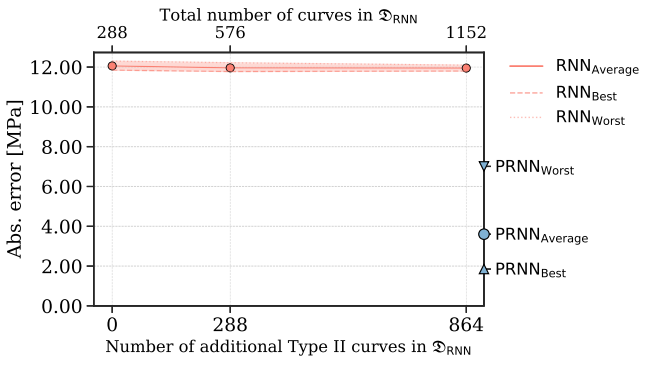
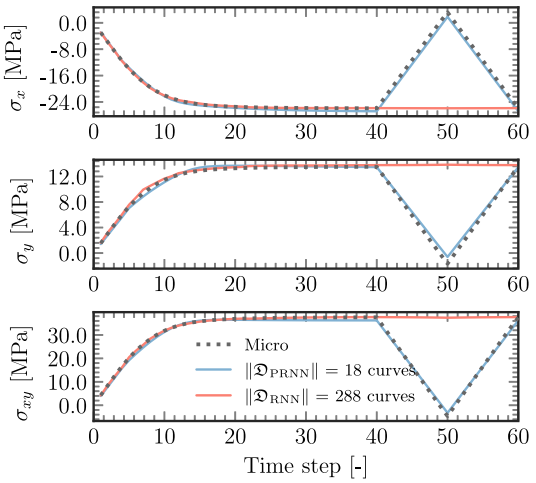


Fig. 11. Absolute error over monotonic test set $\mathcal{T}_{\text{II}} = \{100 \text{ Type II curves}\}$ for RNNs trained on Types I and II and PRNNs on 18 Type I curves.



(a) Absolute error over test set $\mathcal{T}_{\text{III}} = \{100 \text{ Type III curves}\}$ for RNNs trained on Types I and II and PRNNs on Type I only



(b) Stress-time view of representative case from test set $\mathcal{T}_{\text{III}}$ using the best RNN and PRNN

Fig. 12. Absolute error over non-monotonic test set $\mathcal{T}_{\text{III}}$ for RNNs and PRNNs trained only on 18 Type I curves and representative case.

Note that the secondary axis starts with 18 curves, this is because the known directions used for training the PRNNs are also a fixed set in the training of the RNNs. The training of the RNNs is stopped with 288 curves. At that stage, a similar level of accuracy between the PRNNs and the RNNs is obtained (although with a training set 16 times larger) and the addition of new curves only yields a marginal gain in accuracy.

Next, a new test set with 100 Type III curves is evaluated by the same networks trained on the 288 monotonic curves. This time, the RNN fails to capture unloading and the addition of more monotonic data is ineffective, as shown in Fig. 12(a). This outcome is not new to the literature and it is not surprising that RNNs need to see unloading behavior during training in order to be able to describe it. However, in contrast to the RNNs, the PRNNs provide the same level of accuracy for the test sets with and without unloading, even when not exposed to unloading data during training. In Fig. 12(b) a single representative case from test set $\mathcal{T}_{\text{III}}$ is plotted using the best RNN and PRNN. Both networks show good agreement with the reference solution until unloading starts (a feature not covered during training), but only the PRNN is capable of capturing the elastic unloading/reloading.

6.3. Predicting unloading/reloading behavior from non-monotonic data

Following the conclusions of Section 6.2, the training set of the RNN is expanded to include curves with the same unloading behavior as the one observed in the test set $\mathcal{T}_{\text{III}}$. The 288 monotonic curves of Types I and II from

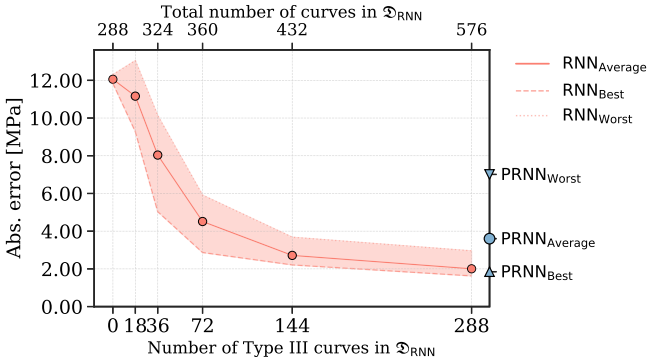


Fig. 13. Absolute error over non-monotonic test set $\mathcal{T}_{\text{III}} = \{100 \text{ Type III curves}\}$ for RNNs trained on Types I, II and II and PRNNs on 18 Type I curves only.

the previous section are combined with an increasing number of non-monotonic curves of Type III. This time, with the right features included in the training set, Fig. 13 shows a monotonic decrease of the average error for the RNN on $\mathcal{T}_{\text{III}}$ curves. However, the performance of the RNN only meets the one obtained by the PRNN with around 32 times more data.

6.4. Predicting unseen patterns from non-monotonic data

In this section, three additional test sets are considered for the RNN trained on Types I, II and III and the PRNN trained on Type I only. The goal is to test the ability of the networks to predict the macroscopic stress with patterns different from those seen during training.

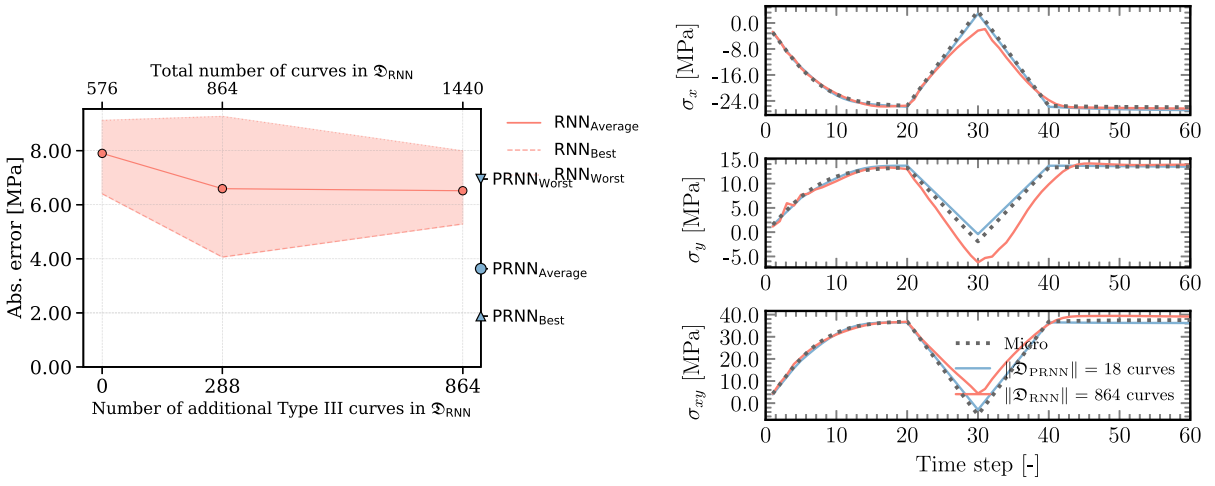
First, we consider a test set with 100 unseen curves of Type IVa, which consists of proportional curves in random directions with a different predefined unloading/reloading behavior than that of Type III. The average error for that set is shown in Fig. 14(a). Here, the 576 curves from the previous section are no longer enough to provide good accuracy when predicting a different unloading/reloading. By adding more Type III curves to the training set of the RNN, the average error decreases from 6.4 MPa to around 4.0 MPa but no significant gain in the accuracy is observed when the total number of curves is larger than 864 curves. Based on that, a representative case from test set $\mathcal{T}_{\text{IVa}}$ is shown in Fig. 14(b). Despite the relative low error from both networks, note that the RNN looses performance once unloading starts while the PRNN continues to show good agreement throughout the entire loading path.

For the next scenario, a test set with 100 Type IVb curves are considered. These curves have the same unloading/reloading behavior as Type III, but with a 10× smaller time step. Fig. 15(a) illustrates the average error of 10 networks over that test set and again, it is clear that the addition of new curves with patterns different from the exact one being tested is not beneficial to the RNN. Again, the PRNN provides good accuracy. Essentially, the PRNN is only as sensitive to step size as the material models embedded in it. Fig. 15(b) illustrates the networks' predictions for a curve in the test set $\mathcal{T}_{\text{IVb}}$.

As a final test, a set of 100 Type V curves, which corresponds to non-proportional and non-monotonic paths, is considered. This type of curve combines the two previous features: different step sizes and different unloading/reloading locations. Fig. 16(a) shows the average error for additional non-monotonic curves in the training of the RNNs. It is clear that the RNN completely fails to capture non-proportional paths (lowest error around 32 MPa) and that the addition of more data with features different than those being tested is a waste of resources. Although in different levels, a similar trend of loss of accuracy is also observed in the PRNNs, where the best, average and worse performances result in errors around 8.9 MPa, 17.0 MPa and 11.2 MPa respectively. Fig. 16(b) illustrates the different order of error between the PRNN and the RNN on a representative case from test set $\mathcal{T}_{\text{V}}$.

6.5. Training both networks on non-monotonic and non-proportional loading

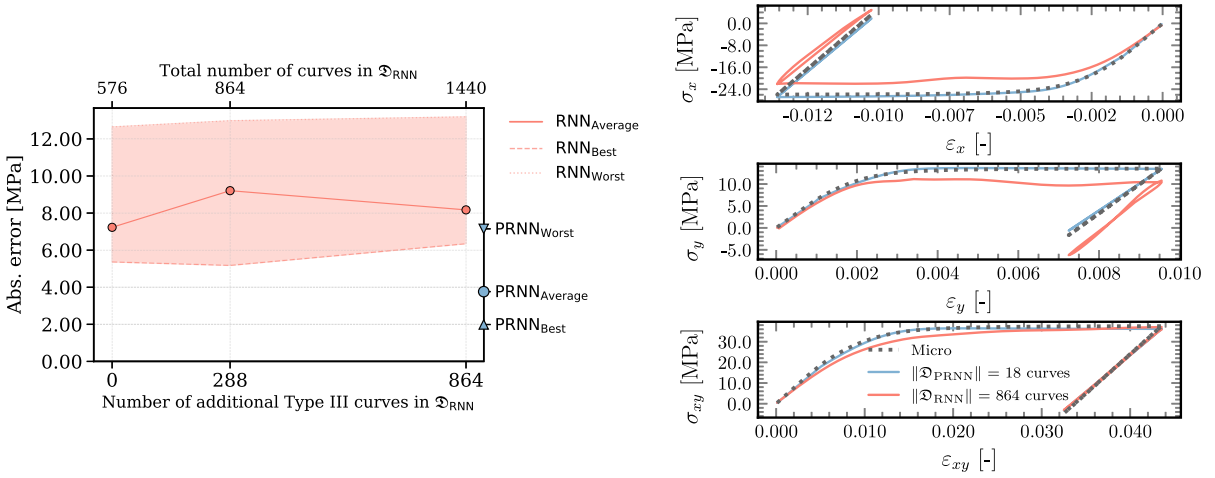
In this section, both networks are trained on the most generic set of curves, *i.e.* random non-monotonic and non-proportional curves of Type V. In addition to that, we trained the PRNNs on the known and proportional loading



(a) Absolute error over test set $\mathcal{T}_{\text{IVa}}$ = {100 Type IVa curves} for RNNs trained on Types I, II and III and PRNN on Type I only

(b) Stress-time view of representative case from test set $\mathcal{T}_{\text{IVa}}$ using the best RNN and PRNN

Fig. 14. Absolute error over non-monotonic test set $\mathcal{T}_{\text{IVa}} = \{100$ Type IVa curves$\}$ for RNNs trained on Types I, II and II and PRNNs on 18 Type I curves and representative case.



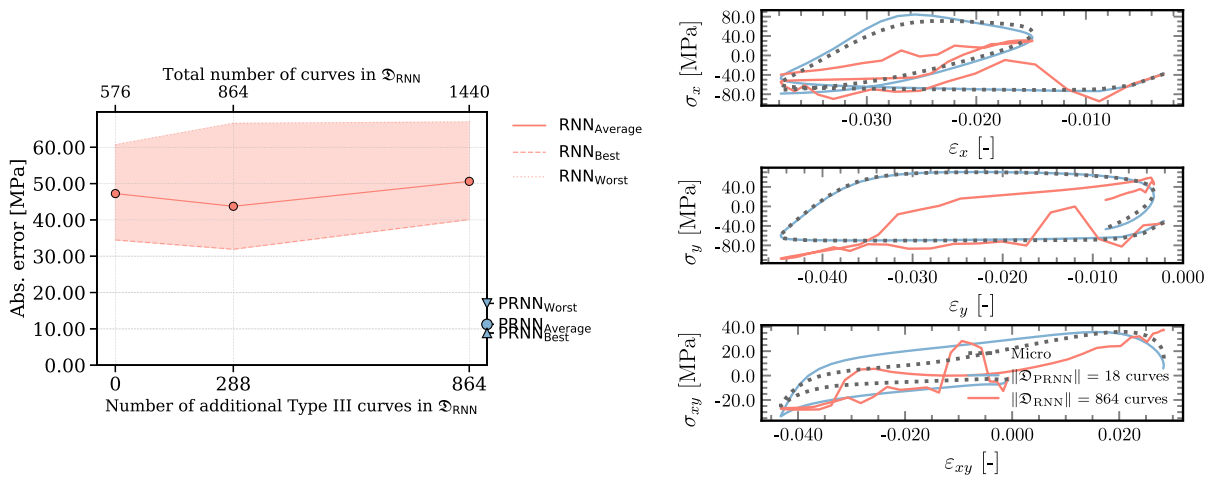
(a) Absolute error over test set $\mathcal{T}_{\text{IVb}}$ = {100 Type IVb curves} for RNNs trained on Types I, II and III and PRNNs on Type I only

(b) Representative $\boldsymbol{\varepsilon}$-$\boldsymbol{\sigma}$ curve from test set $\mathcal{T}_{\text{IVb}}$ using the best RNN and PRNN

Fig. 15. Absolute error over different step size test set $\mathcal{T}_{\text{IVb}} = \{100$ Type IVb curves$\}$ for RNNs trained on Types I, II and III and PRNNs on 18 Type I curves and representative case.

cases for comparison purposes. In that case, the size of the material layer is kept at 6 units and three training dataset sizes are considered. First, only the pure uniaxial cases are included, which yields 6 loading cases. Then, the 4 biaxial cases are added to the previous training dataset, resulting in 10 loading cases. And finally, we add the 8 cases with biaxial and shear loading, which amounts to the 18 fundamental paths shown in Fig. 6(b).

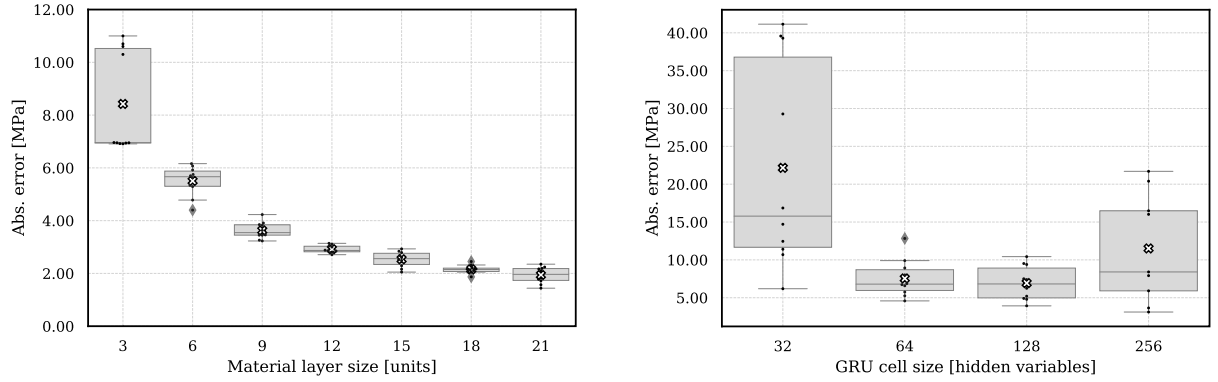
When training both networks on non-proportional paths a new model selection procedure was carried out to determine the optimum size of the material layer and the GRU cell, respectively. In this preliminary study, 10 different weights initialization are considered again. For training the RNNs and the PRNNs, 2304 and 198 Type V curves are used, respectively. Figs. 17(a) and 17(b) show the boxplot with the average error of each run alongside the mean error value. In this case, the networks with 18 units (which corresponds to six fictitious material



(a) Absolute error over test set $\mathcal{T}_{\text{V}}$ = {100 Type V curves} for RNN trained on Types I, II and III and PRNN on Type I only

(b) Representative $\boldsymbol{\varepsilon}$-$\boldsymbol{\sigma}$ curve from test set $\mathcal{T}_{\text{V}}$ using the best RNN and PRNN

Fig. 16. Absolute error over non-proportional and non-monotonic test set $\mathcal{T}_{\text{V}} = \{100 \text{ Type V curves}\}$ for RNNs trained on Types I, II and III and PRNNs on 18 Type I curves and representative case.



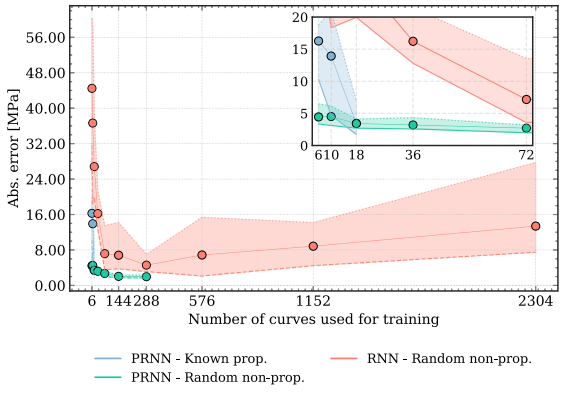
(a) Validation error for PRNNs trained on $\mathfrak{D}_{\text{PRNN}}$ = {198 Type V curves} and $\mathcal{V}_{\text{PRNN}}$ = {54 Type V curves}

(b) Training error for RNNs trained on $\mathfrak{D}_{\text{RNN}}$ = {2304 Type V curves}

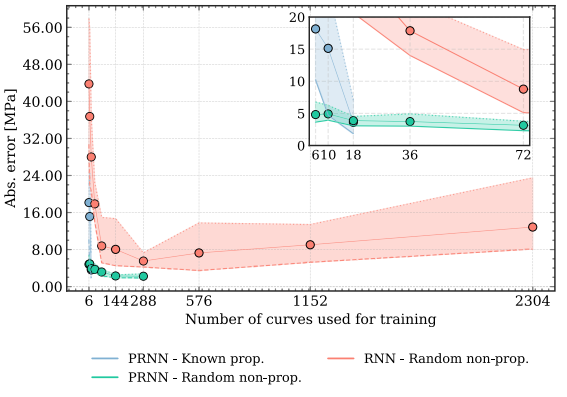
Fig. 17. Model selection of PRNN and RNN for non-monotonic and non-proportional loading.

points) performed better. For the Bayesian RNN, the GRUs with 128 hidden variables continue to provide the best performance. Therefore, this architecture is the one used in the comparison presented below.

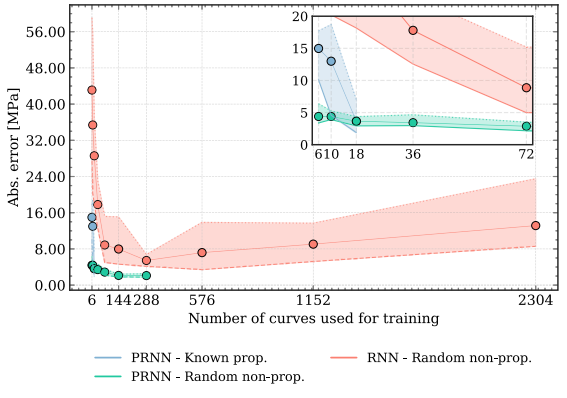
This time, all test sets discussed in Section 6.2, 6.3 and 6.4 are used again to assess the accuracy of the networks with the new sampling strategy. Figs. 18(a)–18(e) show the best, worst and average error for the cases studied so far in order. Based on this study, a few important insights are worth mentioning: after a certain point (around 576 curves), the RNNs reach an optimum level of accuracy and the addition of new curves no longer boosts predictions for proportional loading cases ($\mathcal{T}_{\text{II}}$, $\mathcal{T}_{\text{III}}$, $\mathcal{T}_{\text{IVa}}$ and $\mathcal{T}_{\text{IVb}}$). This is in line with the behavior observed in the previous sections, in which the RNNs would only perform well when trained with the same features as in the test set. And more importantly, changing the sampling strategy also showed to have limited effect on improving their performance. Granted, increasing even further the number of curves used for training as well as the complexity of the RNN might



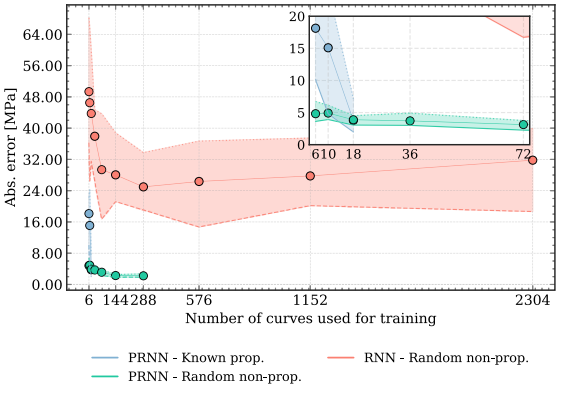
(a) Test set $\mathcal{T}_{\text{II}}$ = {100 Type II curves}



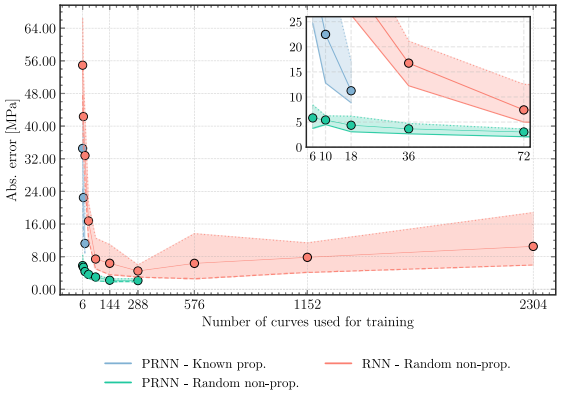
(b) Test set $\mathcal{T}_{\text{III}}$ = {100 Type III curves}



(c) Test set $\mathcal{T}_{\text{IV}_\text{a}}$ = {100 Type IVa curves}



(d) Test set $\mathcal{T}_{\text{IV}_\text{b}}$ = {100 Type IVb curves}

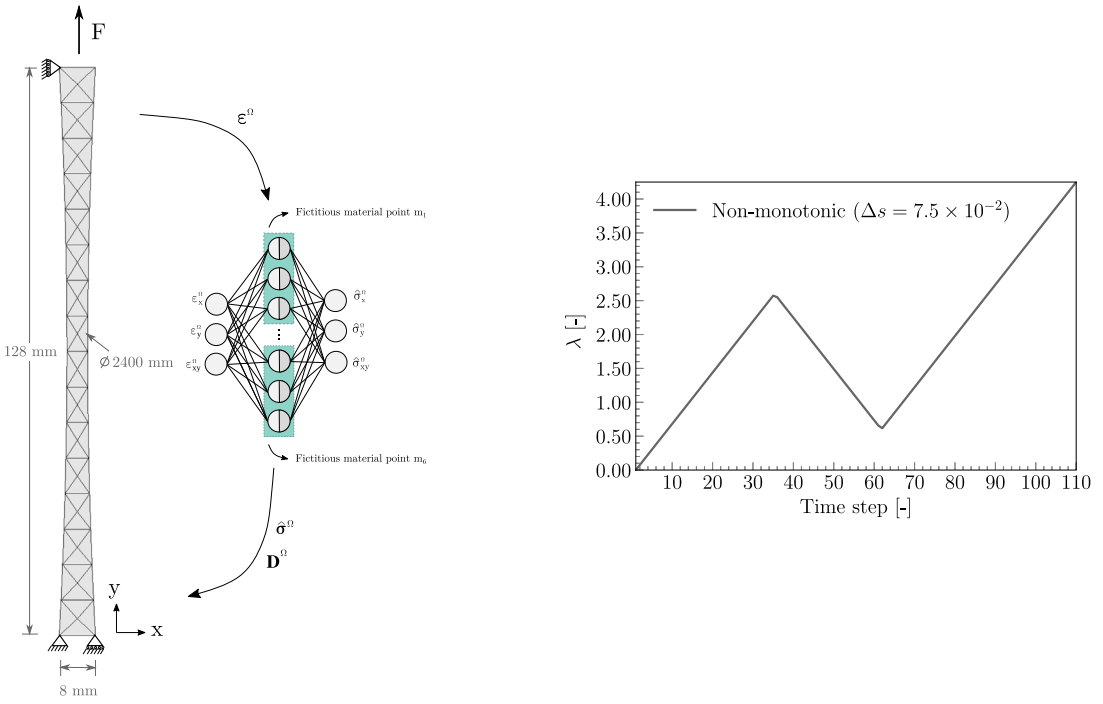


(e) Test set $\mathcal{T}_{\text{V}}$ = {100 Type V curves}

Fig. 18. Absolute error of networks trained on different sampling strategies and different test sets.

help in that task. However, the point stands that with the PRNN, this is not necessary. Note that for the same training set sizes, the PRNN with either the known or the random curves performs better than using the RNNs.

Finally, when choosing between known and random curves for training the PRNN, the latter shows comparable errors with the first when predicting proportional loading, but is significantly more accurate (see detail in Fig. 18(e)) for non-proportional loading. For that reason, the PRNN trained on Type V is chosen to illustrate the network's



(a) Geometry and boundary conditions (b) Loading function

Fig. 19. Tapered bar FE^2 example with (a) geometry and boundary conditions and (b) loading function.

capacity in the following FE^2 examples. On average, the accuracy of the PRNN reaches a plateau around 36 curves. From that point on, the benefit of adding new data is limited.

7. FE^2 applications

In this section, the PRNN trained with 36 non-proportional and non-monotonic Type V curves and lowest test error for test set $\mathcal{T}_V$ in Section 6.5 is employed as the constitutive model in two numerical examples. Results obtained with the PRNN as constitutive model are compared against results obtained with full FE^2 with the same micromodel that the network was trained to be a surrogate for. Both types of analysis are performed with an in-house Finite Element code using the open-source Jem/Jive C++ numerical analysis library [37].

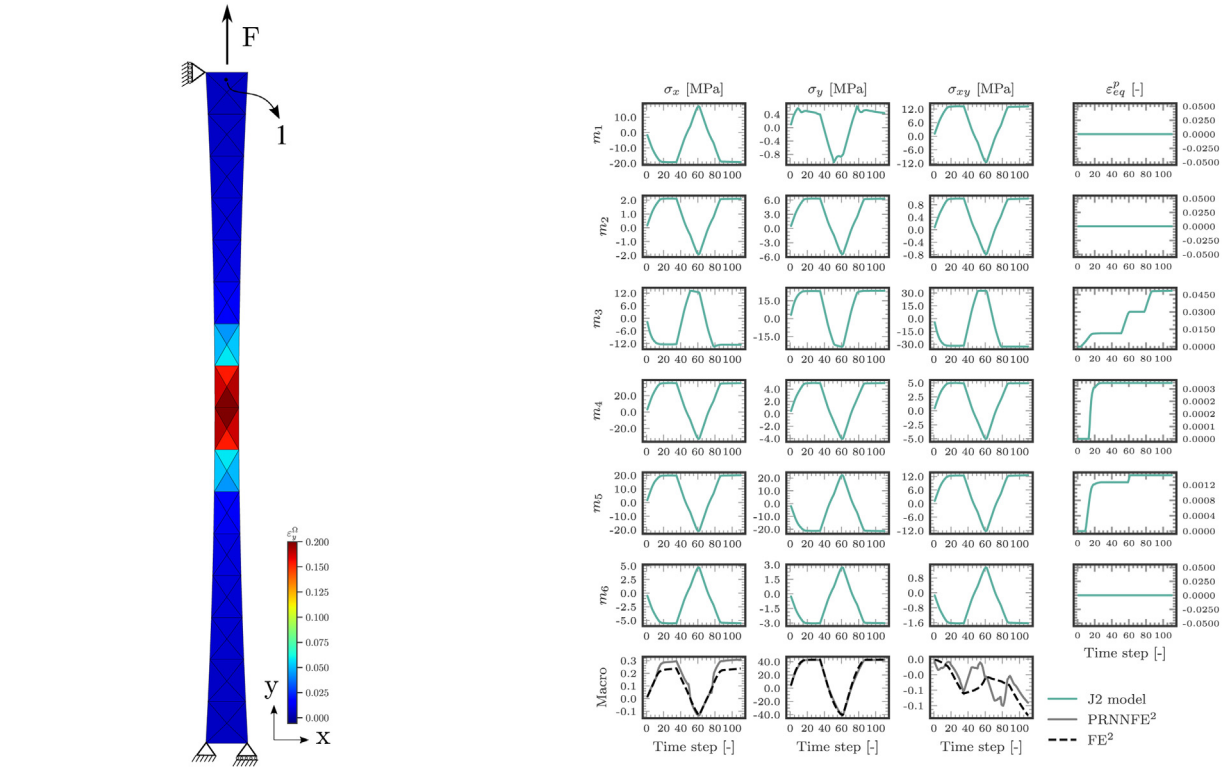
At the macroscale, an arc-length method with an adaptive-stepping scheme [38] is adopted to tackle potential convergence issues. This way, if a loading step does not converge with the given (full) step size, a reduction factor is applied to it until the loading step converges or until a maximum number of reductions in the initial step has been reached, terminating the analysis. Upon convergence, in the following step, the analysis is resumed with the full step size. In this work, each load step can be reduced by a factor of 0.4 for a maximum number of 5 times.

All simulations, including the PRNN training, were executed on a single core of a Xeon E5-2630V4 processor on a cluster node with 128 GB RAM running CentOS 7.

7.1. Tapered bar

The first example consists in a tapered composite specimen with length of 128 mm and height of 8 mm loaded in transverse tension. In this setting, the 36-fiber RVE model used to train the networks in the previous sections is embedded at each integration point of the macroscale. The geometry and the boundary conditions are shown in Fig. 19(a). The FE^2 problem is solved for 110 load steps with unloading according to the function shown in Fig. 19(b). At this point, the macroscopic response is already in the plastic regime.

The strain field at the end of the analysis is shown in Fig. 20(a) along with the location of one macroscopic integration point. This point is used to illustrate the state of the PRNN throughout the time steps. Recall that the



(a) Strain field at the end of the analysis using the PRNN

(b) Detailed view of PRNN for integration point 1

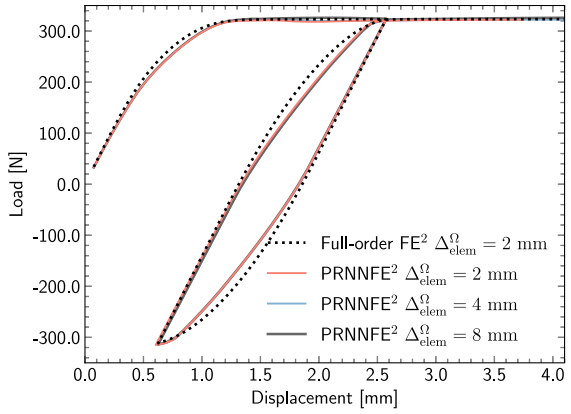
Fig. 20. Strain field using the PRNN on the left and detailed view of PRNN for a single macroscopic integration point on the right. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

network used in this section consists of 18 units (*i.e.* 6 fictitious material points). Thus, each row in Fig. 20(b) corresponds to a fictitious material point, each with its own stress path and internal variables (even though only one of them is plotted).

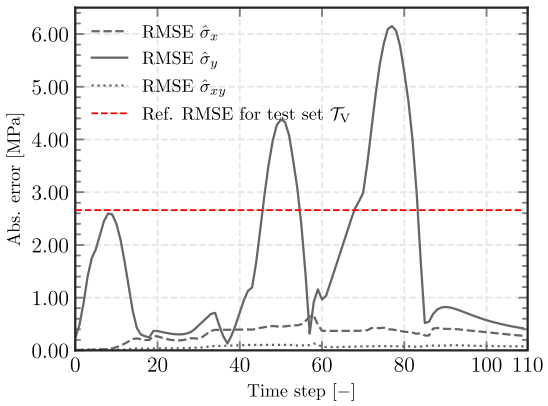
In this case, each component of the macroscopic response (*i.e.* homogenized stresses) is simply the linear combination of the local stresses of the 6 material models. It can be observed that the two macroscopic responses are in excellent agreement, with minor deviations in stress components with low magnitude. This is only visualized for a single point, but it is emphasized that in this multiscale problem, agreement in the evolution of the stresses in a single integration point indicates that the whole problem is solved accurately. Moreover, the equivalent plastic deformation of each material point is plotted in the last column. Note that despite the plastic response of the RVE after time step 20, three of the fictitious material points of the network (m_1, m_2 and m_6) remain in the elastic regime.

The accuracy of the PRNN is further assessed by inspecting the load–displacement curve at the top edge of the bar. Fig. 21(a) shows the load–displacement curve using the full-order solution and the network's response for different macroscopic mesh discretizations. For the refinement studied so far ($\Delta^{\Omega}_{\text{elem}} = 8$ mm), it is clear that the proposed network can capture accurately the entire nonlinear response, albeit with minor deviations when the tapered bar changes from tension to compression and then back again to tension.

Next, the global accuracy of the method is verified. Since the surrogate model is not as accurate as the full-order model, a different equilibrium solution at a certain time step affects the equilibrium in the following loading steps, leading to accumulated error and diverging $\boldsymbol{\varepsilon}$-$\boldsymbol{\sigma}$ paths. In this case, since no reduction in the step size was observed at any moment, the simple average error between the PRNN prediction and the full-order solution is calculated for each time step averaged over all integrations points at the macroscale, as illustrated in Fig. 21(b). For most of the simulation, the average absolute error in the predictions remains below 1 MPa with two peaks around 4 MPa and 6 MPa when the loading is reversed, following the trends observed in Fig. 21(a). For reference, the lowest error of the network for test set $\mathcal{T}_{\text{V}}$ is plotted.



(a) Load-displacement curve obtained with the full-order FE^2 approach and with PRNN for different macroscopic mesh discretizations



(b) Average RMSE at each time step of analysis with $\Delta^{\Omega}_{\text{elem}} = 8$ mm

Fig. 21. Tapered bar FE^2 example with (a) load–displacement curve with the full-order solution and best PRNN and (b) average error of PRNN's predictions at each time step of the analysis with $\Delta^{\Omega}_{\text{elem}} = 8$ mm.

Table 1
Computational cost for different mesh discretizations and efficiency of network in FE^{2}approach.

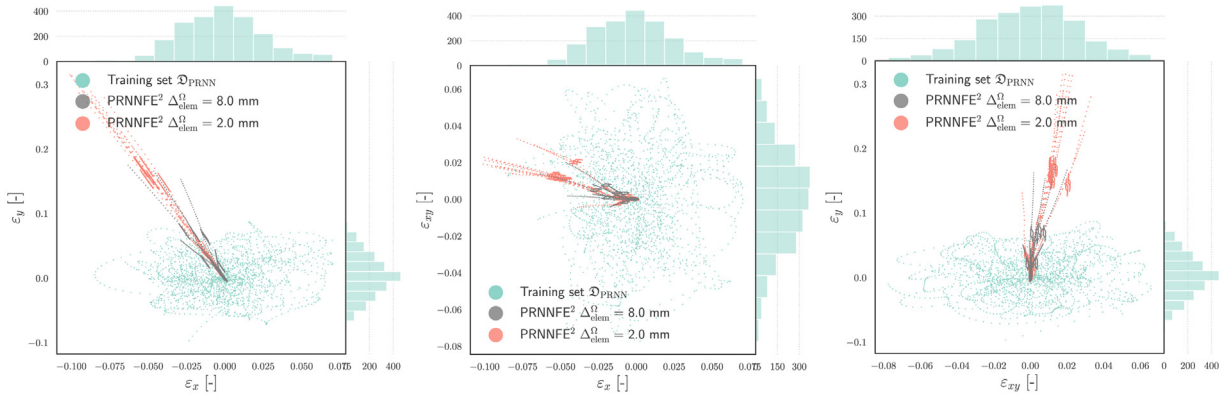
	Macroscale element size ($\Delta^{\Omega}_{\text{elem}}$) [mm]	8	4	2
	Number of elements at the macroscale	64	134	454
Online	FE^{2}wall-clock time [s]	21574	47644	178800
	PRNNFE^{2}wall-clock time [s]	0.81	1.55	8.41
	Speed-up[a] [-]	26560	30746	21526
Offline	Av. wall-clock time per curve (dataset gen.) [s]	265[b]	N/A	N/A
	Av. training time (excl. dataset gen.) [s]	38045[b]	N/A	N/A

[a]Evaluated as FE^{2}wall-clock time/PRNNFE^{2}wall-clock time and averaged over 5 runs.
[b]One-off cost regardless of macroscopic mesh discretization.

For the purpose of assessing the efficiency of the network in accelerating the FE^2 simulations, three different levels of mesh refinement of the tapered bar are taken into account, being the coarses the one shown in Fig. 19. Table 1 summarizes the wall-clock time spent in the analysis of the different discretizations, as well as the speed-up in comparison to the full-order solution. For the mesh used to illustrate this section ($\Delta^{\Omega}_{\text{elem}} = 8$ mm), replacing the solution of the BVP of the micromodel with the network led to a speed-up over 26 000 with the accuracy reported in Fig. 21(b). Considering the offline costs, the training time is still lower than that of using the full-order solution with 134 macroscopic elements, which is a very modest number of elements for a multiscale problem.

Since the network is trained to replace the solution of the microscopic model, no additional training is required for the analysis of more complex cases where the macroscale problems require more elements and time steps. Hence, in general, higher speed-ups should be achieved with denser meshes. However, in this particular problem, this is not always the case. An increase from the coarsest to the intermediary mesh is observed, but no gain is achieved when refining even further. In that case, the reduction in performance due to the higher number of iterations caused by the necessity of adaptively reducing the step size in order to ensure convergence. In contrast, the full-order solution was more numerically stable for this mesh density and the adaptive-stepping scheme was not triggered.

As the mesh is refined and strain localization takes place (see the red region in Fig. 20(a)), even higher strain levels are achieved, pushing the network to make predictions in unexplored regions during training, as illustrated in Fig. 22. Note that the network is already making far-reaching predictions in the coarsest discretization, although in a less extensive way. In the mesh with $\Delta^{\Omega}_{\text{elem}} = 8$ mm, the maximum strain does not exceed 0.2, while $\Delta^{\Omega}_{\text{elem}} = 2$ mm leads to strains higher than 0.3. In spite of these complicating aspects, it is worth mentioning this is still a significant speed-up. Moreover, a far less severe effect on the global accuracy is observed, as illustrated by



(a) Macroscopic strains in x and y (b) Macroscopic strains in x and xy (c) Macroscopic strains in xy and y

Fig. 22. Joint distribution of strains from the training set of the best PRNN and strain distribution obtained by PRNNFE^2 for the tapered bar problem with different macroscopic mesh discretizations.

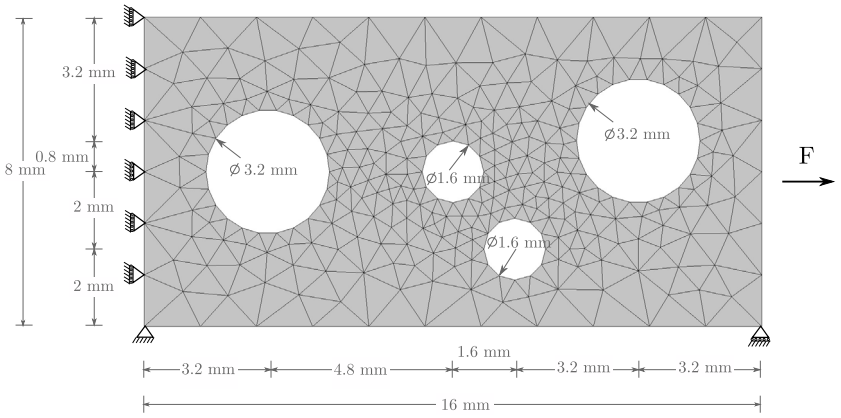


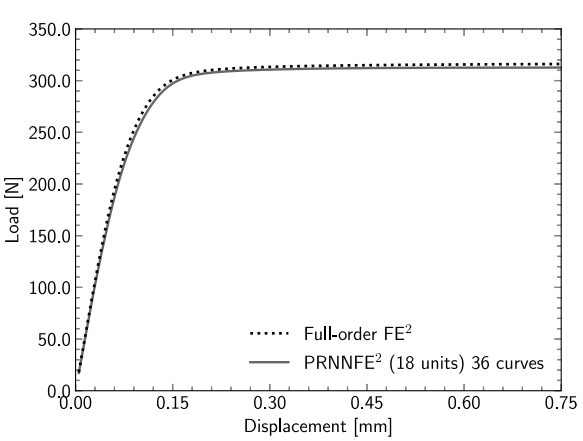
Fig. 23. Plate with cutouts: geometry and boundary and loading conditions.

the almost overlapping load–displacement curves in Fig. 21(a). In that sense, the adaptive-stepping scheme plays an important role to help overcome convergence issues.

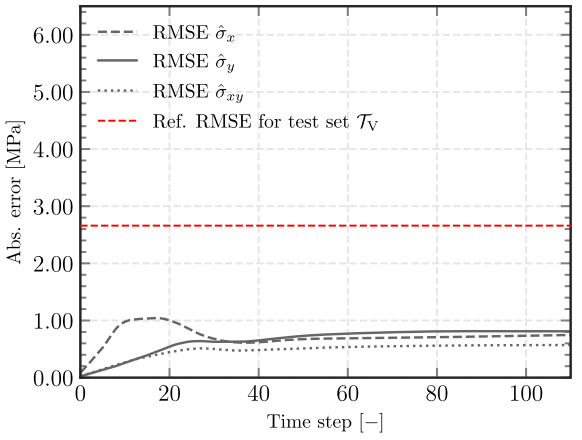
7.2. Plate with multiple holes

As a final example, a composite plate with multiple cutouts with geometry and boundary and loading conditions as illustrated in Fig. 23 is studied. Again, an FE^2 approach is employed to solve the problem for the same microscopic model with which all the networks in Section 6 were trained for. This time, no unloading is imposed and 150 load steps with $\Delta s = 5.0 \times 10^{-3}$ are considered. The load–displacement curve at the right edge of the plate is plotted in Fig. 24(a) using both the full-order solution and the network. Again, good agreement is observed between the macroscopic responses. The slight inaccuracy between those are quantified in Fig. 24(b), in which the average absolute error of the component with the highest magnitudes (σ_x) is around 1 MPa for almost the entire simulation.

The displacement field at the end of the analysis is shown in Fig. 25(a), where the location of five macroscopic integration points are marked for further inspection. The stress paths for each of these points are illustrated in Fig. 25(b), where the full-order solution and the network prediction are plotted in black and gray, respectively. Note how the stress paths are non-proportional even for the relatively simple loading condition observed in the macroscale. The integration point on the edge of one of the cutouts, namely point 4, is also the one with the highest

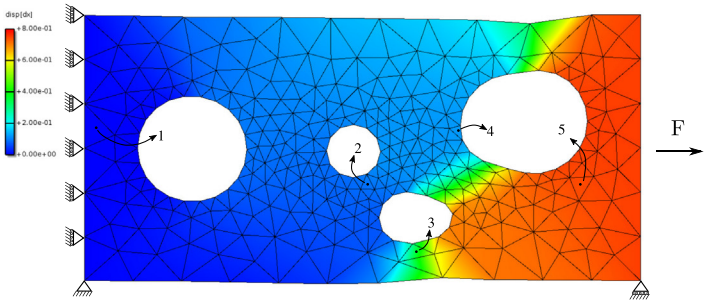


(a) Load-displacement curve

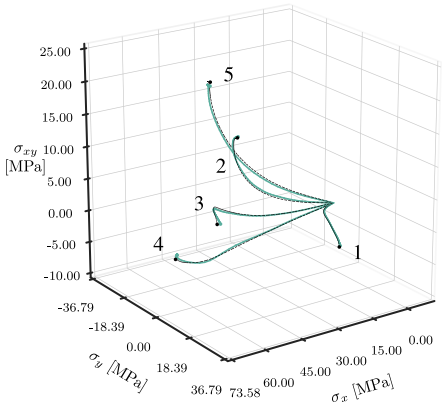


(b) Average RMSE

Fig. 24. Plate with cutouts: (a) load–displacement curve and (b) average error of PRNN's predictions at each time step of the analysis.



(a) Displacement field at the end of the analysis with PRNN and selection of integration points for inspection



(b) Stress-time view of macroscopic integration points. Full-order solution in black and network's prediction in green

Fig. 25. Plate with cutouts: (a) displacement field at the end of the analysis and (b) selected integration points shown in stress–time view. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

stress magnitude and the closest to a uniaxial state in the x direction while the other points experience multiaxial loading more strongly.

In terms of efficiency, the solution using the network is around 26 705 times faster than the full-order solution, which took approximately 258 780 s (around 72 h). The order of magnitude in the speed-up is similar to that obtained in the tapered bar problem. Although no additional offline costs are incurred because the network has been trained before for the same microscopic model and it does not depend on the macroscopic problem at hand, it is worth stressing that the runtime of the full-order solution exceeds the sum of the online and offline costs of the PRNN.

Finally, this example shows that the network can capture multiaxial stress states and non-proportional loading as obtained in FE^2 simulations accurately. No convergence issues were encountered in the PRNNFE^2 simulation which points to the smoothness of the predictions that is not always guaranteed with surrogate models (see *e.g.* RNN curves in Fig. 16(b)).

Table 2
Material properties of RVE modeled by two elastoplastic models.

	E [MPa]	ν [-]	σ_y [MPa]
Constitutive model of matrix $\mathcal{D}_1^\omega$	3130	0.37	$64.8 - 33.6\exp^{-\varepsilon_{eq}^p/0.0003407}$
Constitutive model of fibers $\mathcal{D}_2^\omega$	2130	0.25	$77.76 - 33.6\exp^{-\varepsilon_{eq}^p/0.0003407}$

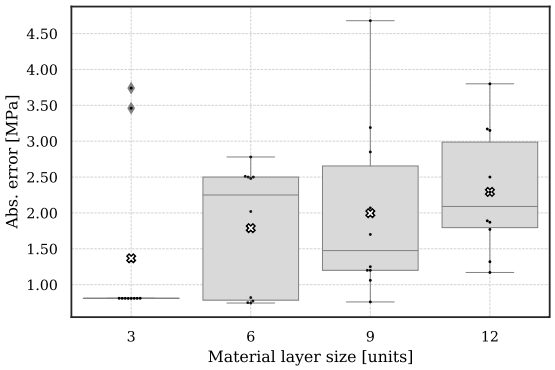


Fig. 26. Absolute error for PRNNs trained on $\mathfrak{D}_{\text{PRNN}} = \{18 \text{ Type I curves}\}$ over validation set $\mathcal{V}_{\text{PRNN}} = \{54 \text{ Type II curves}\}$ for RVE with two elastoplastic phases.

8. Extended experiments

In this section, two additional studies are carried out to demonstrate the flexibility of the proposed approach to handle various types of material models with different levels of complexity, as well as to help identify potential pitfalls when choosing the architecture and the design of experiments. In both scenarios, the same RVE geometry presented in Sections 6 and 7 is used.

8.1. Two elastoplastic phases with different material properties

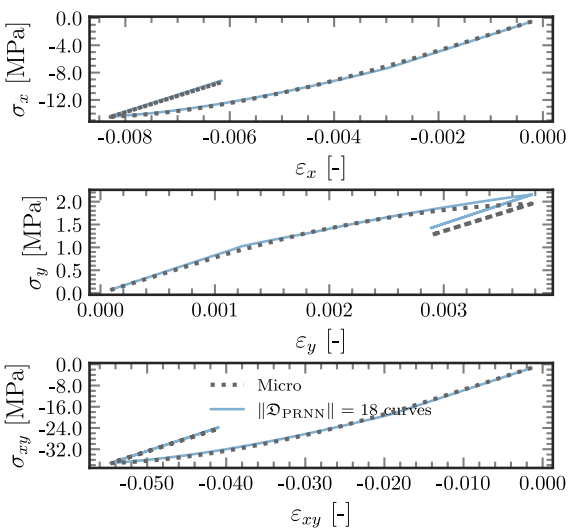
In this study, plane stress conditions are kept, but both matrix and fibers are now described by the elastoplastic model with the von Mises yield criterion and isotropic hardening with the equivalent plastic strain given by Eq. (34) and material properties as described in Table 2.

For training the networks, we follow the steps discussed in Section 6, in which the initial training set comprises 18 Type I curves and the validation set consists of 54 Type II curves. The architecture of the networks consists of an input layer, a material layer with all fictitious material points evaluated by the constitutive model $\mathcal{D}_1^\omega$ and an output layer. Again, 10 initializations and four different layer sizes are considered. Fig. 26 shows the boxplots with the average validation error of each run. We select the architecture with three units (or one fictitious material point) and assess its performance on test sets $\mathcal{T}_{\text{III}}$ and $\mathcal{T}_{\text{V}}$ with 100 curves of Types III and V each respectively. The lowest average error for both test sets are around 0.63 MPa and 3.13 MPa, respectively. Fig. 27 illustrates the accuracy of the network on a representative case from each of the test sets. Note that despite the increase in the average error over non-monotonic and non-proportional curves, the network is still capable of capturing relatively well its global trend.

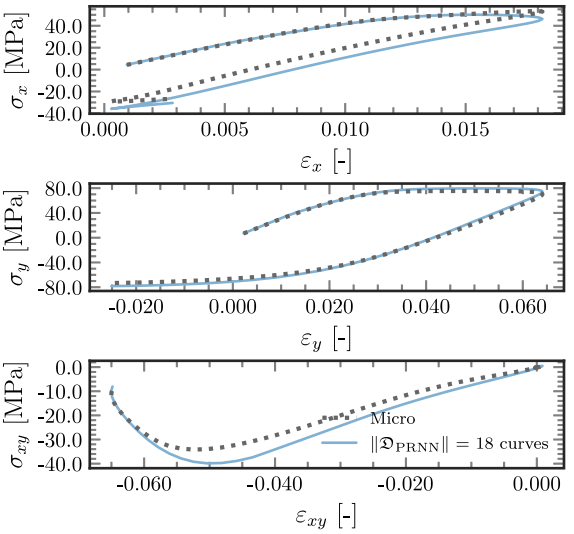
8.2. Elastoplastic and nonlinear elastic phases with the same material properties

In this study, a more complex elastoplastic model is considered: the material model proposed by Melro et al. [39]. For the sake of brevity, the details of the implementation are spared and the reader is referred to [39,40] for further clarification. This model uses a pressure-dependent yield criterion:

$$f(\boldsymbol{\sigma}, \sigma_c, \sigma_t) = 6J_2 + 2I_1(\sigma_c - \sigma_t) - 2\sigma_c\sigma_t \tag{33}$$



(a) Representative $\boldsymbol{\varepsilon}$-$\boldsymbol{\sigma}$ curve from non-monotonic test set $\mathcal{T}_{\text{III}}$



(b) Representative $\boldsymbol{\varepsilon}$-$\boldsymbol{\sigma}$ curve from non-proportional and non-monotonic test set $\mathcal{T}_{\text{V}}$

Fig. 27. Performance of the best PRNNs trained on 18 Type I curves over different test sets for RVE with two elastoplastic phases.

where I_1 and J_2 are stress invariants and σ_c and σ_t are compressive and tensile yield stress, both defined as general hardening functions of the equivalent plastic strain ε^p_{eq} with increment given by:

$$\Delta\varepsilon^p_{eq} = \sqrt{\frac{1}{1+2\nu_p^2}\,\Delta\boldsymbol{\varepsilon}^p : \Delta\boldsymbol{\varepsilon}^p} \tag{34}$$

where ν_p is the plastic Poisson's ratio, related to the non-associative flow rule. Plane strain conditions are assumed.

Based on this constitutive model, we concoct a modified version for which the updated internal variables calculated within the return mapping scheme are not stored at the end of every time step. This way, although dictated by a strictly identical hardening law and yielding criterion, no history-dependence is carried from one loading step to another, resulting in a material that behaves elastically in the sense that the loading and unloading follow the same path. This artificial material model allows us to illustrate unique scenarios that challenge the applicability of the proposed network.

For this study, the elastoplastic model by Melro et al. [39] $\mathcal{D}^{\omega}_1$ is used to describe the matrix and our modified nonlinear elastic version $\mathcal{D}^{\omega}_2$ to describe the fibers. The same material properties are adopted for both constitutive models, with E = 3130 MPa, ν = 0.37, ν_p = 0.32 and the two hardening laws in Eq. (33) given by:

$$\begin{aligned} \sigma_t &= 64.8 - 33.6\exp^{-\varepsilon^p_{eq}/0.0003407} \\ \sigma_c &= 1.2\cdot\left(64.8 - 33.6\exp^{-\varepsilon^p_{eq}/0.0003407}\right) \end{aligned} \tag{35}$$

Three different PRNN models are considered. First, the training and validation sets consist of 18 Type I curves and 54 Type II curves, respectively. The architecture consists of an input layer, a material layer containing two fictitious material points evaluated using the constitutive model $\mathcal{D}^{\omega}_1$ and an output layer. In the second model, the training set consists of 18 Type V curves and 54 Type V curves are used for validation. Recall that these curves are non-proportional and non-monotonic loading paths generated by GPs. The architecture in the first model is kept. Finally, in the third option, we train and validate with the same amount and type of data as in the second model, but the architecture is changed. This time, we evaluate one fictitious material point with $\mathcal{D}^{\omega}_1$ and the other with $\mathcal{D}^{\omega}_2$. Five different initializations are considered for each model. Fig. 28 shows the performance of the best PRNNs using the different strategies on a representative case from test set $\mathcal{T}_{\text{III}}$ which comprises 100 Type III curves.

The model with only $\mathcal{D}^{\omega}_1$ trained on Type I curves does very well in describing the monotonic behavior. Because both materials present in the micromodel have the same monotonic response, a single material model in the network

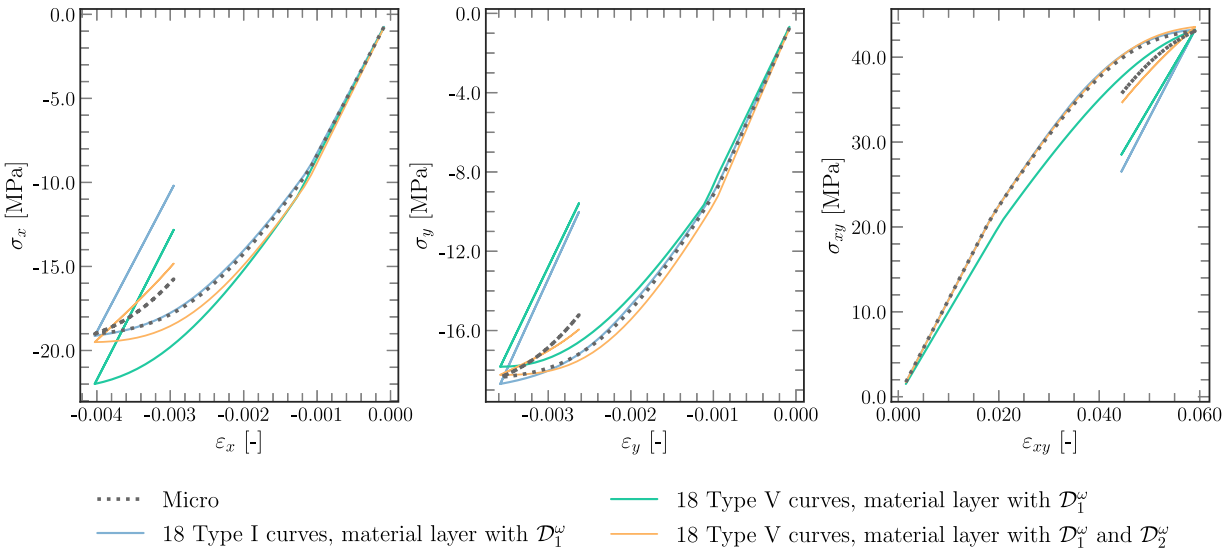


Fig. 28. Representative $\boldsymbol{\varepsilon}$-$\boldsymbol{\sigma}$ curves from test set $\mathcal{T}_{\text{III}}$ using the best PRNN trained on different training sets and constitutive models.

is sufficient for describing the monotonic behavior. However, upon unloading, the network is only able to reproduce the secant unloading behavior embedded in $\mathcal{D}_1^\omega$ and fails to capture the contribution from the nonlinear elastic unloading of $\mathcal{D}_2^\omega$. Furthermore, in this particular combination of constitutive models and under monotonic loading, the response of the micromodel is essentially the same as that of a homogeneous material. This emphasizes the point that the success we have had so far in capturing unloading accurately after training only on monotonic data came from the fact that the material points in the network included representative assumptions on the unloading behavior.

Training the PRNN with Type V curves does not improve the performance. On the contrary, the model loses accuracy even for the monotonic part. By minimizing the error of the entire loading path, which now includes multiple unloading cycles, the fitting ends up as a compromise between the loading and unloading behavior. Finally, it is observed that the network with both materials included in the material layer captures the mixed unloading behavior much better, although this does come at a cost concerning how well the monotonic part is described. Results can probably be improved by including more material points of one or both types and increasing the amount of training data.

9. Concluding remarks

In this paper, a novel network with embedded physics-based constitutive models is proposed as surrogate model for the behavior of path-dependent materials in FE^2 simulations. The central idea is to address common problems in modeling path-dependent materials using black-box models (*e.g.* unique mapping between input and output and limited extrapolation abilities) by taking a step back and reintroducing physics into the network in a way that requires very little extra coding effort with respect to existing FE^2 frameworks. This is done by employing the same material models used for the microscopic level as part of one of the layers of the network.

To accommodate this non-standard neural layer the following changes with respect to standard neural network architectures are proposed. First, neurons are assembled in groups of the size of the number of strain/stress components of the problem. These are referred to as fictitious material points. Secondly, to take advantage of all the information coming from the physics-based material model, we store the updated internal variables used to fully describe the state of the fictitious material point in an auxiliary vector. With that, when new strain values are fed, the material point will start from the last stored internal variables. As a consequence, each subgroup follows a unique path without the need to increase the feature space with extra history variables.

The properties and assumptions made by the physics-based constitutive model are inherited by the network and play a major role in reducing the amount of data required to mirror physical and complex behaviors such as elastic

unloading/reloading. Here, the decomposition of the strain in elastic and plastic parts is an assumption built in the material model used to describe the nonlinear microscopic material phase and is also observed in the network when the local stresses of the fictitious material points are evaluated. This simple but highly-flexible arrangement allows the network to capture arbitrary unloading behaviors with only monotonic data, a stark contrast with other popular models such as RNNs. The PRNN inherits from FE^2 the idea that complex behavior of heterogeneous materials can be accurately described by letting simpler constitutive models that represent the microscopic constituents interact. The difference is that the interaction between the constituents is not based on micromechanics directly but learned from data obtained from micromodel simulations.

Based on that, an extensive numerical comparison involving a state-of-the-art black-box model, namely a Bayesian Recurrent Neural Network (referred to in this work as RNN), was carried out in order to elucidate the abilities of the proposed network (referred to as PRNN). First, we trained both networks only on 18 monotonic curves with known directions and proportional loading in a similar fashion as done to calibrate classical mesomodels. Such strategy led to poor performance when trying to predict other random directions from the RNN, but good accuracy from our method (Fig. 11). Following that conclusion, the size of the RNN's training dataset was sequentially increased until the addition of more data did not result in significant gains in accuracy. At that stage, the PRNN performed with the same level of accuracy but with a factor of 16 times less data.

Next, both networks were used to predict non-monotonic loading. For that scenario, the PRNN showed the same level of accuracy as before with the same minimal training dataset (Fig. 12(a)). Such outstanding result is not observed in the RNNs, which again required a larger training set. This time, non-monotonic loading curves were added until the RNN's accuracy could no longer be significantly improved. As a result, a 32 times larger training dataset in comparison to the one used to train our network was necessary. Furthermore, while our network continues to perform well in all the scenarios tested so far, two other situations exposed the pitfalls of RNNs: (i) when trying to predict unloading in a different location than the one seen in training (Fig. 14(a)) and (ii) when the step size was modified (Fig. 15(a)). This is typically tackled by sampling different unloading behaviors with different step sizes, leading to the choice of arbitrarily long sequences. However, we showed that this is a trivial scenario for the PRNN. The network is only as sensitive to step size as the material models it includes.

In the last test, both networks were used to predict non-proportional and non-monotonic paths and neither succeeded, although they failed at very different levels. While the lowest error of the RNN was around 32 MPa, the best PRNN led to an error around 9 MPa error (Fig. 16(a)). Based on that, a second approach to generate the dataset was considered. Random strain paths were generated from Gaussian Processes priors, which produces non-proportional and non-monotonic loading as opposed to the proportional loading previously considered for training. This time, the size of the training dataset and the type of loading was also a variable for the PRNN. It was found that training the PRNN on random non-proportional and non-monotonic curves yields higher accuracy than training with known, proportional, and monotonic curves for all loading scenarios (Fig. 18). Although training with known directions is appealing, having a network that provides lower errors and consistent performance with random directions is also interesting. Ultimately, the PRNN consistently outperformed the RNN with 64 times less data.

After ensuring the PRNN capacity in several challenging scenarios for black-box models, one of the networks trained on non-proportional and non-monotonic curves was chosen to surrogate the microscopic model in a set of two FE^2 examples. The first example concerned a tapered bar in transverse tension and was used to illustrate how the different material models in the PRNN behave for a single macroscopic integration point (Fig. 19). For different discretizations, speed-ups between 21 000 and 31 000 were obtained for the online phase (Table 1). Such substantial efficiency gain is explained from the dramatically reduced number of material model calls and the bypassing of solving the nonlinear microscopic system of equations for macroscopic stress evaluation for a single load step of each macroscopic integration point.

In the last example, a similar order of magnitude of speed-up was observed and the accuracy of the PRNN was illustrated by comparing the $\boldsymbol{\varepsilon}$-$\boldsymbol{\sigma}$ paths of different macroscopic integration points of a plate with multiple cutouts subjected to tension (Fig. 25(b)). For the analyzed cases, the time needed for a single FE^2 analysis exceeded the total offline and online time for the PRNN analysis, even though the selected problems had a very modest number of macroscopic elements. Moreover, performing subsequent macroscale analysis with the same material would not require any additional offline training and would therefore leverage the complete speed-up of four orders of magnitude.

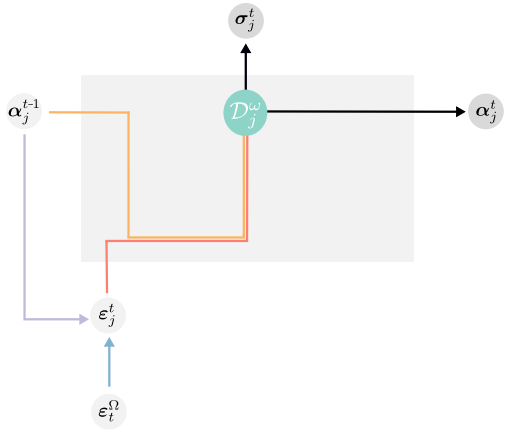


Fig. A.29. PRNN cell with local strains in the fictitious material point j now depend on the last internal variables state. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Finally, two additional studies were carried out to further demonstrate the flexibility of the proposed approach in handling various types of material models with different levels of complexity. In the first study, an elastoplastic model with different material properties was used to describe the two phases of a micromodel, while in the second a more complex elastoplastic model and a nonlinear elastic phase were considered to describe the constituents. For the first part, results followed the trend in which an accurate model can be obtained by training with monotonic data only and a single model in the network. In the second study, results illustrate potential pitfalls in not including both sources of nonlinearity and how to address the issue to obtain an accurate and robust surrogate model.

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

Data will be made available on request.

Acknowledgments

The authors acknowledge the TU Delft AI Initiative for their support through the SLIMM AI Lab. FM acknowledges financial support from the Dutch Research Council (NWO) under Vidi grant 16464.

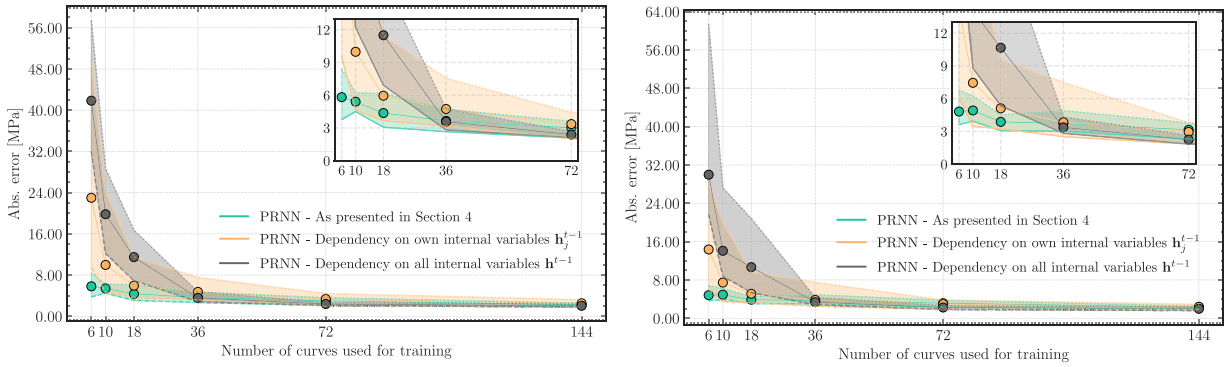
Appendix A. Encoder with explicit path-dependency

In this section, a different architecture for the encoder is presented. Here, the strains in each of the subgroups $\boldsymbol{\varepsilon}_j$ in the material layer are calculated based on the macroscopic strain $\boldsymbol{\varepsilon}^\Omega$ and on the internal variables stored in the previous time step $\boldsymbol{\alpha}_j^{t-1}$. For that purpose, a new set of weights is introduced in the network to learn how $\boldsymbol{\alpha}_j^{t-1}$ relates with the local strains $\boldsymbol{\varepsilon}_j$. This formulation is depicted in Fig. A.29, where an extended version of Fig. 5(b) is used to illustrate the new connections. The purple arrow represents the new set of weights, while the blue refers to the parameters that take into account the macroscopic strain (as originally proposed in Section 4). Yellow, red and black lines represent the flow of inputs and outputs of the constitutive model.

In the particular case where the architecture consists of input, material and output layers and no biases are considered, the current values used as strains input in the subgroup j can be defined as:

$$\boldsymbol{\varepsilon}_j = \mathbf{W}_1^j \, \boldsymbol{\varepsilon}_t^\Omega + \mathbf{H}^j \, \boldsymbol{\alpha}_j^{t-1} \tag{A.1}$$

where $\mathbf{W}_1 \in \mathbb{R}^{n_1 \times n_0}$ is the weight matrix connecting the material layer of size n_1 to the input layer of size n_0, $\mathbf{H} \in \mathbb{R}^{n_1 \times \ IntVar}$ is the additional weight matrix connecting the strains in the material layer to the internal variables



(a) Non-proportional and non-monotonic test set $\mathcal{T}_{\mathrm{V}}$ = {100 Type V curves}

(b) Non-monotonic test set $\mathcal{T}_{\mathrm{III}}$ = {100 Type III curves}

Fig. A.30. Absolute error over different test sets of the three different architectures trained on variable number of Type V curves and validated on 54 Type V curves. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

from the previous time step. In Eq. (A.1), j is used to refer to the part of the matrix (or vector) concerning the connections in the jth subgroup. A more general approach is to make the local strains dependent on the internal variables of all subgroups. In that case, the additional weight matrix $\mathbf{H}$ has size $\mathbb{R}^{n_1 \times\, m \cdot IntVar}$ and Eq. (A.1) simplifies to:

$$\boldsymbol{\varepsilon}_j = \mathbf{W}_1\ \boldsymbol{\varepsilon}_i^{\Omega} + \mathbf{H}\ \boldsymbol{\alpha}^{t-1} \tag{A.2}$$

with $\boldsymbol{\alpha}^{t-1}$ being the concatenation of all internal variables in the material layer. To compute the gradients of the weights $\mathbf{H}$, we follow the procedure and notation described in Section 4.4, but instead of multiplying the values $\bar{\mathbf{d}}_i$ by the activations of the previous layer (or next layer to be backpropagated), which correspond to the strains input, one must multiply these values by the history vector from the previous time step (which corresponds to $\boldsymbol{\alpha}^{t-1}$):

$$\frac{\partial L}{\partial \mathbf{H}} = \bar{\mathbf{d}}_i\ \mathbf{h}_{t-1}^{\mathrm{T}} \tag{A.3}$$

where $\bar{\mathbf{d}}_i$ is defined in Eq. (25). In addition to that, an extra term must be included in Eq. (26), $\mathbf{H}^{\mathrm{T}}\ \bar{\mathbf{d}}_i$, to account for the new connections illustrated by the purple arrow in Fig. A.29.

In both alternatives, the explicit introduction of the internal variables to the encoder makes the distribution of the macroscopic strain path-dependent. At this point, it is worth stressing that the proposal in Section 4 does now show this feature, but does take path-dependency into account in an implicit way by storing the internal variables updated by the material models. The contribution from the path-dependent internal variables also has a role in the tuning of the parameters in the encoder through the backpropagation process (see Eqs. (25) and (26)).

To assess the effect of introducing this feature in the network, we use the same RVE geometry, material models, material properties, and plane stress conditions as in Sections 6 and 7. Following the study in Section 6.4 where the network is trained with Type V curves and the architecture consists of an input layer, a single material layer with six fictitious material points and an output layer. Fig. A.30 shows the performance of the PRNNs on different test sets with non-monotonic loading paths. In both scenarios, the uncertainty bounds over the size of the training set are wider for the case in which the internal variables of all fictitious material points are taken into account to evaluate the strains in each of the subgroups. This is an expected behavior since more parameters need to be tuned by the network in comparison to the methodology proposed in Section 4. This difference is reduced if subgroup j takes into account only its own internal variables to evaluate the local strains (see orange curves).

However, as the size of the training set increases, the uncertainty bounds become narrower and the new architectures outperform the simpler case by a small margin before all three methods converge to a similar error level. On that note, it is still unclear whether this gain is worth the increased amount of data or how these networks perform in FE^2 problems. Both topics are open for discussion in future research work. For the present work, results indicate that the absence of an explicit path-dependent encoder is not impeditive to the performance of the network.

References

[1] O. Goury, D. Amsallem, S. Bordas, W. Liu, P. Kerfriden, Automatised selection of load paths to construct reduced-order models in computational damage micromechanics: from dissipation-driven random selection to Bayesian optimization, Comput. Mech. 58 (2016) 213–234, http://dx.doi.org/10.1007/s00466-016-1290-2.

[2] I.B.C.M. Rocha, P. Kerfriden, F.P. van der Meer, On-the-fly construction of surrogate constitutive models for concurrent multiscale mechanical analysis through probabilistic machine learning, J. Comput. Phys. X 9 (2021) 100083, http://dx.doi.org/10.1016/j.jcpx.2020.100083.

[3] N.N. Vlassis, W. Sun, Sobolev training of thermodynamic-informed neural networks for interpretable elasto-plasticity models with level set hardening, Comput. Methods Appl. Mech. Engrg. 377 (2021) 113695, http://dx.doi.org/10.1016/j.cma.2021.113695.

[4] M. Lefik, D. Boso, B. Schrefler, Artificial Neural Networks in numerical modelling of composites, Comput. Methods Appl. Mech. Engrg. 198 (21) (2009) 1785–1804, http://dx.doi.org/10.1016/j.cma.2008.12.036, Advances in Simulation-Based Engineering Sciences – Honoring J. Tinsley Oden.

[5] D. Huang, J.N. Fuhg, C. Weißenfels, P. Wriggers, A machine learning based plasticity model using proper orthogonal decomposition, Comput. Methods Appl. Mech. Engrg. 365 (2020) 113008, http://dx.doi.org/10.1016/j.cma.2020.113008, arXiv:2001.03438.

[6] F. Ghavamian, A. Simone, Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network, Comput. Methods Appl. Mech. Engrg. 357 (2019) 112594, http://dx.doi.org/10.1016/j.cma.2019.112594.

[7] L. Wu, V.D. Nguyen, N.G. Kilingar, L. Noels, A recurrent neural network-accelerated multi-scale model for elasto-plastic heterogeneous materials subjected to random cyclic and non-proportional loading paths, Comput. Methods Appl. Mech. Engrg. 369 (2020) 113234, http://dx.doi.org/10.1016/j.cma.2020.113234.

[8] M. Mozaffar, R. Bostanabad, W. Chen, K. Ehmann, J. Cao, M.A. Bessa, Deep learning predicts path-dependent plasticity, Proc. Natl. Acad. Sci. 116 (52) (2019) 26414–26420, http://dx.doi.org/10.1073/pnas.1911815116, arXiv:https://www.pnas.org/content/116/52/26414.full.pdf.

[9] G. Chen, Recurrent neural networks (RNNs) learn the constitutive law of viscoelasticity, Comput. Mech. 67 (3) (2021) 1009–1019, http://dx.doi.org/10.1007/s00466-021-01981-y.

[10] A. Koeppe, F. Bamer, M. Selzer, B. Nestler, B. Markert, Explainable artificial intelligence for mechanics: physics-informing neural networks for constitutive models, 2021, arXiv:2104.10683.

[11] H.J. Logarzo, G. Capuano, J.J. Rimoli, Smart constitutive laws: Inelastic homogenization through machine learning, Comput. Methods Appl. Mech. Engrg. 373 (2021) 113482, http://dx.doi.org/10.1016/j.cma.2020.113482.

[12] M.B. Gorji, M. Mozaffar, J.N. Heidenreich, J. Cao, D. Mohr, On the potential of recurrent neural networks for modeling path dependent plasticity, J. Mech. Phys. Solids 143 (2020) 103972, http://dx.doi.org/10.1016/j.jmps.2020.103972.

[13] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707, http://dx.doi.org/10.1016/j.jcp.2018.10.045.

[14] E. Haghighat, M. Raissi, A. Moure, H. Gomez, R. Juanes, A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics, Comput. Methods Appl. Mech. Engrg. 379 (2021) 113741, http://dx.doi.org/10.1016/j.cma.2021.113741.

[15] M. Eghbalian, M. Pouragha, R. Wan, A physics-informed deep neural network for surrogate modeling in classical elasto-plasticity, 2022, http://dx.doi.org/10.48550/ARXIV.2204.12088.

[16] R. Arora, P. Kakkar, B. Dey, A. Chakraborty, Physics-informed neural networks for modeling rate- and temperature-dependent plasticity, 2022, http://dx.doi.org/10.48550/ARXIV.2201.08363.

[17] F. Masi, I. Stefanou, P. Vannucci, V. Maffi-Berthier, Thermodynamics-based Artificial Neural Networks for constitutive modeling, J. Mech. Phys. Solids 147 (2021) arXiv:2005.12183.

[18] F. Masi, I. Stefanou, Multiscale modeling of inelastic materials with Thermodynamics-based Artificial Neural Networks (TANN), Comput. Methods Appl. Mech. Engrg. 398 (2022) 115190, http://dx.doi.org/10.1016/j.cma.2022.115190.

[19] X. Liu, S. Tian, F. Tao, W. Yu, A review of artificial neural networks in the constitutive modeling of composite materials, Composites B 224 (2021) 109152, http://dx.doi.org/10.1016/j.compositesb.2021.109152.

[20] Z. Liu, C.T. Wu, M. Koishi, A deep material network for multiscale topology learning and accelerated nonlinear modeling of heterogeneous materials, Comput. Methods Appl. Mech. Engrg. 345 (2019) 1138–1168, http://dx.doi.org/10.1016/j.cma.2018.09.020, arXiv:1807.09829.

[21] Z. Liu, Cell division in deep material networks applied to multiscale strain localization modeling, CoRR abs/2101.07226, 2021, arXiv:2101.07226.

[22] P. Kerfriden, O. Goury, T. Rabczuk, S. Bordas, A partitioned model order reduction approach to rationalise computational expenses in nonlinear fracture mechanics, Comput. Methods Appl. Mech. Engrg. 256 (2013) 169–188, http://dx.doi.org/10.1016/j.cma.2012.12.004.

[23] M. Bessa, R. Bostanabad, Z. Liu, A. Hu, D.W. Apley, C. Brinson, W. Chen, W. 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, http://dx.doi.org/10.1016/j.cma.2017.03.037.

[24] J. Oliver, M. Caicedo, A. Huespe, J. Hernández, E. Roubin, Reduced order modeling strategies for computational multiscale fracture, Comput. Methods Appl. Mech. Engrg. 313 (2017) 560–595, http://dx.doi.org/10.1016/j.cma.2016.09.039.

[25] J.N. Fuhg, C. Böhm, N. Bouklas, A. Fau, P. Wriggers, M. Marino, Model-data-driven constitutive responses: Application to a multiscale computational framework, Internat. J. Engrg. Sci. 167 (2021) 103522, http://dx.doi.org/10.1016/j.ijengsci.2021.103522.

[26] F. Ghavamian, P. Tiso, A. Simone, POD–DEIM model order reduction for strain-softening viscoplasticity, Comput. Methods Appl. Mech. Engrg. 317 (2017) 458–479, http://dx.doi.org/10.1016/j.cma.2016.11.025.

[27] I.B.C.M. Rocha, F.P. van der Meer, L. Sluys, Efficient micromechanical analysis of fiber-reinforced composites subjected to cyclic loading through time homogenization and reduced-order modeling, Comput. Methods Appl. Mech. Engrg. 345 (2018) http://dx.doi.org/10.1016/j.cma.2018.11.014.

[28] I.B.C.M. Rocha, P. Kerfriden, F.P. van der Meer, Micromechanics-based surrogate models for the response of composites: A critical comparison between a classical mesoscale constitutive model, hyper-reduction and neural networks, Eur. J. Mech. A 82 (2020) 103995, http://dx.doi.org/10.1016/j.euromechsol.2020.103995.

[29] S. Vijayaraghavan, L. Wu, L. Noels, S.P.A. Bordas, S. Natarajan, L.A.A. Beex, Neural-network acceleration of projection-based model-order-reduction for finite plasticity: Application to RVEs, 2021, arXiv:2109.07747.

[30] T. Guo, O. Rokoš, K. Veroy, Learning constitutive models from microstructural simulations via a non-intrusive reduced basis method, Comput. Methods Appl. Mech. Engrg. 384 (2021) 113924, http://dx.doi.org/10.1016/j.cma.2021.113924.

[31] V.P. Nguyen, O. Lloberas-Valls, M. Stroeven, L.J. Sluys, Computational homogenization for multiscale crack modeling. Implementational and computational aspects, Internat. J. Numer. Methods Engrg. 89 (2) (2012) 192–226, http://dx.doi.org/10.1002/nme.3237.

[32] D.P. Kingma, T. Salimans, M. Welling, Variational dropout and the local reparameterization trick, 2015, http://dx.doi.org/10.48550/ARXIV.1506.02557.

[33] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, http://dx.doi.org/10.48550/ARXIV.1412.6980.

[34] J. Hernández, M. Caicedo, A. Ferrer, Dimensional hyper-reduction of nonlinear finite element models via empirical cubature, Comput. Methods Appl. Mech. Engrg. 313 (2017) 687–722, http://dx.doi.org/10.1016/j.cma.2016.10.022.

[35] T. Mori, K. Tanaka, Average stress in matrix and average elastic energy of materials with misfitting inclusions, Acta Metall. 21 (5) (1973) 571–574, http://dx.doi.org/10.1016/0001-6160(73)90064-3.

[36] J.D. Eshelby, R.E. Peierls, The determination of the elastic field of an ellipsoidal inclusion, and related problems, Proc. R. Soc. A 241 (1226) (1957) 376–396, http://dx.doi.org/10.1098/rspa.1957.0133, arXiv:https://royalsocietypublishing.org/doi/pdf/10.1098/rspa.1957.0133.

[37] C. Nguyen-Thanh, V.P. Nguyen, A. de Vaucorbeil, T. Kanti Mandal, J.-Y. Wu, Jive: An open source, research-oriented C++ library for solving partial differential equations, Adv. Eng. Softw. 150 (2020) 102925, http://dx.doi.org/10.1016/j.advengsoft.2020.102925.

[38] F.P. van der Meer, Mesolevel modeling of failure in composite laminates: Constitutive, kinematic and algorithmic aspects, Arch. Comput. Methods Eng. 19 (3) (2012) 381–425, http://dx.doi.org/10.1007/s11831-012-9076-y.

[39] A. Melro, P. Camanho, F. Andrade Pires, S. Pinho, Micromechanical analysis of polymer composites reinforced by unidirectional fibres: Part I – Constitutive modelling, Int. J. Solids Struct. 50 (11) (2013) 1897–1905, http://dx.doi.org/10.1016/j.ijsolstr.2013.02.009.

[40] F.P. van der Meer, Micromechanical validation of a mesomodel for plasticity in composites, Eur. J. Mech. A Solids 60 (2016) 58–69, http://dx.doi.org/10.1016/j.euromechsol.2016.06.008.