Physically recurrent neural network for rate and path-dependent heterogeneous materials in a finite strain framework

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

DOI
10.1016/j.mechmat.2024.105145

Publication date
2024

Document Version
Final published version

Published in
Mechanics of Materials

Mechanics of Materials 198 (2024) 105145

Contents lists available at ScienceDirect

Mechanics of Materials

journal homepage: www.elsevier.com/locate/mecmat

Research paper

Physically recurrent neural network for rate and path-dependent heterogeneous materials in a finite strain framework

M.A. Maia [*], I.B.C.M. Rocha, D. Kovačević, F.P. van der Meer

Delft University of Technology, Department of Civil Engineering and Geosciences, PO Box 5048, 2600 GA, Delft, The Netherlands

ARTICLE INFO

Dataset link: https://github.com/SLIMM-Lab/prnn3d, https://github.com/SLIMM-Lab/pyprnn

Keywords:
Neural networks
Heterogeneous materials
Rate-dependency
Path-dependency
Constitutive model

ABSTRACT

In this work, a hybrid physics-based data-driven surrogate model for the microscale analysis of heterogeneous material is investigated. The proposed model benefits from the physics-based knowledge contained in the constitutive models used in the full-order micromodel by embedding the material models in a neural network. Following previous developments, this paper extends the applicability of the physically recurrent neural network (PRNN) by introducing an architecture suitable for rate-dependent materials in a finite strain framework. In this model, the homogenized deformation gradient of the micromodel is encoded into a set of deformation gradients serving as input to the embedded constitutive models. These constitutive models compute stresses, which are combined in a decoder to predict the homogenized stress, such that the internal variables of the history-dependent constitutive models naturally provide physics-based memory for the network. To demonstrate the capabilities of the surrogate model, we consider a unidirectional composite micromodel with transversely isotropic elastic fibers and elasto-viscoplastic matrix material. The extrapolation properties of the surrogate model trained to replace such micromodel are tested on loading scenarios unseen during training, ranging from different strain-rates to cyclic loading and relaxation. Speed-ups of three orders of magnitude with respect to the runtime of the original micromodel are obtained.

1. Introduction

Owing to their high flexibility and potential to reduce computational costs, machine learning (ML) techniques are becoming increasingly popular in solid mechanics. These techniques can be especially useful in micromechanical and multiscale analysis, where the accurate representations of complex materials are often compromised by their notoriously high computational costs. Complex heterogeneous materials can be modeled on a lower scale, the microscale, through a so-called Representative Volume Element (RVE), a micromodel assumed to statistically represent the material behavior. At that scale, classical constitutive models can be conveniently employed to describe the behavior of each of the constituents. This allows for an accurate representation of intricate phenomena in the composite material behavior without the need for assumptions about the macroscopic material behavior. The generality of the method, however, is not without trade-offs. Large micromodels, fine meshes, and path and rate-dependent materials are some of the features that result in exceedingly high computational costs. A common approach to tackle this issue is to replace the micromodel altogether with a surrogate model that reproduces the relation between the homogenized strains and stresses at a lower computational cost. In applications involving path-dependent materials, variations of Recurrent Neural Networks (RNN) are the most popular choice, but other surrogate modeling strategies built on Gaussian Processes (GPs) and dimensionality reduction techniques (*e.g.* Proper Orthogonal Decomposition (POD) and Hyper-reduction methods) have also showed potential for reducing the computational costs (Oliver et al., 2017; Ghavamian et al., 2017; Rocha et al., 2019, 2021).

When it comes to neural networks, the more complex architectures derived from RNNs, such as Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM), are the predominant choice at present. These models can handle sequential data through mechanisms that propagate information from previous to later states when processing a sequence (*e.g.* an entire path of ε-σ pairs). Several works showcase their potential in modeling path-dependent behavior for both homogeneous (Heider et al., 2020) and heterogeneous materials (Wu and Noels, 2022; Logarzo et al., 2021; Gorji et al., 2020; Mozaffar et al., 2019). Nevertheless, several unresolved issues and challenges remain. One of them lies in the fact that in spite of the similarity between the role of the hidden state in RNNs and the internal variables in a constitutive model, the network mechanism is still regarded as a black-box and insights into any latent physical patterns learned by the network are thus far limited

[*] Corresponding author.
E-mail address: M.AlvesMaia@tudelft.nl (M.A. Maia).

https://doi.org/10.1016/j.mechmat.2024.105145
Received 5 April 2024; Received in revised form 19 July 2024; Accepted 2 September 2024
Available online 4 September 2024

to simple settings (*e.g.* homogeneous material in 1D problems Koeppe et al., 2021; Liu et al., 2023; Bhattacharya et al., 2023).

Another pressing issue in these networks is their limited ability to extrapolate. This is usually tackled with ever larger training sets and intricate design of experiments that aim to uniformly/densely cover the space of strain paths. A complicating aspect is the curse of dimensionality. In that regard, frameworks based on RNNs and variations are typically exemplified in 1D or 2D problems, but even in those cases a large variety of loading/unloading cases is required to cover similar paths and patterns encountered in actual microscale simulations. Recent works (Ghane et al., 2023; Cheung and Mirkhalaf, 2024) illustrate the hurdles with predicting loading types different from the ones used for training. In Ghane et al. (2023), a strategy based on transfer learning is employed to improve the training performance of LSTMs and GRUs and overcome feature sparsity issues. For this, the authors train a network on data generated using a random walk strategy, and then use the optimized parameters in the initialization of a second network trained to predict cyclic loading. Cheung and Mirkhalaf (2024) extend that idea to train GRUs with multi-fidelity data, helping reduce the computational cost of generating large high-fidelity training datasets.

An alternative approach to uncover the black-box nature of these methods is to introduce physics knowledge into the ML-based model. Following that philosophy, Physics-Informed Neural Networks (PINNs) are likely the biggest exponent. Although these networks have been initially designed to solve partial differential equations, the idea of enriching the loss function with extra terms to enforce physics constraints has quickly found its way into the material modeling community. For instance, to predict displacement and stress fields, in addition to terms corresponding to the Neumann and Dirichlet boundary conditions in the loss function of a PINN, one could also include physics-based constraints such as Karush–Kuhn–Tucker conditions when modeling plasticity, as done by Haghighat et al. (2021) or the evolution of the plastic strain-rate for viscoplastic materials, as shown in Arora et al. (2022). In spite of the additional information, Haghighat et al. (2021) report no benefit in using PINNs as forward solvers and Arora et al. (2022) comment on the degrading performance in extrapolation.

Another way to leverage physical consistency in NNs is to encode the physical knowledge directly in the architecture design (Masi and Stefanou, 2022; Eghbalian et al., 2023; Garanger et al., 2023). For instance, in Masi and Stefanou (2022), the authors proposed a framework where stresses and dissipation are obtained through the differentiation of the learned energy potential function. Strategic architectural choices can also help enforce a specific behavior, as done in Garanger et al. (2023). In that work, tensor-based features and activation functions are used in feed-forward and GRUs to enforce material symmetries.

Moving away from the recurrency mechanisms in RNNs, transformers rely on self-attention mechanisms to extract correlations among the elements within a sequence. These models have shown improved performance in comparison to other state-of-the-art methods in capturing long-range dependencies in language processing problems (Vaswani et al., 2017), but have only recently been applied in the computational homogenization field to predict the response of composite materials with elastoplastic behavior (Zhongbo and Hien, 2024; Pitz and Pochiraju, 2024). Beyond the positive assessment on the accuracy and efficiency of the trained models on the online evaluations, a common thread in these works (Zhongbo and Hien, 2024; Pitz and Pochiraju, 2024) encompass the need of very large datasets (ranging from dozens to hundreds of thousands of curves), the difficulty of training models with millions of parameters and the critical scaling of computational memory space required for both the offline and online phases as the sequence length increases.

When dealing with materials with time-dependency, the extra dimensionality related to strain-rate sensitivity adds a new depth to the problem. For clarity, we distinguish time or rate-dependency from path-dependency as the former refers to behavior that is dependent on the duration and speed of the loading, while the latter refers to behavior dependent on the loading sequence and history. In a broader sense, both are framed as history-dependent. In some works, the strain-rate (Wen et al., 2021) and/or the time increment (Ge and Tagarielli, 2021) have been explicitly included in the feature space. In others, a fixed time increment is considered (Ghavamian and Simone, 2019; Chen, 2021). Another interesting strategy was proposed by Liu et al. (2023), where a forward Euler discretization was employed to make the stress prediction independent from the time discretization using two feed-forward NNs. The first model learns the rate of change of a set of internal variables learned from the data based on the current strain and the previous set of internal variables, while the second predicts the stress based on the current strain and the internal variables learned by the first NN. In a follow-up work, Zhang and Bhattacharya (2024) explore how iterated learning can help improve the accuracy of these models in multiscale applications through the inclusion of strain-stress curves extracted from a macroscopic problem of interest, as well as their transferability to other problems.

Finally, Eghtesad et al. (2023) proposed a framework based on the dual potential function to describe rate-dependent viscoplastic flow response in metals. The authors take advantage of input-convex NNs to enforce thermodynamic consistency and leverage automatic differentiation to compute gradients of the output with respect to the inputs, which are used for solving the implicit time-stepping algorithm employed in their elasto-viscoplastic model. Nevertheless, the method is not suitable for FE simulations yet as arbitrary loading and boundary conditions can take place and only uniaxial deformations were considered.

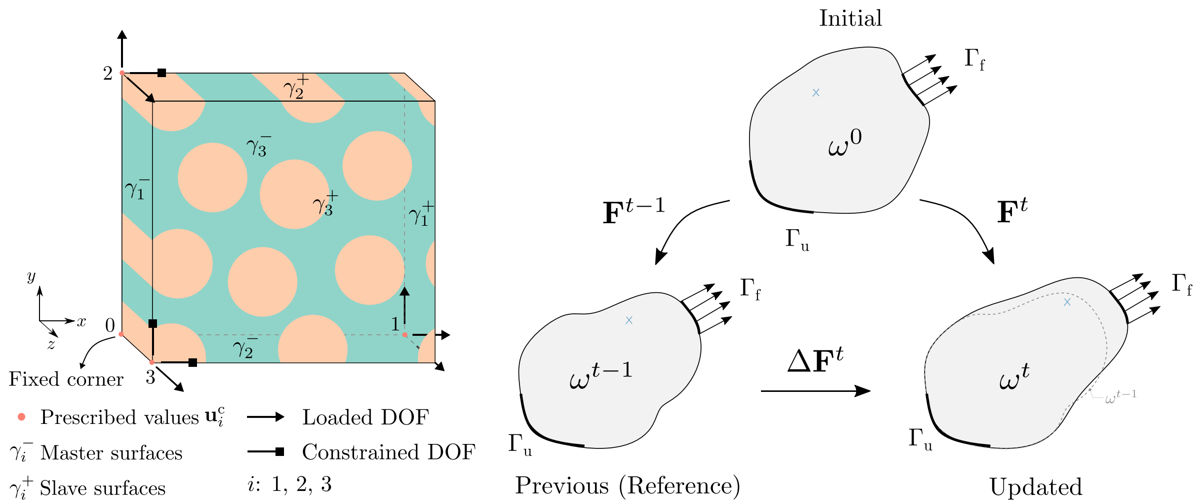
In all of these works, to train a surrogate for rate-dependence the training data needs to account not only for a good coverage of strains but also strain-rates. To deal with time-dependency in a more seamless manner, we propose to expand the applicability of the approach presented in Maia et al. (2023), namely the Physically Recurrent Neural Network (PRNN). In that work, a network with embedded physics-based material models was used to accelerate multiscale analysis of path-dependent heterogeneous materials. The core idea is that the macroscopic strain can be encoded into a set of strains for (fictitious) material points, from which the stress is computed using the same material models and properties as in the micromodel. With these stresses, a decoder is applied to obtain the macroscopic stresses in a homogenization-like step.

A key element in the proposed architecture consists of letting the material model that evaluates the fictitious material point stress also handle the evolution of its own internal variables. This way, the network inherits the assumptions built into the material models used in the micromodel without the need for additional trainable parameters or mechanisms to reproduce history-dependent behavior. In a related work (Rocha et al., 2023), we explore this idea from a different perspective. Instead of learning how to dehomogenize the macroscopic strain, we learn how material properties evolve in time and let the material model be the decoder of a single fictitious material point subjected to the macroscopic strain.

In our previous paper (Maia et al., 2023), the PRNN was demonstrated to work for micromodels with rate-independent plasticity, capturing loading–unloading behavior without seeing it during training. It is anticipated that the same approach can capture rate-dependence. In this paper, we apply the PRNN approach to micromodels where the polymer matrix is described with the Eindhoven Glassy Polymer (EGP) model, an advanced elasto-viscoplastic material model for polymers. For this purpose, the following features are added with respect to the previous work:

- time-dependent material behavior;
- a finite strain formulation;
- generalization to 3D space.

We show how these new and non-trivial features are incorporated into the network and demonstrate that the benefits of the PRNN approach successfully transfer to a much more complex class of models.



(a) Boundary conditions of periodic micromodel ω with constrained dofs to avoid rigid-body motion

(b) Initial (undeformed), previously converged (reference) and updated configurations

Fig. 1. Micromodel and scheme of configurations used in the updated Lagrangian framework.

2. Microscale analysis

This work focuses on the homogenized behavior of a RVE of a microscopic material with both path and time-dependency. For notation purposes, the superscripts Ω and ω refer to the homogenized (macroscopic) and microscopic quantities, respectively. Let ω denote the RVE domain and consider that periodic boundary conditions (PBC) are applied to simulate the behavior of a macroscopic bulk material point, as depicted in Fig. 1(a). In the absence of body forces, the updated Lagrangian formulation, illustrated in Fig. 1(b), can be defined by the weak statement of equilibrium

$$\underbrace{\int_{\omega} \mathbf{B}^{\mathrm{T}} \boldsymbol{\sigma}\, d\omega}_{\mathbf{f}^{\mathrm{int}}} - \underbrace{\int_{\Gamma_{\mathrm{u}}} \mathbf{N}^{\mathrm{T}} \mathbf{t}_{\mathrm{p}}\, d\Gamma}_{\mathbf{f}^{\mathrm{ext}}} = \mathbf{0} \tag{1}$$

where $\mathbf{N}$ is a matrix with the shape functions used to interpolate the nodal displacements $\mathbf{a}$, $\mathbf{B}$ is the strain–displacement matrix with the gradients of the shape functions with respect to the current coordinates $\mathbf{x}$, $\boldsymbol{\sigma}$ is the Cauchy stress, $\mathbf{t}_{\mathrm{p}}$ are the tractions prescribed on the boundary of the domain Γ_{f} (see Fig. 1).

With the domain discretized in a Finite Element (FE) mesh, the displacements at the nodal values, known as the degrees of freedom (DOF), are used to describe the displacement field of the micromodel $\mathbf{u} = \mathbf{N}\mathbf{a}$. In this method, Eq. (1) is solved iteratively as

$$\mathbf{r} = \mathbf{f}^{\mathrm{int}} - \mathbf{f}^{\mathrm{ext}} = \mathbf{0} \tag{2}$$

where $\mathbf{r}$ is the residual vector that vanishes once equilibrium is reached. The iterative procedure involves linearization of $\mathbf{f}^{\mathrm{int}}$ with respect to the DOF vector. In the geometrically nonlinear formulation, this linearization requires accounting for the dependence of $\mathbf{B}$ from Eq. (1) on the displacements through a geometric contribution to the stiffness matrix.

The stress in Eq. (1) is related to the deformations with a constitutive model $\mathcal{C}^{\omega}$, which, in general, can be described by

$$\boldsymbol{\sigma}, \boldsymbol{\alpha} = \mathcal{C}^{\omega}\left(\mathbf{F}, \boldsymbol{\alpha}^{t-1}, \Delta t\right) \tag{3}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^{t-1}$ are the history variables that account for path and rate-dependency at the current and previous time step, Δt is the time increment and $\mathbf{F}$ is the deformation gradient

$$\mathbf{F} = \mathbf{I} + \nabla\mathbf{u} \tag{4}$$

where $\nabla\mathbf{u}$ represents the gradient of the microscopic displacements. Since the deformation gradient is calculated with respect to the initial configuration, its increment can also be easily computed from the current and previous deformation states

$$\Delta\mathbf{F} = \mathbf{F}\mathbf{F}_{t-1}^{-1}. \tag{5}$$

For rate-dependent materials, the stress depends on $\Delta\mathbf{F}$ as well as $\mathbf{F}$, which can be achieved with Eq. (3) if $\mathbf{F}_{t-1}$ is included in the material history $\boldsymbol{\alpha}$. Upon convergence, the homogenized stresses can be averaged out by integrating the microscopic stresses over the volume ω:

$$\boldsymbol{\sigma}^{\Omega} = \frac{1}{|\,\omega\,|} \int_{\omega} \boldsymbol{\sigma}\, \mathrm{d}\omega. \tag{6}$$

2.1. Constitutive models

In this work, we consider a composite micromodel made of unidirectional fibers embedded in a matrix material. To describe the constitutive behavior of the matrix, the EGP model is used, while for the fibers, a hyperelastic transversely isotropic model is assigned. These consist of the same choices adopted in Kovačević and van der Meer (2022), where a thorough validation of the material models was carried out for a carbon/PEEK composite material. Here, we only highlight the main aspects of their formulation and focus on how to incorporate them in a PRNN.

The fiber constitutive law is based on the one developed by Bonet and Burton (1998) with slight modifications (Kovačević and van der Meer, 2022). The constitutive model derives from the strain energy density function and can be split into two components, an isotropic part with a neo-Hookean potential and a transversely isotropic part, with both depending on the right Cauchy–Green deformation tensor

$$\mathbf{C} = \mathbf{F}^{\mathrm{T}}\mathbf{F}. \tag{7}$$

The EGP model for the matrix material consists of a rate and path-dependent elasto-viscoplastic, isotropic, 3D constitutive law. In this model, no explicit yield surface is needed since an Eyring-based viscosity function evolves with the stress applied, leading to the viscoplastic flow of the material. The Cauchy stress calculated by the EGP is composed of three contributions: hydrostatic, hardening and driving stress. While the first two parts are defined in more simple terms as they do not depend on the internal variables, in the third part, where viscoplasticity is introduced, a further decomposition can be considered. In this case, the multiple contributions to the driving stress correspond to different molecular (relaxation) processes. Each relaxation process is represented with a series of Maxwell models (modes) connected in parallel, with

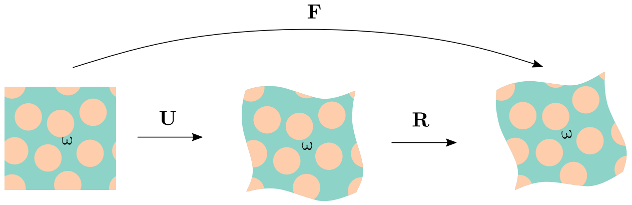


Fig. 2. Right polar decomposition on deformation gradient $\mathbf{F}$ resulting in the stretch and rotation tensors $\mathbf{U}$ and $\mathbf{R}$, respectively.

a shear modulus in the elastic spring and a stress-dependent viscosity in the dashpot. Here, a single relaxation process is considered and represented with 1 mode.

For any of the models discussed so far, a helpful tool to deal with the high-dimensionality of the deformation gradient is the polar decomposition theorem. The theorem states that any deformation gradient $\mathbf{F}$ can be uniquely decomposed into the product of two other tensors: a symmetric one $\mathbf{U}$ and an orthogonal one $\mathbf{R}$, as $\mathbf{F} = \mathbf{R}\mathbf{U}$. These two tensors have physical interpretations and are closely related to the principle of material objectivity or material frame indifference. In short, the symmetric tensor represents the deformation (*i.e.* stretches and shear) and the orthogonal tensor represents a rigid body rotation. When applied in this sequence, the final configuration obtained is the same as the one obtained if the deformation gradient was applied directly.

The particular order of stretch and rotation is known as right polar decomposition and is illustrated in Fig. 2. From these interpretations and considering the principle of material frame indifference, which states the material response is independent of the observer, one can rewrite stresses as

$$\boldsymbol{\sigma}_{\mathrm{U}}, \boldsymbol{\alpha} = \mathcal{C}^{\omega}\ (\mathbf{U}, \boldsymbol{\alpha}^{t-1}, \Delta t) \tag{8}$$

$$\boldsymbol{\sigma}_{\mathrm{F}} = \mathbf{R}\ \boldsymbol{\sigma}_{\mathrm{U}}\ \mathbf{R}^{\mathrm{T}} \tag{9}$$

where $\boldsymbol{\sigma}_{\mathrm{U}}$ are the unrotated stresses and $\boldsymbol{\sigma}_{\mathrm{F}}$ are the stresses in the original frame of reference.

3. Physically recurrent neural network

In this section, we present the new architecture of the Physically Recurrent Neural Network (PRNN) to be used in a 3D finite strain framework for micromodels with path and rate-dependent behavior. Having the network in Maia et al. (2023) as the starting point, we highlight and motivate the main changes in comparison to the 2D formulation. In that work, a NN composed of a data-driven encoder, a material layer with embedded physics-based material models and a data-driven decoder is proposed. The data-driven parts learn how the homogenized strain can be dehomogenized and distributed among a small set of fictitious material points and how the stress obtained in these material points can be homogenized again, respectively. With the same core idea, we propose a set of modifications to extend such model to the current application. For an extended introduction to PRNNs, the interested reader is referred to Maia et al. (2023).

Before diving into the details of the novel architecture, training aspects and its use as a constitutive model, we highlight an important change in its input with respect to the 2D formulation in Maia et al. (2023). Here, the surrogate is trained to learn the mapping from stretch (path) to unrotated stress (Eq. (8)) and let it be embedded between decomposition and rotation operations to recover the stress in the original frame, as illustrated in Fig. 3, instead of mapping deformation gradient to rotated stress directly. With this, the dimensionality of the feature space of the PRNN is reduced from 9 to 6 independent components, alleviating the sampling effort required for training.

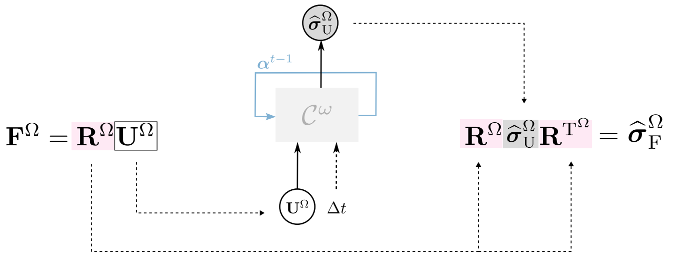

$\widehat{\boldsymbol{\sigma}}_{\mathrm{U}}^{\Omega}$
$\boldsymbol{\alpha}^{t-1}$
$\mathcal{C}^{\omega}$
$\mathbf{F}^{\Omega} = \mathbf{R}^{\Omega}\boxed{\mathbf{U}^{\Omega}}$
$\mathbf{R}^{\Omega}\widehat{\boldsymbol{\sigma}}_{\mathrm{U}}^{\Omega}\mathbf{R}^{\mathrm{T}^{\Omega}} = \widehat{\boldsymbol{\sigma}}_{\mathrm{F}}^{\Omega}$
$\mathbf{U}^{\Omega}$
Δt

Fig. 3. Use of PRNN in a general full-order solution setting with $\mathbf{F}^{\Omega}$ and $\widehat{\boldsymbol{\sigma}}_{\mathrm{F}}^{\Omega}$ as input and output, respectively.

3.1. Encoder

The encoder comprises all parameters and operations that convert the homogenized stretch tensor into local fictitious deformation gradients. In the general PRNN architecture illustrated in Fig. 4, these correspond to the blue connections. In our previous work, the encoder consisted of an arbitrary number of hidden layers fully connected, while in this work a custom layer is proposed to ensure that physical constraints related to the definition of the strain measure are met. Two challenges arise from working with the deformation gradient instead of the small strain vector. Firstly, with the deformation gradient or the stretch, the undeformed state corresponds to the identity and not a null tensor.

In a regular dense layer, if a given set of weights $\mathbf{W}$ were to be applied on the undeformed stretch tensor (*i.e.* $\mathbf{U}^{\Omega} = \mathbf{I}$), the resulting matrix $\mathbf{W}\mathbf{U}^{\Omega}$ would be different from the identity and therefore generate stresses when it should not. To address that, we need to make a few changes to the encoder, starting with the way we treat the input. Now, instead of applying weights to transform a vector with dimension 6, we work on the actual tensor $\mathbf{U}^{\Omega}$ that is 3 × 3. Note that this is only a *reshaping* operation, and no additional features are needed to fill the tensor.

With that, the weights connecting $\mathbf{U}^{\Omega}$ to the inputs of the material layer can be applied in a similar fashion to the fictitious material points, in groups, to generate the deformation gradients used in that layer. In this case, for each point, a 3 × 3 weight matrix is needed. Another important change to ensure the zero stress-state comes from the definition of the deformation gradient (see Eq. (4)). Based on that, we subtract the identity matrix from the homogenized stretch tensor and only then apply the weights to the remaining values. After that transformation, we add the identity back and obtain the final deformation gradient.

Secondly, because the deformation gradient determinant represents the change in volume from the undeformed to the current configuration, the local deformation gradients learned by the network should have strictly positive determinants. One way to avoid negative determinants consists in ensuring that the determinant of the weight matrices applied on $\mathbf{U}^{\Omega} - \mathbf{I}$ to obtain the fictitious local deformation gradients are always positive. This is done by imposing a structured weight matrix $\mathbf{W}_j$ originated from a Cholesky decomposition for each subgroup j. The determinant of the decomposed triangular matrices simplifies to the multiplication of their diagonal elements, so positivity is therefore ensured by applying a softplus function to those diagonal entries. In this case, only 6 learnable parameters are associated to each fictitious material point. The scheme in Fig. 5 summarizes how the local strain of one fictitious material point is obtained after the proposed changes.

3.2. Material layer

This layer contains the embedded physics-based constitutive models, arranged into a series of fictitious integration (material) points. Because a material model is not a scalar-valued function like typical

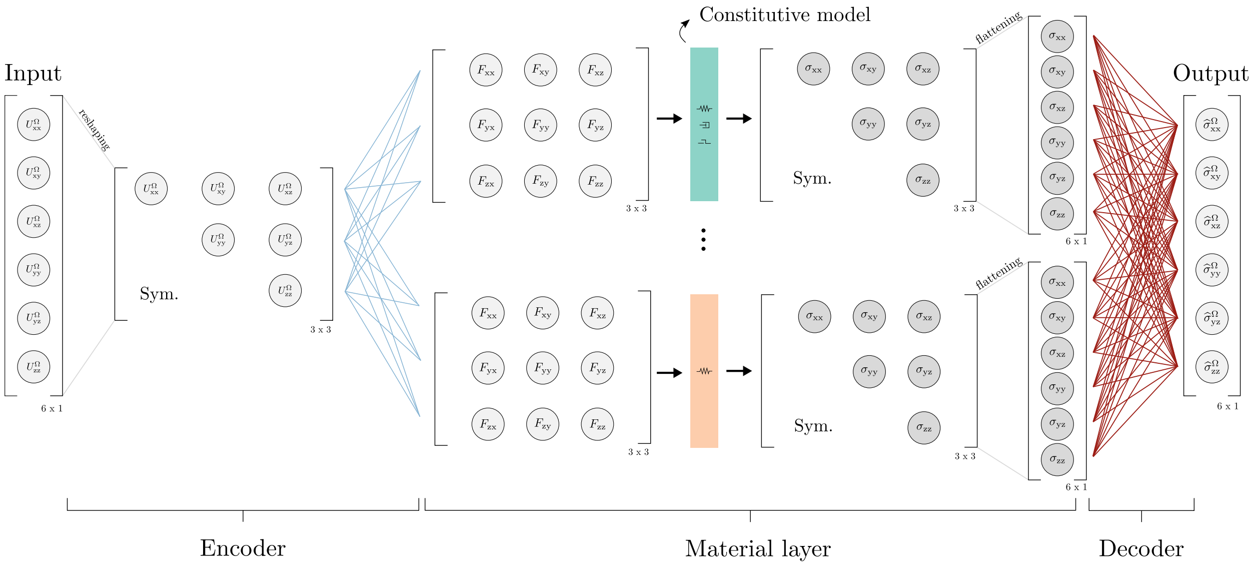


Fig. 4. New architecture of PRNN for finite strain framework. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

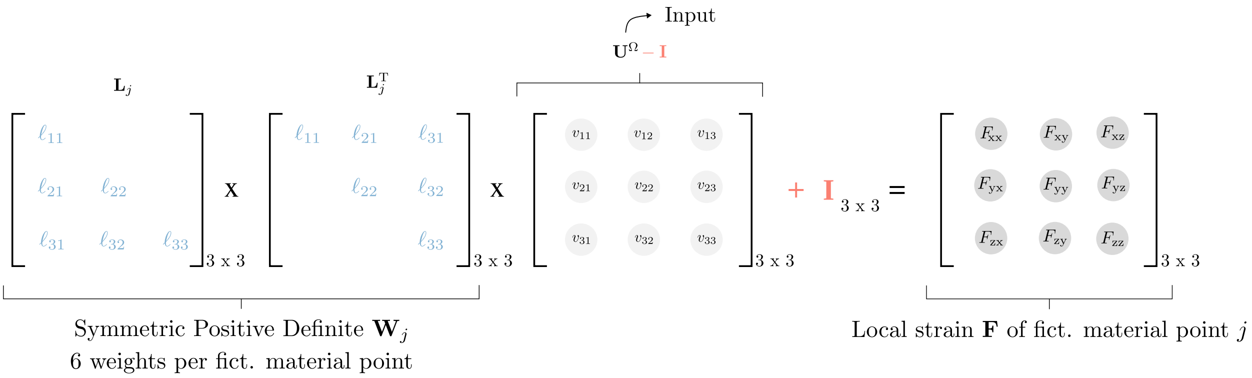

Input
$\mathbf{U}^{\Omega} - \mathbf{I}$
$\mathbf{L}_j$
$\mathbf{L}_j^{\mathrm{T}}$
X
X
3 x 3
$+ \mathbf{I}_{3 \times 3} =$
Symmetric Positive Definite $\mathbf{W}_j$
6 weights per fict. material point
Local strain $\mathbf{F}$ of fict. material point j

Fig. 5. Encoder architecture applied to obtain the local strain of a fictitious material point j based on the input $\mathbf{U}^{\Omega}$.

activation functions (*e.g.* sigmoid, tanh, relu, etc.), a special architecture is required. In that sense, an important change compared to Maia et al. (2023) is the way neurons are interpreted. Here, we group them together in m subgroups, each consisting of a tensor with the same order tensor and dimensions as the deformation gradient in the input layer (3 × 3 for the present investigation in three dimensions), whereas in Maia et al. (2023) the subgroups consists of vectors of length 3, representing the strain vector in 2D. In this arrangement, each subgroup corresponds to one *fictitious material point*. The basic idea is that the values reaching the subgroup can be seen as a local deformation learned by the encoder, which will then be evaluated by one of the constitutive models used in the full-order solution with the same material properties.

Once a given constitutive model with its respective material properties is assigned to the subgroup j, say $\mathcal{C}_j^{\omega}$, the next step is to use it to obtain the stresses and the updated internal variables (if any). These internal variables are present in rate and path-dependent material models and are the core of the physics-based memory of the proposed network. However, rate and path-independent constitutive models can also be used in the material layer without further adaptations. A brief discussion on the choice of the constitutive models used in this layer follows at the end of the section.

Consider that $\mathcal{C}_j^{\omega}$ takes as input the deformation gradient $\mathbf{F}$, the internal variables from previous time step $\boldsymbol{\alpha}^{t-1}$ and the increment of time Δt. In the first time step, the internal variables of all material points are properly initialized based on the undeformed state $\boldsymbol{\alpha}_j^0$. In every time step, the current stresses $\boldsymbol{\sigma}$ and updated internal variables $\boldsymbol{\alpha}$ of each subgroup are obtained. These variables are preserved in each subgroup so that in the following load step, when a new $\mathbf{F}$ is fed to the material point, the history of the material can be updated accordingly. A representation of this workflow is shown in Fig. 6. Note that the "flattening" operation transforming the 3 × 3 tensor into a vector with only the 6 independent components, is analogous to the reshaping operation used at the encoder. This condensation does not imply in loss of information since the Cauchy stress tensor is symmetric.

An important aspect illustrated in Fig. 6 is that no additional time-related features or trainable parameters are needed to learn the time-dependence. The network learns the strain distribution over the fictitious material points through the encoder, which works the same for all constitutive models. The time increment Δt is passed to the rate-dependent material as additional input, but it has the same value for all material points as would be done in the micromodel simulation. By directly employing the same material models and properties considered in the micromodel with internal variables that naturally follow physics-based assumptions, we can capture the rate and path-dependent behavior in a more straightforward way. With RNNs, the mechanisms behind the evolution of internal variables need to be learnt from the data.

Finally, the user is left with the choice of which constitutive model to employ in the material points. Our recommendation is to employ all nonlinear constitutive models used in the micromodel with their respective known material properties. To illustrate that, consider the

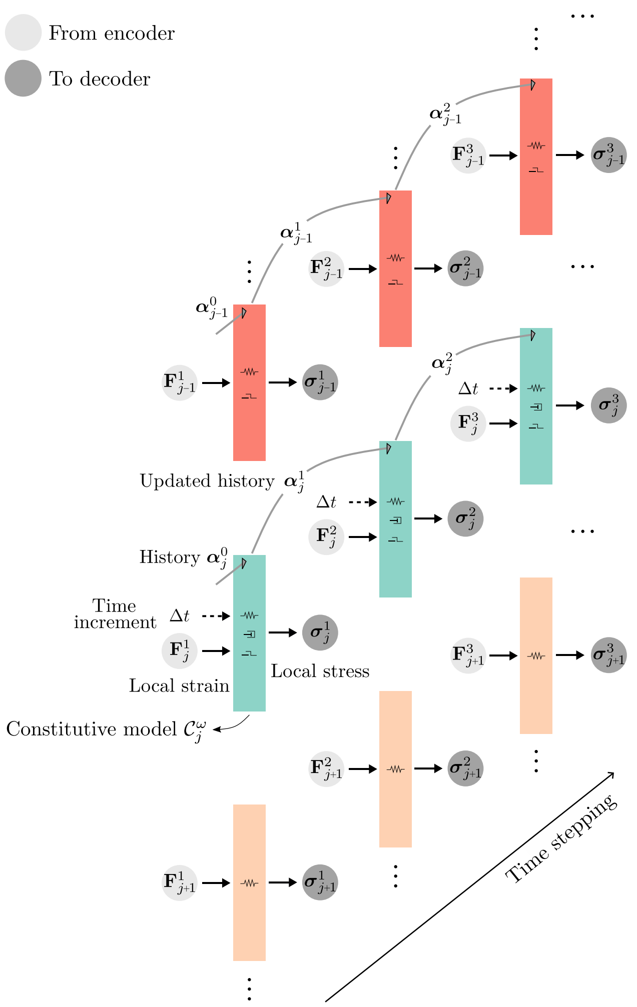

From encoder
To decoder
$\boldsymbol{\alpha}^2_{j-1}$
$\mathbf{F}^3_{j-1}$
$\boldsymbol{\sigma}^3_{j-1}$
$\boldsymbol{\alpha}^1_{j-1}$
$\mathbf{F}^2_{j-1}$
$\boldsymbol{\sigma}^2_{j-1}$
$\boldsymbol{\alpha}^0_{j-1}$
$\mathbf{F}^1_{j-1}$
$\boldsymbol{\sigma}^1_{j-1}$
$\boldsymbol{\alpha}^2_{j}$
Δt
$\mathbf{F}^3_{j}$
$\boldsymbol{\sigma}^3_{j}$
Updated history $\boldsymbol{\alpha}^1_{j}$
Δt
$\mathbf{F}^2_{j}$
$\boldsymbol{\sigma}^2_{j}$
History $\boldsymbol{\alpha}^0_{j}$
Time increment Δt
$\mathbf{F}^1_{j}$
$\boldsymbol{\sigma}^1_{j}$
Local strain
Local stress
Constitutive model $\mathcal{C}^{\omega}_{j}$
$\mathbf{F}^3_{j+1}$
$\boldsymbol{\sigma}^3_{j+1}$
$\mathbf{F}^2_{j+1}$
$\boldsymbol{\sigma}^2_{j+1}$
$\mathbf{F}^1_{j+1}$
$\boldsymbol{\sigma}^1_{j+1}$
Time stepping

Fig. 6. Scheme of fictitious material points unrolled in time, each colored box corresponding to a different constitutive model. From top to bottom: path-dependent, path and rate-dependent, and path and rate-independent constitutive models. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

composite micromodel studied in the numerical examples of this work, in which an orthotropic hyperelastic model is used to describe the fibers and an elasto-viscoplastic model for the matrix. Since both models include nonlinearity in their formulations, we include both types in the material layer. In addition to that, in the present case, each model has distinctive behavior in terms of path and rate-dependence, which emphasizes the importance of both in the network. This topic is further discussed in Section 3.4 along with other training aspects and model selection procedure, including the definition of the proportion of the constitutive models in this layer.

3.3. Decoder

The decoder comprises all network parameters that work on the outputs of the material layer to obtain the homogenized stresses $\hat{\boldsymbol{\sigma}}^{\Omega}$. Note that because the outputs of the material layer consist of the stresses from the fictitious material points, the role of the decoder parameters is well aligned with the actual full-order solution. In the micromodel, once the full-field of stresses is obtained, the homogenized stresses are obtained by averaging the stresses over the entire domain. Here, instead of integrating the field with hundreds or thousands of integration points, only a few fictitious material points contribute to the homogenized response where the relative contributions of each fictitious point are learnt from data.

For that purpose, an arbitrary number of neurons and layers can be used. In this work in particular, for a more direct analogy with the homogenization process, a single dense layer with linear activation and physics-motivated modifications is considered. In this way, the weights of the output layer reflect the relative contribution of each of the material points to the predicted homogenized response. In the actual micromodel, weights come from a numerical integration scheme and are strictly positive. To reflect that, an absolute function $\rho(\cdot)$ is applied element-wise on the weights of the decoder $\mathbf{W}_{\mathrm{d}}$. For the present architecture (see Fig. 4), it then follows that the predicted homogenized stress is given by

$$\hat{\boldsymbol{\sigma}}^{\Omega} = \rho\left(\mathbf{W}_{\mathrm{d}}\right)\mathbf{a} \tag{10}$$

where $\mathbf{a}$ corresponds to the concatenation of local stresses from the material points.

In addition to that, we also investigate the use of a sparsification approach, where instead of having a regular dense layer that connects all components of the local stress tensor to the predicted homogenized stress, only the component-wise contributions are taken into account, as illustrated in Fig. 7. For instance, only the stresses σ_{xy} from each of the subgroups are weighted in for obtaining $\hat{\sigma}^{\Omega}_{\mathrm{xy}}$. This sparsification also brings the decoder closer to the actual homogenization procedure, in which stresses are averaged component-wise.

3.4. Training aspects and error metrics

The goal of the optimization procedure is to minimize a loss function that quantifies how close the network's prediction are from the actual solution. In this work, the standard loss function based on the mean square error (MSE) is used:

$$L = \frac{1}{N_{\mathrm{train}}}\sum_{t=1}^{N_{\mathrm{train}}}\frac{1}{2}\left\|\mathrm{vec}\left(\boldsymbol{\sigma}^{\Omega}_{t}\right) - \mathrm{vec}\left(\hat{\boldsymbol{\sigma}}^{\Omega}_{t}\left(\mathbf{U}^{\Omega}_{t}, \mathbf{W}, \mathbf{W}_{\mathrm{d}}\right)\right)\right\|^{2} \tag{11}$$

where N_{train} is the number of loading paths used for training, $\mathbf{W}$ and $\mathbf{W}_{\mathrm{d}}$ are the network parameters for the encoder and decoder, respectively, and vec$(\cdot)$ corresponds to the Voigt representation of the homogenized stress tensor, which consists of 6 components in the 3D case (*i.e.* the "flattening" mentioned in the previous sections). From that, one can compute the gradients of the loss function with respect to the trainable parameters using a backpropagation procedure and then update those accordingly, for which we use the Adam optimizer (Kingma and Ba, 2014).

The backpropagation here follows the same methodology as in Maia et al. (2023). Note that the gradients of the parameters in the decoder can be obtained based on the conventional backpropagation procedure, but for the ones in the encoder, backpropagation through time is needed. This is a vital aspect of the training and stems from the path-dependency of the material models embedded in the material layer. For completeness, we include the expression for computing the gradients of the weights in the encoder at time step t for a given loading path:

$$\begin{aligned}\frac{\partial L^{t}}{\partial \mathbf{W}_{j}} = \frac{\partial L}{\partial \hat{\boldsymbol{\sigma}}^{\Omega}_{t}}\frac{\partial \hat{\boldsymbol{\sigma}}^{\Omega}_{t}}{\partial \boldsymbol{\sigma}^{t}_{j}}&\left\{\frac{\partial \boldsymbol{\sigma}^{t}_{j}}{\partial \mathbf{F}^{t}_{j}}\frac{\partial \mathbf{F}^{t}_{j}}{\partial \mathbf{W}_{j}} + \frac{\partial \boldsymbol{\sigma}^{t}_{j}}{\partial \boldsymbol{\alpha}^{t}_{j}}\frac{\partial \boldsymbol{\alpha}^{t}_{j}}{\partial \mathbf{F}^{t}_{j}}\frac{\partial \mathbf{F}^{t}_{j}}{\partial \mathbf{W}_{j}}\right.\\ &\left.+\frac{\partial \boldsymbol{\sigma}^{t}_{j}}{\partial \boldsymbol{\alpha}^{t}_{j}}\sum_{\tilde{t}=1}^{t-1}\left[\left(\prod_{\tilde{i}=\tilde{t}+1}^{t}\frac{\partial \boldsymbol{\alpha}^{\tilde{i}}_{j}}{\partial \boldsymbol{\alpha}^{\tilde{i}-1}_{j}}\right)\frac{\partial \boldsymbol{\alpha}^{\tilde{t}}_{j}}{\partial \mathbf{F}^{\tilde{t}}_{j}}\frac{\partial \mathbf{F}^{\tilde{t}}_{j}}{\partial \mathbf{W}_{j}}\right]\right\}\end{aligned} \tag{12}$$

where $\mathbf{W}_{j}$ corresponds to the weights associated to the material point j. The gradients related to the internal variables are evaluated using central finite differences. However, if the material models are implemented with automatic differentiation support (*e.g.* PyTorch and TensorFlow), these gradients and dependencies can be automatically taken into account with tools such as Autograd and GradientTape, as done with off-the-shelf RNNs.

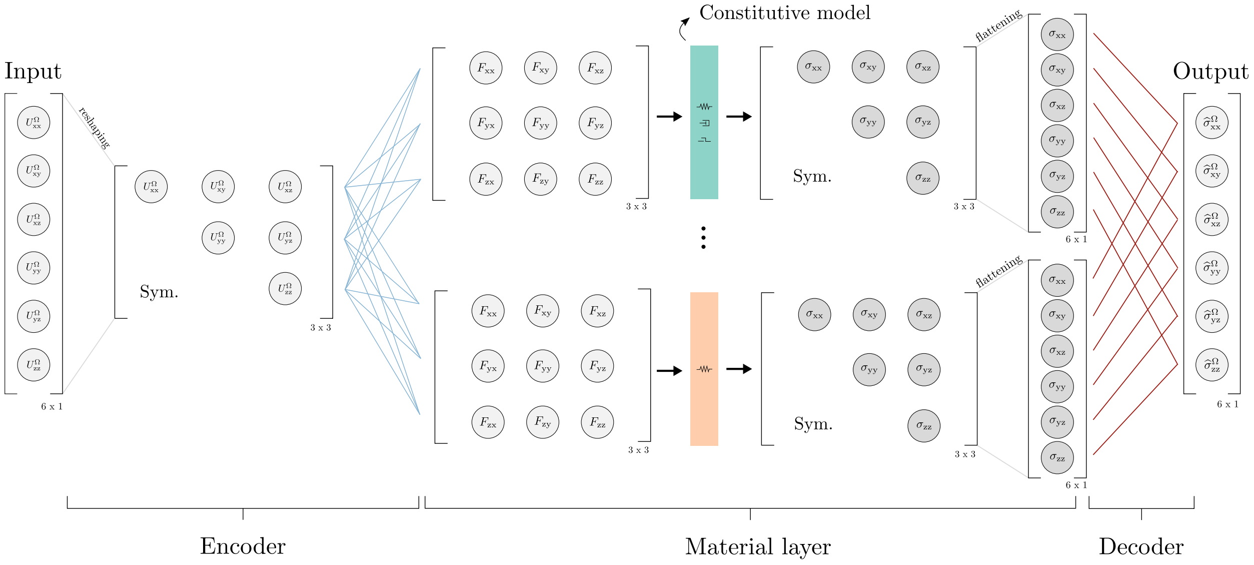

Input
reshaping
U^{Ω}_{xx}
U^{Ω}_{xy}
U^{Ω}_{xz}
U^{Ω}_{yy}
U^{Ω}_{yz}
U^{Ω}_{zz}
6 x 1
Sym.
3 x 3
F_{xx} F_{xy} F_{xz}
F_{yx} F_{yy} F_{yz}
F_{zx} F_{zy} F_{zz}
3 x 3
Constitutive model
σ_{xx} σ_{xy} σ_{xz}
σ_{yy} σ_{yz}
Sym. σ_{zz}
3 x 3
flattening
σ_{xx}
σ_{xy}
σ_{xz}
σ_{yy}
σ_{yz}
σ_{zz}
6 x 1
Output
$\hat{\sigma}^{\Omega}_{xx}$
$\hat{\sigma}^{\Omega}_{xy}$
$\hat{\sigma}^{\Omega}_{xz}$
$\hat{\sigma}^{\Omega}_{yy}$
$\hat{\sigma}^{\Omega}_{yz}$
$\hat{\sigma}^{\Omega}_{zz}$
6 x 1
Encoder
Material layer
Decoder

Fig. 7. PRNN with sparse decoder.

A potential issue in training with Eq. (11) is the large variations of values across the multiple outputs due to the orthotropy of the composite material with high stiffness contrast. In such scenario, one component can disproportionately dominate over the others, leading to unstabilities in the training process and overall poor performance. To mitigate that, each component of $\boldsymbol{\sigma}^{\Omega}$ is normalized to $[-1,\,1]$ as follows:

$$\sigma^{\Omega}_{(\cdot)\,\text{norm}} = 2\left(\frac{\sigma^{\Omega}_{(\cdot)} - \min \sigma^{\Omega}_{(\cdot)}}{\max \sigma^{\Omega}_{(\cdot)} - \min \sigma^{\Omega}_{(\cdot)}}\right) - 1 \tag{13}$$

where max refers to the maximum absolute homogenized stress values of the component $(\cdot)$ in the training data and min is the negative of that value. The symmetric bounds in each of the components ensures that the zero-stress state from the material points will be reflected in the homogenized stress. Furthermore, to preserve the role of the decoder as the homogenization-like step, the normalization in Eq. (13) is also applied to the local stresses from the fictitious material points. This ensures that all material point stresses are within the same range expected at the output layer. Lastly, no normalization is considered for the inputs, since the range of the features are similar and, more importantly, are compatible with the range expected by the models in the material layer.

For the model selection and performance assessment, we consider two error metrics:

$$\begin{aligned} &\text{Absolute error}: \frac{1}{N_{\text{paths}}}\sum_{i=1}^{N_{\text{paths}}}\frac{1}{L_{\text{path}}}\sum_{t=1}^{L_{\text{path}}}\left\|\operatorname{vec}(\boldsymbol{\sigma}^{\Omega}_{t}) - \operatorname{vec}(\hat{\boldsymbol{\sigma}}^{\Omega}_{t})\right\| \\ &\text{Relative error}: \frac{1}{N_{\text{paths}}}\sum_{i=1}^{N_{\text{paths}}}\frac{1}{L_{\text{path}}}\sum_{t=1}^{L_{\text{path}}}\frac{\left\|\operatorname{vec}(\boldsymbol{\sigma}^{\Omega}_{t}) - \operatorname{vec}(\hat{\boldsymbol{\sigma}}^{\Omega}_{t})\right\|}{\left\|\operatorname{vec}(\boldsymbol{\sigma}^{\Omega}_{t}) + \boldsymbol{\varepsilon}\right\|} \end{aligned} \tag{14}$$

where N_{paths} refers to the number of loading paths in the validation/test sets, L_{path} is the length of each path and $\boldsymbol{\varepsilon}$ is a stabilizing term with the same dimensions as $\operatorname{vec}(\boldsymbol{\sigma}^{\Omega}_{t})$ filled with 10^{-8} used to avoid division by zero.

To reduce the number of hyper-parameters to be tuned and keep the model selection as simple and straightforward as possible, we define a minibatch as 2 paths, the stopping criterion as the maximum number of epochs (1000) and use the recommend default settings from Kingma and Ba (2014) in the Adam optimizer, including its standard learning rate decay update per iteration. When training the network, the validation set is evaluated every 50 epochs, and the best set of parameters is updated only if the current error is lower than the historical lowest validation error, thus mitigating the risk of overfitting. Further details on the model selection procedure, including the definition of the material layer size, the type of decoder (dense or sparse), and the size of the training set, are presented in Section 5.

When choosing which constitutive models to assign to the fictitious material points, we follow the idea of including all sources of nonlinearity. In this case, both the fiber and the matrix constitutive models qualify. At this point, it is worth highlighting another aspect that makes having both models in the network important. Although the fiber constitutive model adopted in this work only shows non-linearity at very large strains, in our case, it is also the one introducing the transversal isotropy in the micromodel and has distinct behavior from the matrix in terms of path and rate-dependency. Those unique characteristics need to be present in the network so that the encoder and decoder can leverage them into the homogenized stress response.

Related to that is the definition of how many of the fictitious material points are assigned to each of the models. This proportion itself is a hyper-parameter, but to reduce the amount of variables in the upcoming studies, we define a fixed splitting ratio. The hyperelastic and elasto-viscoplastic models correspond to 25 % and 75 % of the material points, respectively, rounding the number of hyperelastic models up when the total number of points is even but not divisible by 4. The higher proportion of points associated to the elasto-viscoplastic model is rooted in the fact that this is the most complex constitutive model in the micromodel, from which we expect higher expressibility. Furthermore, it is also a model with internal variables, 24 per point to be precise, which effectively work as the physics-based memory of the network. Thus, we expect to achieve good performance with smaller networks (*i.e.* more parsimonious PRNNs) compared to splitting ratios that favor hyperelastic models. Other than the difference in the material layer size itself, we expect no significant changes in the overall accuracy of the network granted model selection has been performed correctly.

Finally, we highlight that the choice of constitutive models, regardless of history-dependence, and their splitting ratio in the material layer do not affect the total number of trainable parameters in the network. For the dense decoder architecture depicted in Fig. 4, the number of trainable parameters is given by the total number of fictitious material points multiplied by 42 (6 from the encoder and 36 from the decoder). In the sparse decoder archictecture illustrated in Fig. 7, this number drops to 12 trainable parameters per fictitious material point (6 from the encoder and 6 from the decoder).

3.5. Use as constitutive model

For incorporating the present network as a constitutive model in a microscale analysis that takes as input the homogenized deformation gradient ($\mathbf{F}^{\Omega}$) and the increment of time (Δt) and outputs homogenized stresses $\hat{\boldsymbol{\sigma}}_{\mathrm{F}}^{\Omega}$, a few additional steps are introduced. First, the polar decomposition theorem is applied on the deformation gradient in order to obtain the rotation $\mathbf{R}^{\Omega}$ and the stretch tensors $\mathbf{U}^{\Omega}$. Once the stretch tensor is obtained and the increment of time is known, the network is used to predict the unrotated stresses $\hat{\boldsymbol{\sigma}}_{\mathrm{U}}^{\Omega}$ in a forward pass. The rotation tensor is then used to transform the predicted unrotated stresses back into the rotated system.

Obtaining the tangent stiffness matrix is not as straightforward. In this framework, the jacobian of the network is only one part of the tangent stiffness matrix expression for the entire mapping between rotated stresses and the deformation gradient

$$\frac{\partial \hat{\boldsymbol{\sigma}}_{\mathrm{F}}^{\Omega}}{\partial \mathbf{F}^{\Omega}} = \frac{\partial \hat{\boldsymbol{\sigma}}_{\mathrm{F}}^{\Omega}}{\partial \hat{\boldsymbol{\sigma}}_{\mathrm{U}}^{\Omega}} \frac{\partial \hat{\boldsymbol{\sigma}}_{\mathrm{U}}^{\Omega}}{\partial \mathbf{U}^{\Omega}} \frac{\partial \mathbf{U}^{\Omega}}{\partial \mathbf{F}^{\Omega}} + \frac{\partial \hat{\boldsymbol{\sigma}}_{\mathrm{F}}^{\Omega}}{\partial \mathbf{R}^{\Omega}} \frac{\partial \mathbf{R}^{\Omega}}{\partial \mathbf{F}^{\Omega}} \tag{15}$$

where the partial derivatives of the homogenized rotation and stretch tensors with respect to the homogenized deformation gradient are given by the expressions derived by Chen and Wheeler (1993) and $\partial \hat{\boldsymbol{\sigma}}_{\mathrm{U}}^{\Omega} / \partial \mathbf{U}^{\Omega}$ is given by performing a complete backward pass through the network. Moreover, the partial derivative of the stresses with respect to the unrotated stresses is given by

$$\frac{\partial \hat{\boldsymbol{\sigma}}_{\mathrm{F}}^{\Omega}}{\partial \hat{\boldsymbol{\sigma}}_{\mathrm{U}}^{\Omega}} = \mathbf{R}^{\Omega} \otimes \mathbf{R}^{\Omega} \tag{16}$$

where $\otimes$ represents the Kronecker product between two second-order tensors of dimensions $n_{\mathrm{rank}} \times n_{\mathrm{rank}}$, resulting in a second-order tensor of dimensions $n_{\mathrm{rank}}\ n_{\mathrm{rank}} \times n_{\mathrm{rank}}\ n_{\mathrm{rank}}$. Finally, the partial derivative of the stresses with respect to the rotation tensor are evaluated as

$$\frac{\partial \hat{\boldsymbol{\sigma}}_{\mathrm{F}}^{\Omega}}{\partial \mathbf{R}^{\Omega}} = \bar{\mathbf{P}}\ (\mathbf{I} \otimes \hat{\boldsymbol{\sigma}}_{\mathrm{U}}^{\Omega}\ \mathbf{R}^{\Omega^{\mathrm{T}}}) + (\mathbf{I} \otimes \mathbf{R}^{\Omega}\ \hat{\boldsymbol{\sigma}}_{\mathrm{U}}^{\Omega})\ \mathbf{P} \tag{17}$$

where $\bar{\mathbf{P}}$ and $\mathbf{P}$ are two permutation matrices given by

$$\begin{aligned} \bar{\mathbf{P}} &= \sum_{i,j} E_{ij} \otimes E_{ij} \\ \mathbf{P} &= \sum_{i,j} E_{ij} \otimes E_{ji} \end{aligned} \tag{18}$$

with E_{ij} being a null matrix except for the unit value at $E_{i,j}$.

4. Data generation

In general, surrogate models need to be trained with an extensive amount of data covering several types of loading. This is because it is virtually impossible to have fine control over what types of loading the micromodel will experience upfront even in the simplest scenarios. Therefore, to investigate how well the proposed network can generalize to unseen scenarios, a variety of loading functions and methods for generating the loading paths are considered.

First, we define the geometry and the discretization of the micromodel. In this case, the same composite RVE used in Kovačević and van der Meer (2022), and illustrated in Fig. 8, with 9 fibers embedded in a matrix material is adopted. The material models and properties assigned to each of the phases also follow from that work with a minor change in one of the material properties of the matrix. The reinforcements are assumed to be carbon fibers and can be described by the hyperelastic, transversely isotropic material model developed by Bonet and Burton (1998). For the matrix, the elasto-viscoplastic EGP model is considered with the relaxation spectrum now consisting of one mode (the first). Both of these models are briefly discussed in Section 2.1, but for further details on their implementation and numerical validation in the 3D finite strain framework, the reader is directed to the reference paper (Kovačević and van der Meer, 2022).

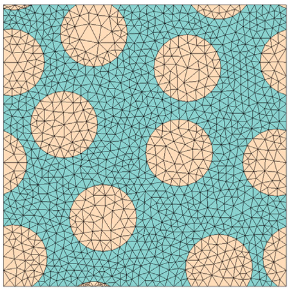


Fig. 8. Geometry and mesh discretization of micromodel used to generate the data.

To generate the data, two strategies are devised, one producing proportional loading paths, and the other non-proportional loading paths. We use the first type to train and test the network, while the second is reserved for testing only. By proportional we refer to curves in which the loading direction is fixed. For this, we adopt the arc-length formulation with indirect displacement control derived in Rocha et al. (2020), in which a constant unit load vector is considered and the additional constraint consists in the unsigned sum of the controlled displacements. For stress measures based on the undeformed state, that also entails a constant stress ratio.

For creating proportional paths, three main ingredients are needed: the loading direction $\mathbf{n}$, the loading function λ and the time increment Δt. In the previous work (Maia et al., 2023), basic load cases (*e.g.* uniaxial and biaxial tension and compression, transverse and longitudinal shear, etc.) were used for training PRNNs subjected to general stress states. Here, due to the increased problem dimensionality, we train with a more general approach of random loading directions. For each path, the unit load vector is obtained by sampling values from 6 independent Gaussian distributions ($X \sim \mathcal{N}(0, 1)$) and normalizing them to a unit vector, one for each prescribed corner displacement. As for the time increment, we set it to $\Delta t = 1\,\mathrm{s}$ for all time steps. Fixing the time increments allows for a straightforward assessment of the ability of the network to extrapolate to unseen strain-rates.

The last ingredient to create the proportional curves is the loading function λ. We use the two loading functions depicted in Figs. 10(a) and 10(b) as pre-defined monotonic and non-monotonic curves, respectively. Although useful for testing, this non-monotonic set is not as valuable for training since all curves follow the same unloading/reloading behavior. An alternative with more unloading variety is to sample λ from a Gaussian Process (GP) with $X \sim \mathcal{N}(\mu, \sigma^2)$ and covariance function given by

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp\left(-\frac{1}{2\ell^2} \|\mathbf{x}_p - \mathbf{x}_q\|^2\right) \tag{19}$$

where $\mathbf{x}_p$ and $\mathbf{x}_q$ are the time step indices of the sequence of loading function values, σ_f^2 is the variance and ℓ is the length scale. These hyper-parameters control the smoothness and how large the unsign sum of the controlled displacements can be, and are tuned to obtain smooth loading functions, as the ones illustrated in Fig. 10(c) (see Fig. 10).

To create a more diverse set in terms of strain-rate compared to the curves using a single pre-defined loading function, for the proportional GP-based curves, the time increment of each path is drawn from a bounded uniform distribution $\Delta t \sim U\,(0.01\,\mathrm{s}, 100\,\mathrm{s})$.

Fig. 9 shows a summary of the three loading types discussed so far ordered by their level of complexity. In this work, we train with two of them, namely monotonic curves (in blue) and proportional GP-based curves (in brown). For testing, we take a step further and generate non-proportional and non-monotonic paths. These are the most complex paths considered and also employ GPs in their formulation. To create these curves, first we switch to a displacement control method and follow a similar procedure as the one employed in Maia et al. (2023).

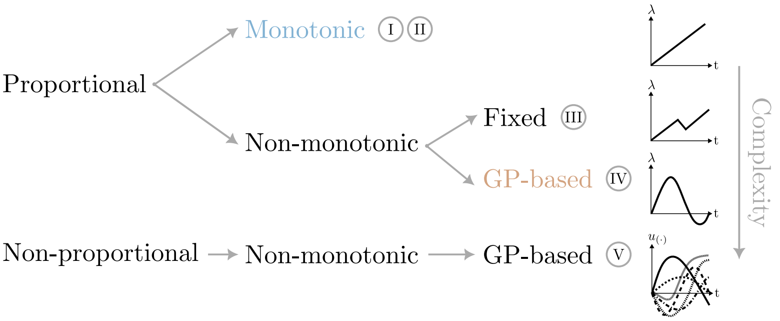
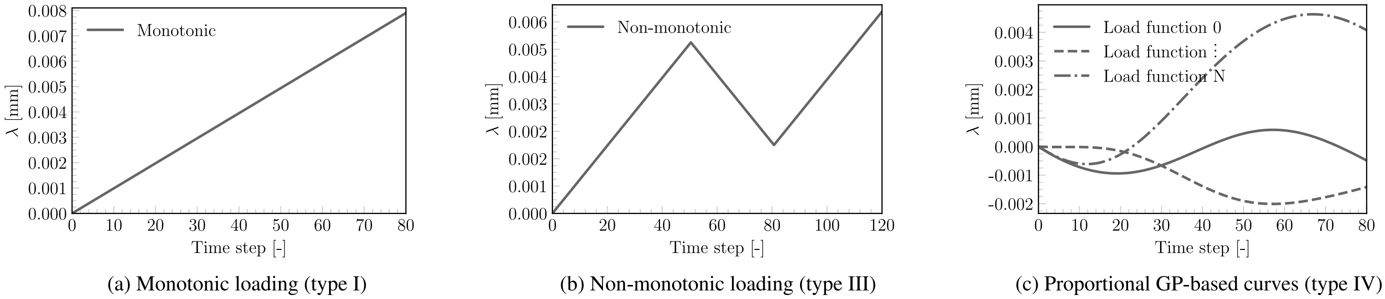


Fig. 9. Scheme of loading types considered in this work, with colored types being used for training and testing, while remaining are for testing. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Monotonic loading (type I)

(b) Non-monotonic loading (type III)

(c) Proportional GP-based curves (type IV)

Fig. 10. Loading functions used to create proportional loading paths.

Here, we sample the displacements at the controlling nodes from 6 independent GPs, allowing unloading/reloading to take place at different times across the components of the homogenized deformation gradient. This is illustrated in the bottom right plot of Fig. 9, where independent $u_{(\cdot)}$–t functions are plotted for the different components.

For reference, all the types of loading paths studied in the following section are listed below in ascending order of complexity:

- Type I: proportional and monotonic loading path. The direction $\mathbf{n}$ is generated randomly, the loading function λ is as illustrated in Fig. 10(a) with step size $\Delta\lambda = 1 \times 10^{-4}$ mm, and $\Delta t = 1$ s. In the following sections, data sets using this type of path carry the subscript "mono";
- Type II: proportional and monotonic loading path with same loading function and step size as Type I, but different strain-rate. Data sets with this type of path carry the subscript "mono" and two variations of superscript, "faster" and "slower". To generate those, $\Delta t = 0.01$ s and $\Delta t = 100$ s are used, respectively;
- Type III: proportional and non-monotonic loading path with fixed unloading/reloading behavior λ as illustrated in Fig. 10(b) with $\Delta\lambda = 1 \times 10^{-4}$ mm, and $\Delta t = 1$ s. Data sets with this type of path have the subscript "unl" and the superscript "fixed";
- Type IV: proportional and non-monotonic loading path with loading function given by a GP with variable step size, and $\Delta t \sim U\,(0.01\,\text{s}, 100\,\text{s})$. In this case, each loading path follows a different unloading/reloading function. Fig. 10(c) illustrates some of the loading functions generated by this approach with $\ell = 30$ and $\sigma_f^2 = 1{\times}10^{-5}$ as the hyper-parameters of the GP. Data sets with this type of path have the subscript "unl" and the superscript "prop. GP";
- Type V: non-proportional and non-monotonic loading path with GPs to describe the displacements, and $\Delta t \sim U\,(0.01\,\text{s}, 100\,\text{s})$. Each controlled displacement in the micromodel is assigned to an independent GP, from which we sample smooth and random functions with variable step size. In this case, the hyper-parameters of the GPs are $\ell = 30$ and $\sigma_f^2 = 2.5 \cdot 10^{-7}$, with the exception of the variance of the GP associated to the displacement in the fiber direction, which is 10 times smaller than the others to prevent excessively high stress values that can dominate the homogenized stress state. Data sets with this type of path have the subscript "unl" and the superscript "non-prop GP".

5. Numerical experiments

In this section, the accuracy of the network is assessed in a set of numerical experiments. The goal is to illustrate the extrapolation properties of the method given the different training strategies. The test cases cover loading directions and strain-rates different from those seen in training, as well as complex unloading/reloading cases. Since we are focusing on the network's accuracy only, the following sections deal with the stretch and the unrotated stresses as their inputs and outputs, respectively.

5.1. Model selection

First, two preliminary studies are carried out for model selection. The first one is used to choose between sparse and dense decoders (see Section 3.3), while the second is focused on defining the material layer size. The comparison is carried out with varying size of the material layer each time considering the largest training set with monotonic loading paths. For each combination of decoder architecture and material layer size, 10 random initializations of the PRNN are considered. In each of them, the training set $\mathcal{D}_{\text{mono}}$ consists of 100 monotonic curves randomly selected from a pool of 1000 curves of the same type (Type I). For validation, a fixed set $\mathcal{V}_{\text{mono}}$ with 100 monotonic curves is considered. In Fig. 11, the colored areas correspond to the envelope with the highest and lowest absolute errors for each combination, along with the average errors represented by the solid lines with markers. In all cases, we emphasize that the reported errors over validation and test sets correspond to the network parameters associated to the historical best performance during training, as discussed in Section 3.4. A marked difference in accuracy between the two types of decoder for all range of material layer size over $\mathcal{V}_{\text{mono}}$ is observed. Therefore, in the remainder of this paper, all networks have a sparse decoder.

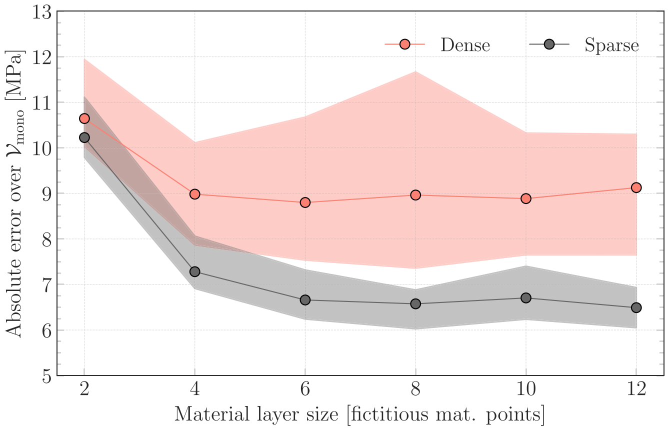


Fig. 11. Envelope of highest and lowest absolute validation error from 10 PRNNs trained on $\mathcal{D}_{\text{mono}}$ = {144 monotonic curves} over validation set $\mathcal{V}_{\text{mono}}$ = {100 monotonic curves} with different material layer sizes and decoder architectures. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The second model selection step is focused on finding an optimal size for the material layer. For this purpose, the material layer size is varied considering a range of different training set sizes. Note that at this stage there is no direct comparison between the two training strategies since their training and validation are of matching types. Similarly to the plot in Fig. 11, in Fig. 12 we show the envelope of best and worst performances, along with the average absolute errors over the validation set $\mathcal{V}_{(\cdot)}$, which this time consists of either monotonic or proportional GP-based curves. In the following sections, we use the PRNNs with material layer size of 8 for both cases, which corresponds to the point where errors are either the lowest among all training sets or have negligible difference with respect to larger layer sizes.

5.2. Monotonic loading

As first test scenario, we consider a test set $\mathcal{T}_{\text{mono}}$ consisting of 100 monotonic curves in random and unseen directions (type I). We evaluate the networks trained on monotonic (type I) and GP-based paths (type IV) over that test set for different training set sizes. Fig. 13(a) shows the lowest absolute and relative errors for both strategies, along with the envelope of absolute errors from 10 initializations. As more data is considered, the error bounds shrink and an optimal training set size can be identified around 72 curves. Although the difference in the lowest errors is still significant, 6.2 MPa (5.4 %) vs. 7.6 MPa (6.3 %), more data translates into marginal gain to both. In the breakdown of the error per component in Fig. 13(b), the largest differences in the accuracy are in the $\sigma^{\Omega}_{\text{yy}}$ and $\sigma^{\Omega}_{\text{zz}}$ components. The overall performance gap between the two training strategies in this scenario is expected since we are testing on the same loading behavior used to generate the training data of one of the strategies. Another aspect to be considered is the fact that the proportional GP-based curves reach lower strain ranges compared to the monotonic paths for the same number of time steps and step size (see Fig. 13).

To illustrate the difference in performance, we select a curve from $\mathcal{T}_{\text{mono}}$ with an absolute error close to the best performances from both training strategies. In this case, the prediction error on the curve shown in Fig. 14 is around 5.5 MPa and 7.2 MPa for the networks trained on monotonic and proportional GP-based curves, respectively. Note that the accuracy loss stands out more in the components with lower stress magnitude such as $\sigma^{\Omega}_{\text{xx}}$ and $\sigma^{\Omega}_{\text{zz}}$. An explanation for that comes from the choice of the loss function, the mean squared error. Recall that although normalization of the outputs is considered to balance the difference between stress magnitudes among the components, the MSE remains an absolute metric error. As such, values on the higher end of the normalized range can still dominate the loss, leading to a better fit. Neverthess, satisfatory agreement is observed in the remaining components with the network trained on monotonic data, while the network trained on proportional GP-based curves shows more significant errors.

Table 1
Summary of lowest absolute errors from 10 PRNNs trained on different types of curves over test sets $\mathcal{T}_{\text{mono}}$, $\mathcal{T}_{\text{faster}}$ and $\mathcal{T}_{\text{slower}}$.

Training loading type	Monotonic		Prop. GP	
Training set size	72	144	72	144
Abs. error over $\mathcal{T}_{\text{mono}}$ [MPa]	6.2	6.1	7.6	6.6
Abs. error over $\mathcal{T}_{\text{faster}}$ [MPa]	6.6	6.5	7.5	6.7
Abs. error over $\mathcal{T}_{\text{slower}}$ [MPa]	5.7	5.5	7.1	6.2

5.3. Monotonic loading with different strain-rates

Next, we test the ability of the PRNN to capture rate-dependency. For that, two new test sets are considered, $\mathcal{T}_{\text{slower}}$ and $\mathcal{T}_{\text{faster}}$, with 100 curves each again in unseen directions (type II). In the first one, the time increment Δt is set to be 100 times larger than the reference one (1 s) used for generating the monotonic curves for training, and in the second, the time increment is 100 times smaller. The best performances from the 10 PRNNs trained on different types and numbers of curves are summarized in Table 1. Again, the slight advantage of the networks trained on monotonic curves is expected since the loading function in both test sets remains monotonic and reaches similar strain levels. As a result, networks trained with proportional GP-based curves show greater benefit from larger sets, as was the case in the previous assessment. Similarly, since the gain is still relatively small compared to doubling the training set size, we continue the analysis with the smaller set for both types.

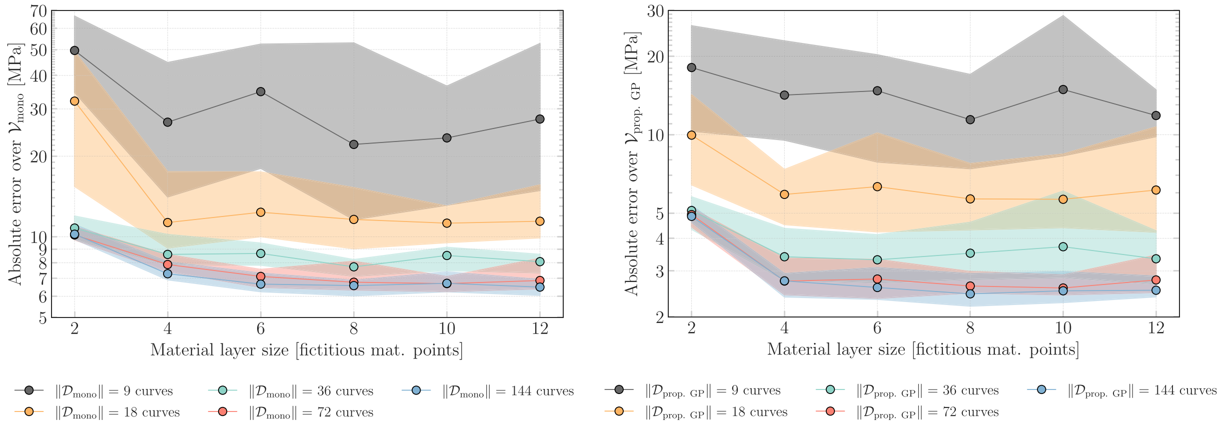
To illustrate the rate-dependent behavior, we use the best networks over each of the test sets and select a representative curve from them to visualize the effect of the different strain-rates (see Fig. 15). This is an important milestone of this contribution, especially considering that these strain rates are far from the reference values considered to generate the monotonic curves. The rate dependency in this case is a natural outcome of the elastoviscoplastic model used in the material layer. Encoding rate-dependence in the material layer allows for reproducing this effect without training for it. This is most evident from the error values reported in Table 1, where the test errors are of similar magnitude for test sets with unseen strain-rates as for the test set with the same strain rate as used for the training data. In contrast to modern RNNs, our latent variables have physical interpretation, and, more importantly, evolve according to the same physics-based assumptions considered in the micromodel.

5.4. Unloading/reloading behavior

In this section, three types of unloading/reloading paths are tested with data sets from type III, IV and V. In all cases, every scenario is assessed based on a test set with 100 curves. Networks trained with both training strategies (based on type I and type IV curves) are evaluated.

5.4.1. Predefined unloading/reloading function

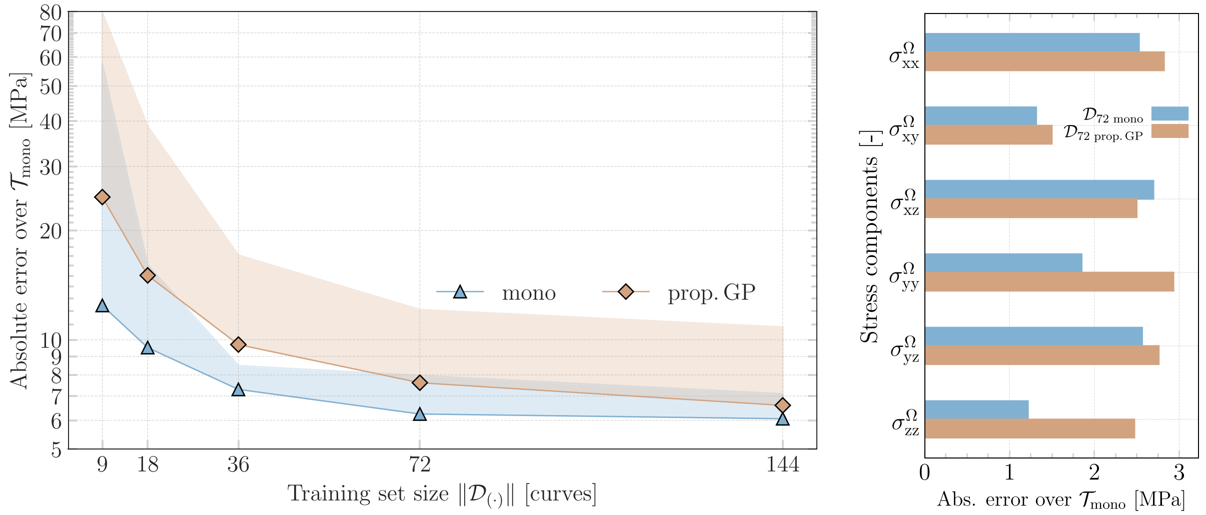
Table 2 presents the lowest error from 10 networks over the test set of proportional curves with pre-defined unloading $\mathcal{T}^{\text{fixed}}_{\text{unl}}$ (type III). It can be observed that both training strategies lead to similar performances. Note that although the networks trained on proportional GP-based curves can still benefit from a larger training set, we continue the experiments with 72 curves as the gain in accuracy from doubling the training set is minimal. It is also interesting how the networks trained on monotonic paths are still slightly more accurate than the



(a) $\mathcal{V}_{\text{mono}}$ = {100 monotonic loading paths}

(b) $\mathcal{V}_{\text{prop. GP}}$ = {100 proportional GP-based loading paths}

Fig. 12. Envelope of highest and lowest errors in logarithmic scale from 10 initializations of PRNNs trained on different types of loading and material layer sizes over validation set $\mathcal{V}_{(\cdot)}$. Solid lines with markers correspond to the average validation errors.



(a) Envelope of highest and lowest errors in log scale from 10 initializations of PRNNs trained on different types of loading over $\mathcal{T}_{\text{mono}}$

(b) Errors per component for best PRNNs trained on 72 curves over $\mathcal{T}_{\text{mono}}$

Fig. 13. Envelope of absolute errors from 10 PRNNs trained on different sets and evaluated on test set $\mathcal{T}_{\text{mono}}$ = {100 monotonic curves} on the left and absolute errors per component using the best performing networks with 72 curves on the right. Solid lines with markers correspond to the best performances of each training loading type for several training set sizes.

ones that have been trained with unloading. We see this as a result of two subtle advantages: (i) a loading/unloading test function much similar to the monotonic loading paths, especially the first half of the curves in $\mathcal{T}^{\text{fixed}}_{\text{unl}}$, than to the arbitrary unloading in the proportional GP-based curves and (ii) the time increment in the test curves are the same as the ones in the monotonic curves.

While these aspects help elucidate the similar performances, they do not express their significance. These networks have never seen any sort of unloading in training but are still quite capable of extrapolating to such behavior, correctly accounting for the effect of the plastic deformation. This corroborates the findings in Maia et al. (2023), where a path-dependent material model in the material layer allowed path-dependency to arise naturally. Here, we verify that the method is general and can be extended to account for other non-linearities and time dependencies. Fig. 16 shows the predictions on a curve from $\mathcal{T}^{\text{fixed}}_{\text{unl}}$ with representative errors using the best performing network. Note how close the predictions are to each other and the good agreement with respect to the micromodel solution.

Table 2
Summary of lowest absolute errors from 10 PRNNs trained on different types of curves over test set $\mathcal{T}^{\text{fixed}}_{\text{unl}}$.

Training loading type	Monotonic		Prop. GP	
Training set size	72	144	72	144
Abs. error over $\mathcal{T}^{\text{fixed}}_{\text{unl}}$ [MPa]	6.7	6.8	7.0	6.5

5.4.2. Proportional and random unloading/reloading

In this experiment, the test set $\mathcal{T}^{\text{prop. GP}}_{\text{unl}}$ is used to represent loading paths with unloading–reloading taking place at random times. These curves consist of the same type of loading used in one of the training strategies, which is similar to the situation discussed in Section 5.2. Naturally, this results in lower test errors compared to the networks trained on monotonic loading paths, as shown in Table 3. To illustrate the best performance among the 10 networks considered for each strategy, we select a curve $\mathcal{T}^{\text{prop. GP}}_{\text{unl}}$ with errors close to the average lowest absolute error (see Fig. 17(a)), along with the errors per component (see Fig. 17(b)) (see Fig. 17).

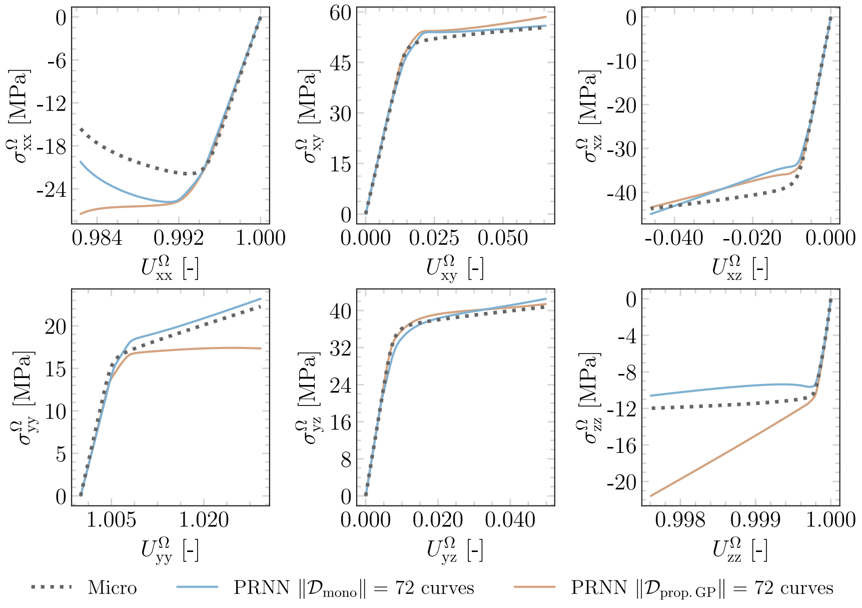


Fig. 14. Best PRNNs trained on monotonic and GP-based curves on representative curve from test set $\mathcal{T}_{\text{mono}}$.

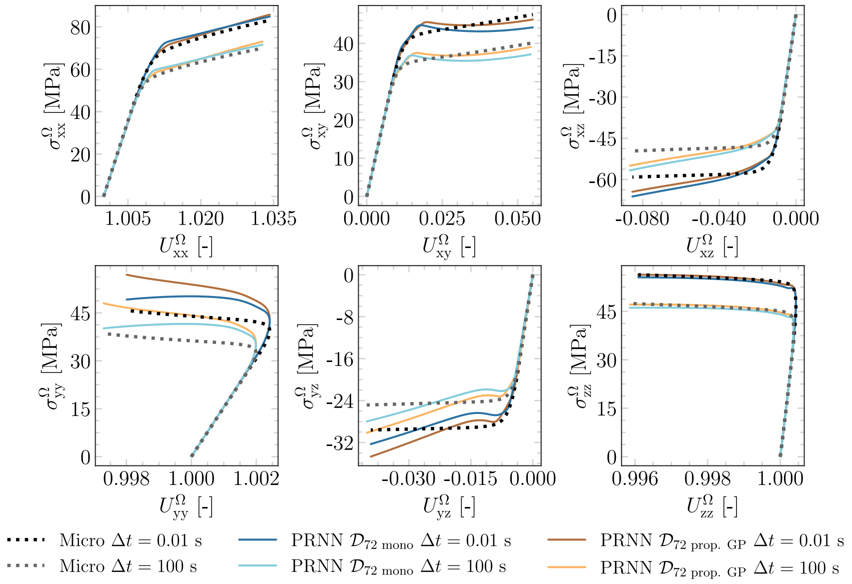


Fig. 15. Performance of PRNNs trained on monotonic and proportional GP-based curves on test sets with strain-rates 100 times slower and 100 times faster than the one used to create the monotonic training data.

Table 3
Summary of lowest absolute and relative errors from 10 PRNNs trained on different types of curves over test sets $\mathcal{T}^{\text{prop. GP}}_{\text{unl}}$ and $\mathcal{T}^{\text{non-prop. GP}}_{\text{unl}}$.

Training loading type	Monotonic		Prop. GP	
Training set size	72	144	72	144
Abs. (and rel.) error over $\mathcal{T}^{\text{prop. GP}}_{\text{unl}}$ [MPa] (%)	3.2 (8.1)	3.4 (7.8)	2.7 (5.6)	2.6 (5.1)
Abs. (and rel.) error over $\mathcal{T}^{\text{non-prop. GP}}_{\text{unl}}$ [MPa] (%)	11.5 (3.4)	12.2 (3.6)	11.0 (3.1)	10.9 (3.0)

5.4.3. Non-proportional and random unloading/reloading

For the last part of the experiments on the accuracy of the network, the test set $\mathcal{T}^{\text{non-prop. GP}}_{\text{unl}}$ is considered. Curves from this set have more complex unloading behavior and significantly higher stress levels compared to the proportional paths in $\mathcal{T}^{\text{prop. GP}}_{\text{unl}}$. This time, the slight gain in accuracy shown in Table 3 from training with the proportional non-monotonic data is examined along with the relative errors. This way, we verify that although the absolute test errors have increased, the performances remain consistent with the values seen so far (below 10 %) in terms of relative errors.

In Fig. 17(c), a representative curve from $\mathcal{T}^{\text{non-prop. GP}}_{\text{unl}}$ illustrates the best performance of both strategies over this set. The difficulty in predicting the lowest magnitude stress (in this case, $\hat{\sigma}^{\Omega}_{xz}$) becomes more evident, as well as the variety of unloading, which this time is different in each of the components. While some components go through unloading (*e.g.* $\hat{\sigma}^{\Omega}_{xx}$ and $\hat{\sigma}^{\Omega}_{xy}$), others are monotonically increasing (*e.g.*

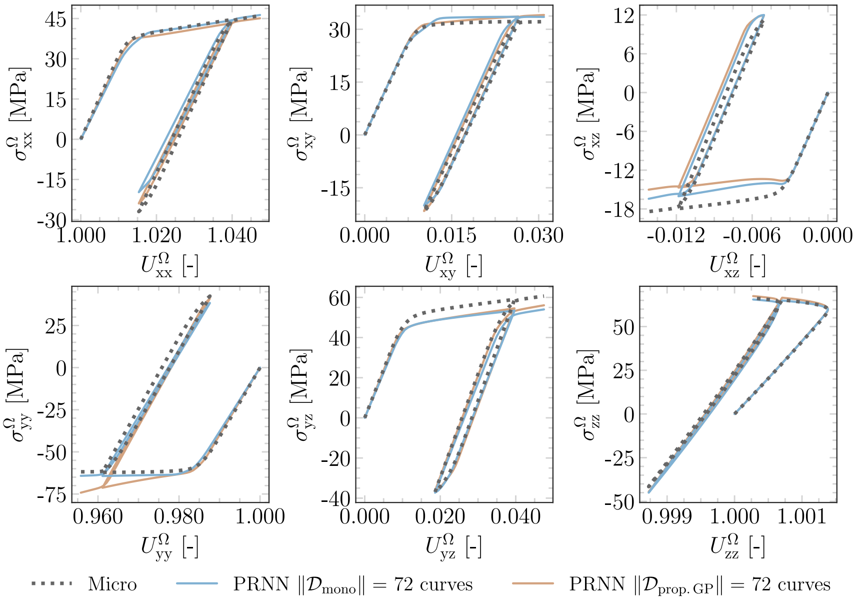


Fig. 16. Best PRNNs trained on monotonic and proportional GP-based curves on representative curve from test set $\mathcal{T}^{\mathrm{fixed}}_{\mathrm{unl}}$.

$\hat{\sigma}^{\Omega}_{\mathrm{zz}}$) and reaching high stress levels, which are naturally followed by higher absolute errors per component as seen in Fig. 17(d).

An additional curve from $\mathcal{T}^{\mathrm{non\text{-}prop.\ GP}}_{\mathrm{unl}}$ is selected and shown in Fig. 18 to highlight another aspect not yet discussed, the orthotropic behavior of the micromodel. Note that the unloading in the z-direction follows the same stress–strain path as the loading, indicating that the elastic fiber is acting as the main load-bearing component. In contrast, the shear stress in yz follows unloading in a different branch due to the development of plastic strains in the matrix.

Finally, although training with monotonic curves showed consistent and relatively accurate responses in most scenarios, the random and smooth type of loading paths explored in the previous and the current section are deemed to be more general and better representative of arbitrary functions. In both cases, training with 72 proportional GP-based curves has shown better performance and is therefore used to assess the network's capabilities in Section 7, where the network is used as a material model in several applications.

6. Runtime comparison

In this section, we perform a runtime comparison to assess the speed-up of the proposed approach in terms of the homogenized stress evaluation. For that purpose, we continue with the loading type investigated in Section 5.4.3 (type V), and select one model from the 10 initializations trained on 72 proportional GP-based curves to represent the best overall performance. Here, we use the network with the lowest error over $\mathcal{T}^{\mathrm{prop.\ GP}}_{\mathrm{unl}}$. The choice could also have been based on $\mathcal{T}^{\mathrm{non\text{-}prop.\ GP}}_{\mathrm{unl}}$, but in favor of simplicity, in a case where the experiments presented in Section 5 are not carried out, choosing from $\mathcal{T}^{\mathrm{prop.\ GP}}_{\mathrm{unl}}$ implies a simpler model selection based on a single type of loading.

In this work, all simulations, including the data generation and training procedure for the network, were executed on a single core of a Xeon E5-2630V4 processor on a cluster node with 128 GB RAM running CentOS 7. Because we are interested in the final homogenized stress $\boldsymbol{\sigma}^{\Omega}_{\mathrm{F}}$, we include in the PRNN runtime, the time spent in the transformations to bring the predicted homogenized stress back to the original frame, as illustrated in Fig. 3. For this comparison, we use as input the converged strain path and time increments from the micromodel simulations. The micromodel mesh is shown in Fig. 8 and consists of wedge elements integrated with 2 points in the thickness direction, comprising 4992 integration points and 7860 degrees of freedom.

Table 4
Computational offline costs averaged over 1100 training and validation proportional GP-based curves and over 10 PRNNs.

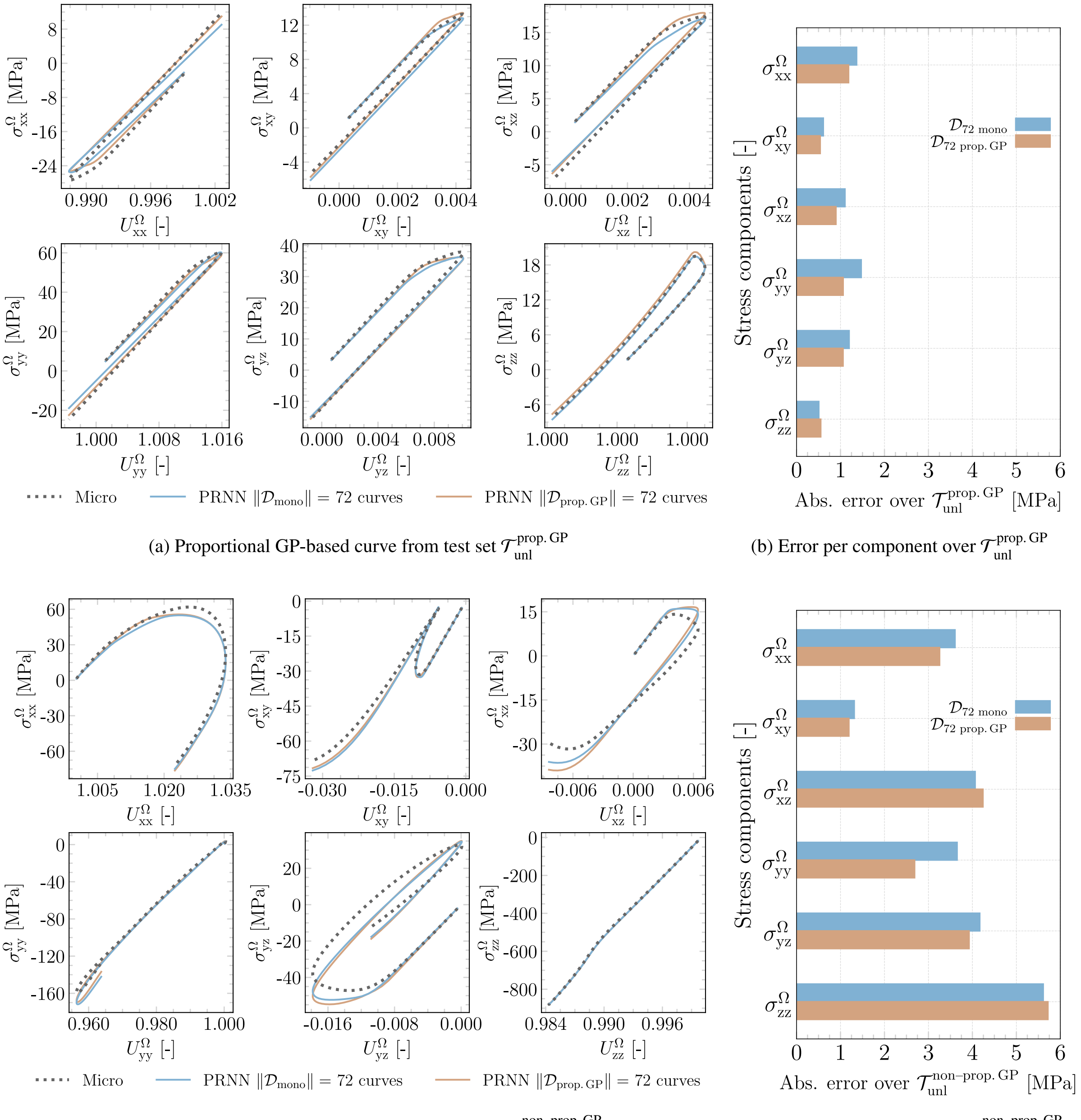
	Stress–strain curve	Training
Av. wall-clock time	3.92 min	20.34 h

Averaging over the results from 150 simulations, we break down the runtime from the full-order model in the three main parts depicted in Fig. 19. With the micromodel, roughly 30 % of the simulation is spent evaluating the constitutive models at the integration points, around 15% goes to the assembly of the global stiffness matrix and internal force vector and more than half of the total time is spent solving the system, totaling 186 s. In contrast, the network needs only 0.08 s to compute the homogenized stress state, which results in a speed-up of three orders of magnitude when compared to the full-order solution.

In terms of offline costs, we show the average times of the two main tasks involved in the training of the networks Table 4. First, the time needed to generate a full path of stretches and unrotated stresses, including the polar decomposition and rotation operations; and second, the time spent on training the PRNNs with 8 fictitious material points and 72 proportional GP-based curves itself. It is worth mentioning that, regardless of the offline costs, this section presents only an estimate of the actual speed-up. In the general case, the speed-up depends on several other aspects, such as the robustness of the tangent stiffness matrix, the complexity of the loading case, and the size of the micromodel. In multiscale settings, the gain can be higher since the cost associated with an iteration at the macroscale builds on a much higher execution time when using the micromodel compared to the network, exceeding the sum of the online evaluation and offline costs. To illustrate the potential to achieve higher speed-ups, we include an additional runtime comparison in the last application of Section 7.

7. Applications

In this section, the PRNN trained to surrogate the constitutive behavior of the micromodel in Section 5 is tested in applications in which its robustness also plays a role in obtaining the equilibrium path. By robustness, we understand the ability of the network to provide not only accurate stress predictions, as verified in Section 5, but also a tangent stiffness matrix that is stable enough for tracing an equilibrium



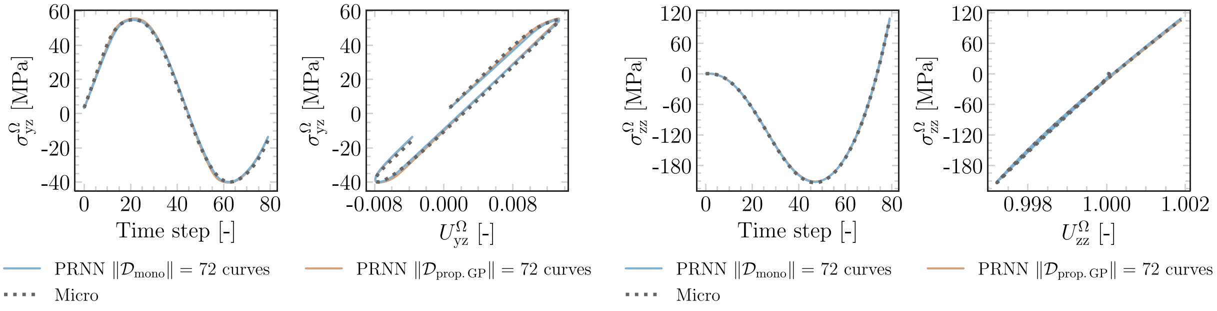
(a) Proportional GP-based curve from test set $\mathcal{T}_{\text{unl}}^{\text{prop. GP}}$

(b) Error per component over $\mathcal{T}_{\text{unl}}^{\text{prop. GP}}$

(c) Non-proportional GP-based curve from test set $\mathcal{T}_{\text{unl}}^{\text{non-prop. GP}}$

(d) Error per component over $\mathcal{T}_{\text{unl}}^{\text{non-prop. GP}}$

Fig. 17. Best PRNNs trained on monotonic and GP-based curves on representative curves from two different test sets with random unloading/reloading.



(a) Stress-time and stress-strain views of component σ_{yz}^{Ω}

(b) Stress-time and stress-strain views of component σ_{zz}^{Ω}

Fig. 18. Orthotropic behavior of selected components in loading path from $\mathcal{T}_{\text{unl}}^{\text{non-prop. GP}}$.

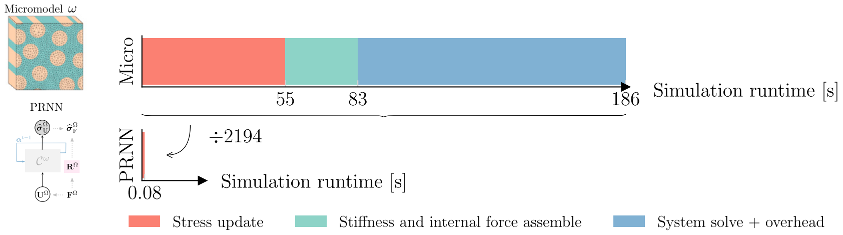


Fig. 19. Breakdown of simulation runtime using the micromodel and the PRNN averaged over 150 type V loading paths.

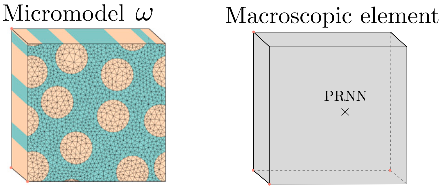


Fig. 20. Micromodel and PRNN meshes used in the applications.

path as close as possible to the one obtained by solving the micromodel. Previously, the entire strain path and time increments obtained from converged micromodel simulations were used as input. Here, the network is directly employed as the material model and therefore the stress prediction at each time step affects the following stress/strain state. In this case, lack of smoothness of the surrogate output may lead the iterative procedure to venture outside the training domain, potentially giving rise to divergence from the true solution.

For all applications in this section, we use the network with the lowest error on $\mathcal{T}_{\text{unl}}^{\text{prop. GP}}$. This time, to simulate its performance as a surrogate model to the micromodel, the network is embedded in a FE mesh that consists of a single 8 node hexahedral element with the same dimensions as the micromodel and one integration point with constitutive response given by the PRNN, as illustrated in Fig. 20. To process the deformation gradient $\mathbf{F}^{\Omega}$ into a simpler input space for the network (*i.e.* $\mathbf{U}^{\Omega}$) and obtain the stresses in their original frame of reference ($\hat{\boldsymbol{\sigma}}_{\text{F}}^{\Omega}$) using $\mathbf{R}^{\Omega}$, we use the scheme in Fig. 3. For better readability, we drop the subscript, and refer to the final stresses simply as $\hat{\boldsymbol{\sigma}}^{\Omega}$. Furthermore, for both the micromodel and the hexahedral element, in addition to the constrained displacements to avoid rigid body motion (see Fig. 1(a)), periodic boundary conditions are applied.

In the first application, we test the ability of the model to reproduce the stress relaxation phenomenon. In the second, we deal with cyclic loading and in the last application, the network is embedded in the general nonlinear framework developed by Kovačević and van der Meer (2022) to account for off-axis and constant strain-rate loading conditions. For the latter, we also include speed-up measurements to illustrate how aspects such as step size and tangent stiffness smoothness can play a role in increasing or decreasing the speed-up compared to the study in Section 6.

7.1. Relaxation

In this study, a loading function to reproduce the stress relaxation phenomenon is devised. For that, the micromodel and the PRNN are loaded until a given strain level is reached $\boldsymbol{\varepsilon}_0^{\Omega}$ at $t = t_0$, when the stress level is $\boldsymbol{\sigma}_0^{\Omega}$. After that, the strain is held constant, while a gradual stress reduction takes place. For that, we use the arc-length control introduced in Section 4 and control the stretching in the x-direction, leaving the remaining directions free to deform (see Fig. 21).

In this example, the micromodel and the homogeneous hexahedral element are loaded with $\|\Delta\mathbf{u}^{\text{c}}\| = 5 \times 10^{-6}$ mm and $\Delta t = 1$ s until $t_0 = 160$ s, when the strain level at that point is held constant until the total time of 500 s is reached and the analysis is terminated, as depicted in the lower plot of Fig. 21(a). In the upper plot, despite the mismatch in the stress before the start of the constant strain plateau, where the maximum error reaches 11.9 MPa (9%), the overall stress-time response of the micromodel is in relatively good agreement with the network's prediction, with an average error of 6 MPa (5 %). While this case represents a challenging scenario for even modern RNNs due to the long strain repetition, the expected stress decaying behavior in the prediction comes as an inherent outcome of using a material model that incorporates a spectrum of relaxation times in the material layer. To illustrate the slight difference in the stress state at the beginning and end of the constant strain plateau, we show in Fig. 21(b) two snapshots of the full-field solution.

7.2. Cyclic loading

To assess the network's performance on cyclic loading, we continue with the arc-length method and same boundary conditions as the previous application but now the uniaxial stretch at time t is described as

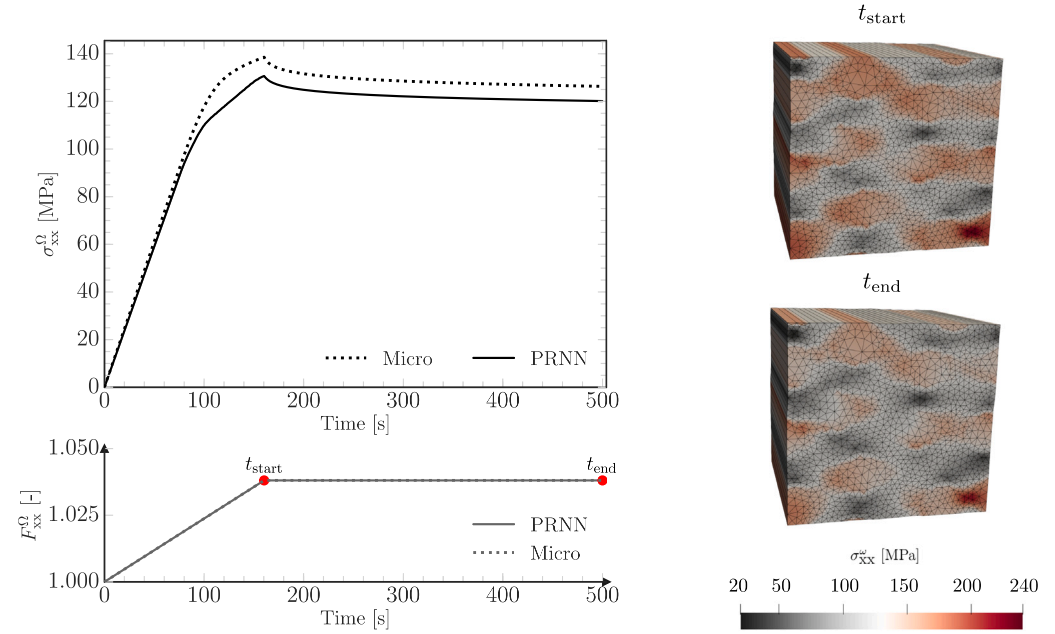
$$F_{\text{xx}}^{\Omega} = 1 + \frac{6 \times 10^{-3}}{l} \sin\left(\frac{2\pi}{1000}t\right) \tag{20}$$

where t is the time step index and $l = 0.021$ mm is the side length of the micromodel. 20 cycles are considered, each consisting of 1000 steps with $\Delta t = 1$ s. Fig. 22(a) shows the stress–strain curve for the entire loading history. The network reproduces the reverse plasticity and the hysteresis behavior in the cyclic response. Because Eq. (20) consists of a symmetric loading with constant peak and valley strains, a slow stress decay over the cycles takes place. This asymptotic relaxation process can be observed in the inset in Fig. 22(a) and is of similar nature to the one discussed in Section 7.1. Overall, good agreement is found between the PRNN and the micromodel solution. This is further assessed by unrolling the stress–strain response in time and extracting the peak and valley quantities.

First, the peak strain values from the diagonal components not controlled by the arc-length are plotted in Fig. 22(b). In this case, the strain path obtained by the network remains close to the true solution and only minor deviations are observed in the F_{yy} component. Naturally, different loading conditions lead to different levels of accuracy of the strain paths due to the indirect displacement control equation considered in this work. As for the stresses, the envelopes of maximum and minimum values for the entire loading history are shown in Fig. 23. In each, the highest absolute error is marked by double arrows, along with the corresponding relative error. Both absolute and relative errors are within the range of errors obtained in previous sections (see Fig. 22).

7.3. Constant strain-rate under off-axis loading

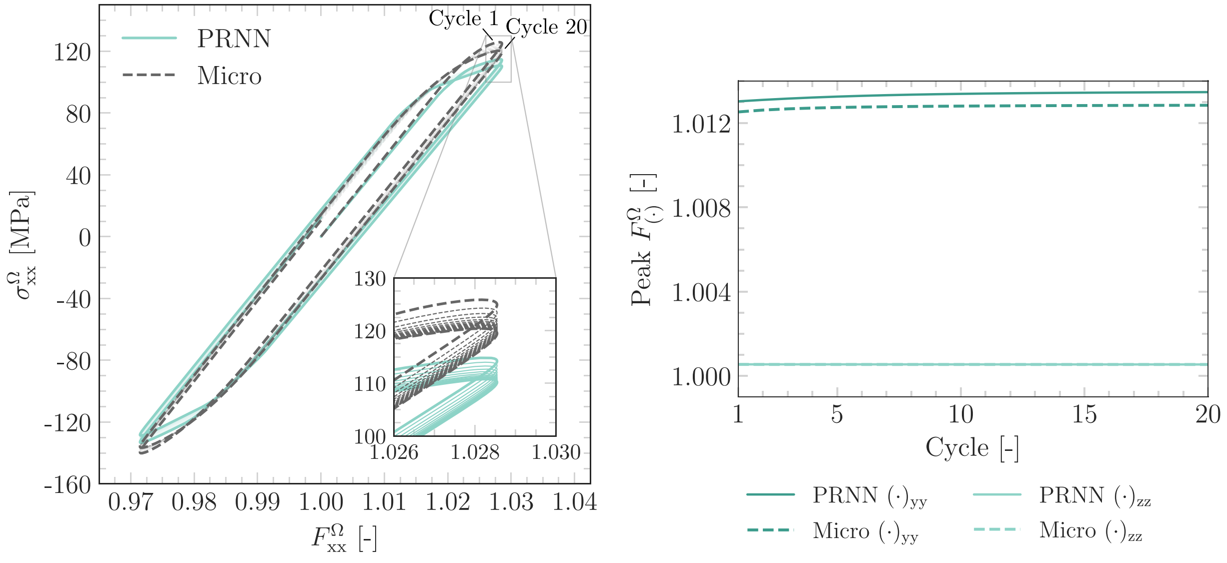
For the last application, a dedicated strain-rate based arc-length formulation is used to reproduce the response of unidirectional composites subjected to off-axis loading (Kovačević and van der Meer, 2022). In this formulation, two coordinate systems are needed: the global (x and y axes) and the local (1, 2 and 3 axes), as depicted in Fig. 24. In the



(a) Homogenized stress-time response of micromodel and PRNN on the top and homogenized strain-time on the bottom

(b) Micromodel state at the start and end of constant strain loading

Fig. 21. Homogenized stress-time response of micromodel and PRNN subjected to uniaxial stretch in x until t = 160 s, when the strain is held constant until the end of the simulation t = 500 s. On the right, the full-field of stresses of the micromodel for the start and end of the constant strain loading (in *red*).



(a) Stress-strain curve with first and last cycle highlighted

(b) Peak strains of selected components

Fig. 22. Stress–strain response of micromodel and PRNN subjected to uniaxial cyclic loading.

global coordinate system, the initial fiber orientation with respect to the y-axis is defined according to a given off-axis angle χ. The micromodel is then subjected to constant strain-rate ($\dot{\varepsilon}_{\text{yy}}$) under uniaxial stress conditions. With that, equivalent homogenized deformation and stress states need to be derived in the local frame, and the transformations between global and local coordinate systems are taken care by the custom arc-length model.

In this work, we embed the network in the *local* frame, with the time increment Δt and the homogenized deformation gradient $\overline{\mathbf{F}}$ as input and the homogenized stress $\overline{\boldsymbol{\sigma}}$ as the output. In Fig. 24(c), we show the three relevant configurations in this framework. In the simulation, due to the applied loading, the micromodel edge 0–1 tied to the local axis $\boldsymbol{e}_1$ should rotate with an angle ϕ with respect to the initial configuration (from “a” to “b”), going from the initial angle θ_0 to a new angle $\theta_1 = \theta_0 + \phi$. However, to avoid rigid-body rotation of the RVE, the controlling node 1 is fixed in the shearing direction, but the angle ϕ is implicitly taken into account through the constraint equation and the unit force vector of the arc-length model. For that reason, configuration “c”, in which $\boldsymbol{e}_1$ is always aligned to the initial fiber orientation, is used to evaluate ϕ.

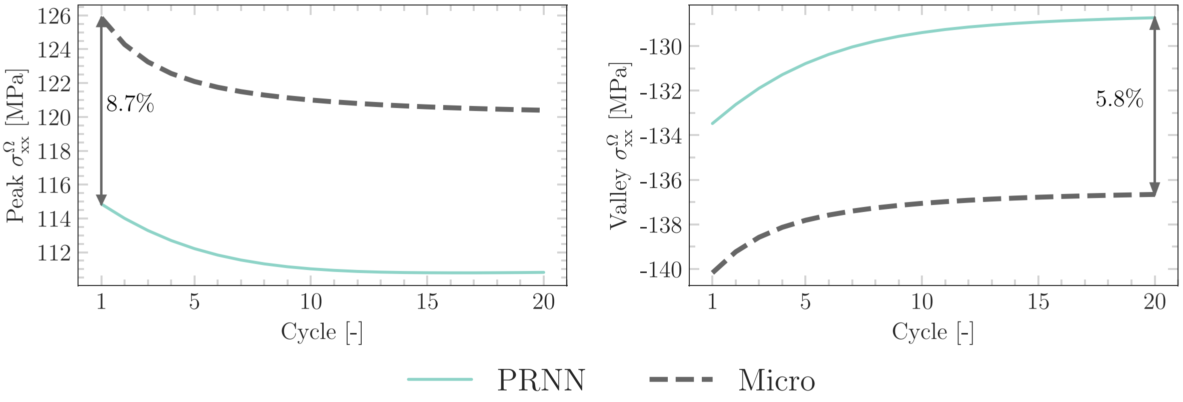
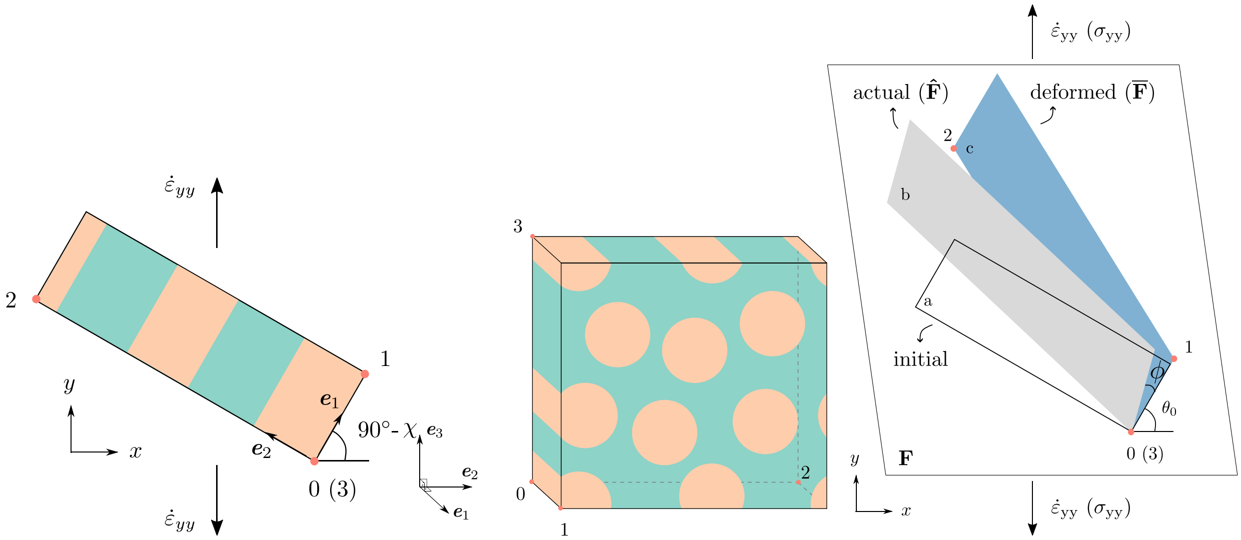


Fig. 23. Evolution of maximum and minimum stresses for all cycles with double arrows marking the relative error corresponding to the highest absolute difference between the micromodel and PRNN subjected to uniaxial cyclic loading.



(a) Global and local coordinate systems

(b) Micromodel in the local system

(c) Reorientation of micromodel due to applied loading $\dot{\varepsilon}_{yy}$

Fig. 24. Global and local coordinate systems with imposed strain-rate $\dot{\varepsilon}_{yy}$ on y direction and off-axis angle χ and reorientation of micromodel due to applied loading $\dot{\varepsilon}_{yy}$ from initial angle θ_0 to $\theta_1 = \theta_0 + \phi$ based on the deformed state (Kovačević and van der Meer, 2022).

In the local frame, the homogenized deformation gradient $\overline{\mathbf{F}}$ is given by:

$$\overline{\mathbf{F}} = \begin{bmatrix} \overline{F}_{11} & \overline{F}_{12} & 0 \\ 0 & \overline{F}_{22} & 0 \\ 0 & 0 & \overline{F}_{33} \end{bmatrix}. \tag{21}$$

To ensure the global constant strain rate condition, a special constraint equation g derived by equating the homogenized deformation gradient component in the global frame F_{yy} to the value imposed from the input is considered

$$g = \underbrace{\overline{F}_{11}\sin(\theta_0)\sin(\theta_1) + \overline{F}_{22}\cos(\theta_0)\cos(\theta_1) + \overline{F}_{12}\cos(\theta_0)\sin(\theta_1)}_{F_{yy}\text{ calculated from micromodel}} - \underbrace{\exp(\varepsilon^{t-1}_{yy} + \dot{\varepsilon}_{yy}\Delta t)}_{F_{yy}\text{ imposed from input}} = 0 \tag{22}$$

where ε^{t-1}_{yy} is the total strain in the global loading direction from the last converged time step. Another vital part of the framework is related to the update on the unit force vector applied at the controlling nodes. In this case, the geometrically nonlinear effect on the unit force vector comes not only from the change in configuration "a" to "c" but also from the change in orientation of the micromodel that ϕ introduces. Finally, to relate the stresses from both frames, one can use the load factor λ from the arc-length formulation, which is equivalent to the σ_{yy} stress component in the global frame, to transform it to the local frame:

$$\overline{\boldsymbol{\sigma}} = \sigma_{yy} \begin{bmatrix} \sin^2(\theta_1) & \cos(\theta_1)\ \sin(\theta_1) & 0 \\ \cos(\theta_1)\ \sin(\theta_1) & \cos^2(\theta_1) & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & 0 \\ \sigma_{21} & \sigma_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{23}$$

In this contribution, we particularize the framework to $\chi = 45°$ and strain-rates $\dot{\varepsilon}_{yy} = [10^{-5}\,\text{s}^{-1}, 10^{-4}\,\text{s}^{-1}, 10^{-3}\,\text{s}^{-1}]$, resulting in three simulations in total. For more details on the formulation and derivation of the expressions presented in this section, the reader is referred to Kovačević and van der Meer (2022). Starting with the global stress–strain response, results in Fig. 25 show satisfactory agreement with the full-order solution. This is yet another verification of the capability of the network to handle rate-dependency. We also inspect in Fig. 26 the evolution of separate pairs of stress and deformation gradient components in the local frame. It is emphasized, that in this simulation, none of these stress and strain components is directly controlled since there is a nonlinear relation where the evolution of the load in local frame depends on the computed deformation, except for the $\bar{\sigma}_{33}$ which

Table 5
Breakdown of simulation time and speed-up for different strain-rates $\dot{\varepsilon}_{yy}$ and $\chi = 45°$, each averaged over 10 simulations.

$\dot{\varepsilon}_{yy}$ [s^{-1}]	10^{-5}		10^{-4}		10^{-3}	
Type of analysis	Micro	PRNN	Micro	PRNN	Micro	PRNN
N_{steps} [–]	293	95	290	95	279	65
Stress evaluation [s] (%)	165 (15)	.222 (59)	160 (15)	.219 (59)	160 (14)	.168 (60)
Stiff. and int. force assemble [s] (%)	91.6 (8)	.0117 (3)	89.4 (8)	.0116 (3)	91.4 (8)	.00875 (3)
System solve + overhead [s] (%)	843 (77)	.142 (38)	843 (77)	.142 (38)	872 (78)	.105 (37)
Total simulation time [s]	1099	.375	1092	.373	1123	.282
Speed-up [–]	2929		2932		3980	

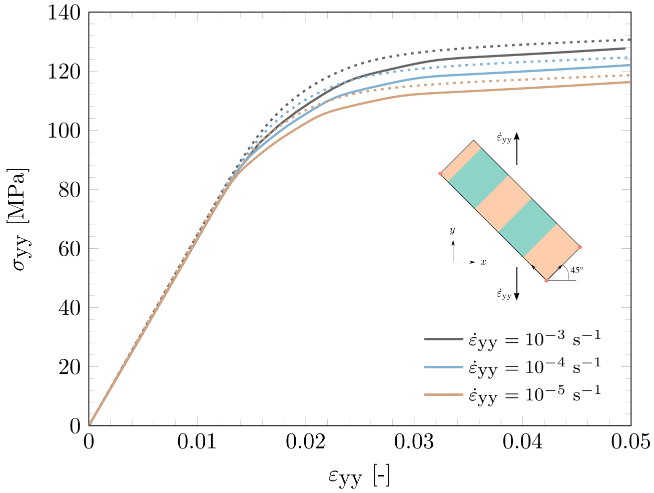


Fig. 25. Global stress–strain curve from off-axis composite with $\chi = 45°$ and different strain-rates $\dot{\varepsilon}_{yy}$. Solid and dashed lines refer to the micromodel solution and the PRNN prediction, respectively.

is kept at zero. It is observed that all deformation and stress components computed with the PRNN remain close to those coming from the micromodel.

A final assessment is made in terms of speed-up. This time, because an adaptive stepping scheme is used, the termination criterion (maximum norm) can be reached with a different number of macroscopic steps depending on the tangent stiffness matrix. For that reason, in Table 5, in addition to the breakdown of the total simulation time into the three tasks shown in Fig. 19 and the speed-up, we also show the number of steps. With less iterations, speed-ups range from 2900 to 4000, which is significantly higher than the one obtained in Section 6 (≈2200), where neither the adaptive stepping scheme nor the network's tangent and predictions are used to define the next step in tracing the equilibrium path. Other aspects, such as macroscopic mesh density and algorithmic parameters, can also influence the speed-up and the relative times of each task with respect to the total time. In this particular case with a single macroscopic element, using the PRNN as the homogenized constitutive model means that most of the time is dedicated to evaluating the network. With that, we demonstrate the potential of the proposed approach as a robust and efficient model in a practical application.

8. Concluding remarks

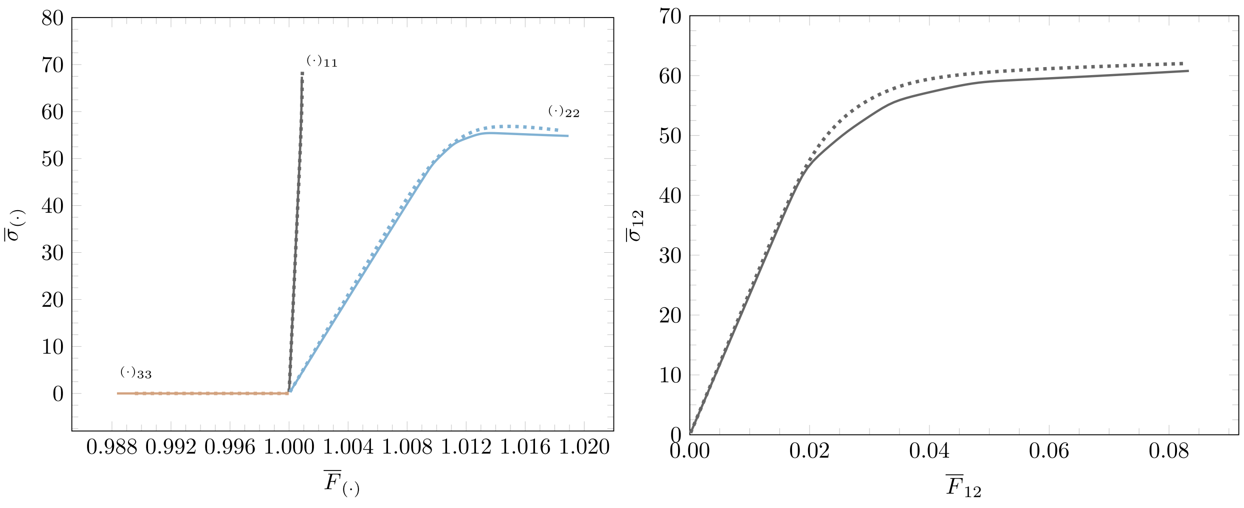
A novel Physically Recurrent Neural Network (PRNN) architecture has been developed to accelerate the microscale analysis of path and rate-dependent heterogeneous materials. The formulation follows the core idea in Maia et al. (2023), where the homogenized response of a micromodel is obtained by a network with constitutive models embedded in one of its layers. In this *material layer*, we have *fictitious material points* with the same constitutive models and properties as used in the micromodel. The values passed from encoder to the material layer are interpreted as (fictitious) local strains, which are input to the constitutive model assigned to the material points, yielding (fictitious) local stresses. These local stresses are subsequently transformed by a decoder to obtain the homogenized stress.

What distinguishes the present methodology from the state-of-the-art surrogate models, particularly the ones based on RNNs, is the strong physics-based assumptions built into the model. Here, history-dependency is a natural outcome of the embedded material models. This is because, in addition to the local stress, the material model assigned to a fictitious material point is also in charge of updating its own internal variables (if any), which are stored from one time step to another. Therefore, PRNNs naturally inherit rich memory mechanisms from the constitutive models, bypassing the need to learn these latent dynamics from data.

While the concept of having few fictitious material points representing the homogenized response of a micromodel remains at the core of the method, a new architecture is required to extend the applicability of the network to 3D problems in a finite strain framework. Among the key changes compared to Maia et al. (2023) are the use of the polar decomposition theorem and the principle of material objectivity. With the former, the deformation gradient can be uniquely decomposed into two tensors, namely stretch and rotation. The network is then used to learn the mapping between stretch and unrotated stress, from which the stress in the global coordinate frame is retrieved using the principle of material objectivity.

For the numerical examples, we considered a unidirectional composite micromodel with rate-dependent plasticity in the matrix and hyperelasticity in the fibers. Two different training strategies (monotonic vs. non-monotonic) were considered. When creating the monotonic curves, a single value of time increment was considered so that we could clearly illustrate the exceptional ability of the network to extrapolate to strain rates far from the ones seen during training. We have also tested the performance of the network on curves with increasingly complex unloading behavior. In this case, although the networks trained on monotonic data could capture unloading behavior and performed well in most of the considered scenarios, training on non-monotonic curves led to better performance overall. Comparing the number of curves of the network selected for the numerical applications with previous developments (Maia et al., 2023), now we need twice as many curves to train a PRNN that is twice as big. This linear scaling should not be expected given the exponential increase nature from the curse of dimensionality, yet we can still achieve it.

In Section 7, we shifted our focus to applications where the PRNN is directly replacing the micromodel in the solution of the equilibrium problem. In the first application, we demonstrated that the network can reproduce relaxation, which can be a difficult behavior to capture with RNNs due to the long repetition of the input (*i.e.* constant strain). In our case, since the constitutive models in the network have such behavior in their formulation, the homogenized response also reflected it robustly. In the second example, cyclic loading was considered, again showing the ability of the network to extrapolate to loading conditions and direction different than those trained for. For the last application, the special arc-length formulation proposed in Kovačević and van der Meer (2022) to account for off-axis loading and constant strain-rate conditions was employed. We particularized the framework to one off-axis angle and three different strain-rates and showed good agreement



(a) Diagonal components

(b) Off-diagonal component

Fig. 26. Stress and deformation in the local system for $\chi = 45°$ and $\dot{\epsilon}_{yy} = 10^{-4}$ s^{-1}. Solid and dashed lines refer to the micromodel solution and the PRNN prediction, respectively.

with the actual micromodel for a case where the network and its tangent are used to compute the solution of a nonlinear problem.

To assess the network's potential to accelerate micromodel simulations, we investigated two scenarios. Firstly, the network was used to predict stresses based on the converged strain paths from 150 micromodel simulations, leading to a stress evaluation 2200 times faster compared to the full-order model. Then, we assessed the speed-up on a problem in which the PRNN was directly involved in tracing the solution. In that case, the constant strain-rate application was used as a reference. It was observed that the lower number of steps needed when using the PRNN as the material model led to speed-ups even higher, between 2900 and 4000 for the different strain-rates. In summary, the proposed network provides an efficient model that can describe the rate-dependent, orthotropic response of thermoplastic composites in large deformations. Trained on data generated with a micromodel, the PRNN response remains close to that of the micromodel for a wide range of loading scenarios, including those outside the training range.

CRediT authorship contribution statement

M.A. Maia: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **I.B.C.M. Rocha:** Writing – review & editing, Supervision, Software, Project administration, Methodology, Funding acquisition, Conceptualization. **D. Kovačević:** Writing – review & editing, Software. **F.P. van der Meer:** Writing – review & editing, Supervision, Software, Project administration, Methodology, Conceptualization.

Declaration of competing interest

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

Data availability

The C++ code used in this paper is available at https://github.com/SLIMM-Lab/prnn3d A centralized repository for PRNN developments is available at https://github.com/SLIMM-Lab/pyprnn.

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.

References

Arora, R., Kakkar, P., Dey, B., Chakraborty, A., 2022. Physics-informed neural networks for modeling rate- and temperature-dependent plasticity. arXiv:2201.08363.

Bhattacharya, K., Liu, B., Stuart, A., Trautner, M., 2023. Learning Markovian homogenized models in viscoelasticity. Multiscale Model. Simul. 21 (2), 641–679. http://dx.doi.org/10.1137/22M1499200.

Bonet, J., Burton, A., 1998. A simple orthotropic, transversely isotropic hyperelastic constitutive equation for large strain computations. Comput. Methods Appl. Mech. Engrg. 162 (1), 151–164. http://dx.doi.org/10.1016/S0045-7825(97)00339-3, URL https://www.sciencedirect.com/science/article/pii/S0045782597003393.

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

Chen, Y.-C., Wheeler, L., 1993. Derivatives of the stretch and rotation tensors. J. Elasticity 32 (3), 175–182. http://dx.doi.org/10.1007/bf00131659.

Cheung, H.L., Mirkhalaf, M., 2024. A multi-fidelity data-driven model for highly accurate and computationally efficient modeling of short fiber composites. Compos. Sci. Technol. 246, 110359. http://dx.doi.org/10.1016/j.compscitech.2023.110359, URL https://www.sciencedirect.com/science/article/pii/S0266353823004530.

Eghbalian, M., Pouragha, M., Wan, R., 2023. A physics-informed deep neural network for surrogate modeling in classical elasto-plasticity. Comput. Geotech. 159, 105472. http://dx.doi.org/10.1016/j.compgeo.2023.105472.

Eghtesad, A., Fuhg, J.N., Bouklas, N., 2023. NN-EVP: A physics informed neural network-based elasto-viscoplastic framework for predictions of grain size-aware flow response under large deformations. arXiv:2307.04301.

Garanger, K., Kraus, J., Rimoli, J.J., 2023. Symmetry-enforcing neural networks with applications to constitutive modeling. arXiv:2312.13511.

Ge, W., Tagarielli, V.L., 2021. A computational framework to establish data-driven constitutive models for time- or path-dependent heterogeneous solids. Sci. Rep. 11 (1), http://dx.doi.org/10.1038/s41598-021-94957-0.

Ghane, E., Fagerström, M., Mirkhalaf, M., 2023. Recurrent neural networks and transfer learning for elasto-plasticity in woven composites. arXiv:2311.13434.

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

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

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

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

Heider, Y., Wang, K., Sun, W., 2020. SO(3)-invariance of informed-graph-based deep neural network for anisotropic elastoplastic materials. Comput. Methods Appl. Mech. Engrg. 363, 112875. http://dx.doi.org/10.1016/j.cma.2020.112875.

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

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

Kovačević, D., van der Meer, F.P., 2022. Strain-rate based arclength model for nonlinear microscale analysis of unidirectional composites under off-axis loading. Int. J. Solids Struct. 250, 111697. http://dx.doi.org/10.1016/j.ijsolstr.2022.111697.

Liu, B., Ocegueda, E., Trautner, M., Stuart, A.M., Bhattacharya, K., 2023. Learning macroscopic internal variables and history dependence from microscopic models. J. Mech. Phys. Solids 178, 105329. http://dx.doi.org/10.1016/j.jmps.2023.105329.

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

Maia, M., Rocha, I., Kerfriden, P., van der Meer, F., 2023. Physically recurrent neural networks for path-dependent heterogeneous materials: Embedding constitutive models in a data-driven surrogate. Comput. Methods Appl. Mech. Engrg. 407, 115934. http://dx.doi.org/10.1016/j.cma.2023.115934.

Masi, F., Stefanou, I., 2022. Multiscale modeling of inelastic materials with thermodynamics-based artificial neural networks (TANN). Comput. Methods Appl. Mech. Engrg. 398, 115190. http://dx.doi.org/10.1016/j.cma.2022.115190.

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

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

Pitz, E., Pochiraju, K., 2024. A neural network transformer model for composite microstructure homogenization. Eng. Appl. Artif. Intell. 134, 108622. http://dx.doi.org/10.1016/j.engappai.2024.108622.

Rocha, I.B., Kerfriden, P., van der Meer, F.P., 2020. 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 Solids 82, 103995. http://dx.doi.org/10.1016/j.euromechsol.2020.103995.

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

Rocha, I., Kerfriden, P., van der Meer, F., 2023. Machine learning of evolving physics-based material models for multiscale solid mechanics. Mech. Mater. 184, 104707. http://dx.doi.org/10.1016/j.mechmat.2023.104707.

Rocha, I., van der Meer, F., Sluys, L., 2019. Efficient micromechanical analysis of fiber-reinforced composites subjected to cyclic loading through time homogenization and reduced-order modeling. Comput. Methods Appl. Mech. Engrg. 345, 644–670. http://dx.doi.org/10.1016/j.cma.2018.11.014.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I., 2017. Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems. Vol. 30, Curran Associates, Inc..

Wen, J., Zou, Q., Wei, Y., 2021. Physics-driven machine learning model on temperature and time-dependent deformation in lithium metal and its finite element implementation. J. Mech. Phys. Solids 153, 104481. http://dx.doi.org/10.1016/j.jmps.2021.104481.

Wu, L., Noels, L., 2022. Recurrent neural networks (RNNs) with dimensionality reduction and break down in computational mechanics; application to multi-scale localization step. Comput. Methods Appl. Mech. Engrg. 390, 114476. http://dx.doi.org/10.1016/j.cma.2021.114476.

Zhang, Y., Bhattacharya, K., 2024. Iterated learning and multiscale modeling of history-dependent architectured metamaterials. arXiv:2402.12674.

Zhongbo, Y., Hien, P.L., 2024. Pre-trained transformer model as a surrogate in multiscale computational homogenization framework for elastoplastic composite materials subjected to generic loading paths. Comput. Methods Appl. Mech. Engrg. 421, 116745. http://dx.doi.org/10.1016/j.cma.2024.116745.