Machine learning of evolving physics-based material models for multiscale solid mechanics

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

DOI
10.1016/j.mechmat.2023.104707

Publication date
2023

Document Version
Final published version

Published in
Mechanics of Materials

Mechanics of Materials 184 (2023) 104707

Contents lists available at ScienceDirect

Mechanics of Materials

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

Machine learning of evolving physics-based material models for multiscale solid mechanics

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

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

ARTICLE INFO

Dataset link: https://gitlab.tudelft.nl/ibarceloscarne/evolving-material-models

Keywords:
Concurrent multiscale (FE^2) modeling
Surrogate modeling
Hybrid learning

ABSTRACT

In this work we present a hybrid physics-based and data-driven learning approach to construct surrogate models for concurrent multiscale simulations of complex material behavior. We start from robust but inflexible physics-based constitutive models and increase their expressivity by allowing a subset of their material parameters to change in time according to an evolution operator learned from data. This leads to a flexible hybrid model combining a data-driven encoder and a physics-based decoder. Apart from introducing physics-motivated bias to the resulting surrogate, the internal variables of the decoder act as a memory mechanism that allows path dependency to arise naturally. We demonstrate the capabilities of the approach by combining an FNN encoder with several plasticity decoders and training the model to reproduce the macroscopic behavior of fiber-reinforced composites. The hybrid models are able to provide reasonable predictions of unloading/reloading behavior while being trained exclusively on monotonic data. Furthermore, in contrast to traditional surrogates mapping strains to stresses, the specific architecture of the hybrid model allows for lossless dimensionality reduction and straightforward enforcement of frame invariance by using strain invariants as the feature space of the encoder.

1. Introduction

Recent advances in materials science and manufacturing techniques are paving the way for the design of materials with highly-tailored microstructures, including metamaterials (Kumar et al., 2020; Bessa et al., 2019), novel composite material systems (Gantenbein et al., 2021; Woigk et al., 2022), printed cementitious materials (Xu et al., 2022) and multifunctional living materials (Gantenbein et al., 2023). The common thread in these new developments is a shift from traditional design focused on tailoring structures to material constraints towards tailoring material microstructures to macroscopic constraints. This shift in turn requires the development of highly-detailed models of material behavior across spatial scales and a shift to virtual structural certification, as trial-and-error design becomes infeasible (Telgen et al., 2022; Khajehtourian and Kochmann, 2021; Furtado et al., 2021).

Scale bridging has been traditionally performed through a bottom-up approach: physics-based constitutive models at smaller scales are calibrated using experiments and used to perform numerical simulations (using *e.g.* the Finite Element (FE) method) on representative lower-scale domains from which higher-scale physics-based models can be calibrated (Van der Meer, 2016; Krauklis et al., 2019). However, physics-based constitutive models come with *a priori* assumptions that often fail to reproduce complex lower-scale behavior (Van der Meer, 2016). The alternative is to opt for an FE^2 (or Computational Homogenization) approach: lower-scale FE models are embedded at every Gauss point of a higher-scale model and material behavior is directly upscaled with no constitutive assumptions at the higher scale (Feyel, 1999; Kouznetsova et al., 2001; Geers et al., 2010). Yet, the computational cost associated with repeatedly solving a large number of micromodels quickly becomes a bottleneck, in particular for many-query procedures such as design exploration and optimization that require several higher-scale simulations to be performed.

Since the bottleneck of FE^2 lies in computing lower-scale models, a popular approach to reduce computational effort is to substitute the original FE micromodels with either structure-preserving reduced-order models (Ferreira et al., 2022; Goury et al., 2016; Ryckelynck, 2005; Ghavamian et al., 2017; Rocha et al., 2020b; Daniel et al., 2022; Scanff et al., 2022) or purely data-driven surrogates (Ghaboussi et al., 1991; Lefik et al., 2009; Le et al., 2015; Bessa et al., 2017; Rocha et al., 2020a; Wang et al., 2022) trained offline. More recently, Recurrent Neural Networks (RNN) have become the model of choice especially for strain path-dependent materials, with a large body of literature dedicated to their use and tuning to different applications (Ghavamian and Simone,

* Corresponding author.
E-mail address: i.rocha@tudelft.nl (I.B.C.M. Rocha).

https://doi.org/10.1016/j.mechmat.2023.104707
Received 30 January 2023; Received in revised form 13 April 2023; Accepted 26 May 2023
Available online 1 June 2023

2019; Mozaffar et al., 2019; Gorji et al., 2020; Abueidda et al., 2021; Chen, 2021; Logarzo et al., 2021; Borkowski et al., 2022). RNNs can reproduce complex long-term time dependencies in material behavior by learning latent representations of the material state, making them fast and flexible surrogates. However, these learned representations are not a priori related to actual thermodynamic internal state variables and the model is therefore poorly interpretable (see Koeppe et al., 2021 for an interesting discussion on the subject). Furthermore, training for path dependency requires sampling from a potentially infinite-dimensional space of arbitrarily-long strain paths. This means training RNNs to reproduce complex material behavior often requires an inordinate amount of data (*curse of dimensionality*) and their purely data-driven nature limits their ability to extrapolate away from paths seen during training.

In order to address these drawbacks, a growing number of recent works are shifting focus to models with a fusion of data-driven and physics-based components. Inspired by physics-informed neural networks (Raissi et al., 2019), the authors in Masi et al. (2021) opt for data-driven models with physics-inspired bias by enforcing thermodynamic principles in a weak sense through an augmented loss function. In a similar vein, the model in Linka et al. (2021) learns hyperelasticity by linking together several carefully crafted neural nets to represent quantities with clear physical meaning, improving the interpretability of the resulting model. In Vlassis and Sun (2021) the authors extend a similar hyperelastic surrogate with a network that learns plastic flow direction and the evolution of a yield surface parametrized by a level set function, resulting in a hyperelastic–plastic model with superior extrapolation capabilities. A common thread in the aforementioned approaches, however, is that their learning architectures are heavily dependent on the type of model being learned (*e.g.* hyperelasticity, plasticity), making extensions to other models a convoluted task. In contrast, the authors in Liu et al. (2019), Liu (2021) propose a surrogate for heterogeneous micromodels constructed by directly employing unmodified versions of the constitutive models used for the micro constituents and using a customized network architecture to infer a homogenization operator from data that combines their responses. Nevertheless, the method employs a highly-specialized iterative online prediction routine requiring extra implementation effort and with increased computational overhead when compared to that of traditional surrogates mapping strains to stresses. Finally, in Wang et al. (2019), Flaschel et al. (2021, 2022) a dictionary of candidate physics-based models is assumed and the role of machine learning shifts instead to that of performing model selection and/or design of experiments.

In this work we explore an alternative approach for constructing hybrid surrogate models for path-dependent multiscale simulations. We start from the premise that existing physics-based models — *e.g.* the ones used to describe microscale constituents — are not flexible enough to reproduce macroscale behavior but nonetheless encapsulate crucial physical features such as frame invariance and loading/unloading conditions. It is our aim to avoid learning these features directly from data, as that would require either an excessively large dataset or a highly-specialized learning architecture. We therefore opt for keeping the constitutive model as intact as possible and instead increasing flexibility by allowing some (or all) of its material parameters to evolve in time. The resulting model can be seen in Fig. 1: a data-driven encoder[1] that learns the evolution of a set of material properties is linked to a physics-based material model decoder that maps strains to stresses. In contrast to other strategies in literature, we keep the architecture as general as possible: a general feature extractor parses macroscopic strains into input features for the encoder — which can be as simple as the strains themselves or other derived quantities (*e.g.* strain invariants) — and any type of constitutive model can in principle act as decoder (*e.g.* hyperelasticity, plasticity, damage). By relegating stress computations to the decoder, we effectively introduce physics-based bias to the model.[2] Furthermore, by letting the material model handle the evolution of its own internal variables, the model benefits from a recurrent component with interpretable memory structure that allows path dependency to arise naturally. The strategy we explore here is related to the one we propose in Maia et al. (2023), but in that work we let an encoder learn local strain distributions for several virtual material points with fixed properties. We see the two approaches as being complementary, and therefore with potential for being used in combination to form a flexible range of hybrid surrogates.

The remainder of the work is organized as follows. Section 2 contains a primer on concurrent multiscale (FE^2) modeling and discusses the difficulties of training purely data-driven surrogates. In Section 3, we particularize the model of Fig. 1 to the case of a feedforward neural network encoder and discuss aspects related to offline training and online numerical stabilization. In Section 4 we assess the performance of the hybrid model in reproducing the behavior of fiber-reinforced composites using different encoder input features and decoder models. Finally, some concluding remarks and future research directions are discussed in Section 5.

2. Concurrent multiscale (FE^2) modeling

In this section we present a short discussion on FE^2 modeling. The goal is not to be comprehensive — the interested reader is referred to Kouznetsova et al. (2001), Geers et al. (2010) for detailed discussions on the subject — but rather to expose the computational bottleneck associated with the method and pinpoint where surrogate models can be used to alleviate the issue. We then demonstrate how a Recurrent Neural Network (RNN) can be used as surrogate model and showcase some of the difficulties associated with their training and their extrapolation capabilities.

2.1. Scale separation and coupling

In FE^2 we assume the problem being solved can be split into a homogeneous macroscopic domain Ω and a heterogeneous microscopic domain $\omega \ll \Omega$ where small-scale geometric features are resolved. Here we opt for a first-order homogenization approach assuming the displacements on both scales can be related by:

$$\mathbf{u}^{\omega} = \boldsymbol{\varepsilon}^{\Omega}\mathbf{x}^{\omega} + \tilde{\mathbf{u}} \tag{1}$$

where microscopic displacements $\mathbf{u}^{\omega}$ are split into a linear contribution proportional to the macroscopic strains $\boldsymbol{\varepsilon}^{\Omega}$ and a fluctuation term $\tilde{\mathbf{u}}$ that accounts for microscopic heterogeneities.

Since $\boldsymbol{\varepsilon}^{\Omega}$ varies throughout the macroscopic domain, a micromodel for ω is embedded at each Gauss point in Ω and a microscopic boundary-value equilibrium problem assuming small displacements and strains is solved:

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

microscopic stress $\boldsymbol{\sigma}^{\omega}$ is related to microscopic strain $\boldsymbol{\varepsilon}^{\omega}$ with traditional physics-based constitutive models for each phase in the heterogeneous domain. In the general case where the material models feature internal

[1] We adopt the terms *encoder* and *decoder* in order to clearly split the model in two distinct parts which we will treat with different learning approaches. As in conventional machine learning models, our encoder–decoder architecture transforms to and from a clearly defined latent space. In the present discussion it does not imply a dimensionality bottleneck.

[2] In purely data-driven surrogates, we accept some bias in exchange for reduced variance — *e.g.* by employing regularization or adopting prior distributions for model parameters (Bishop, 2006) — in order to counter overfitting and improve generalization. But in that case the bias is merely a way to reduce complexity, with no physical interpretation and no *a priori* impact on the extrapolation capabilities of the model.

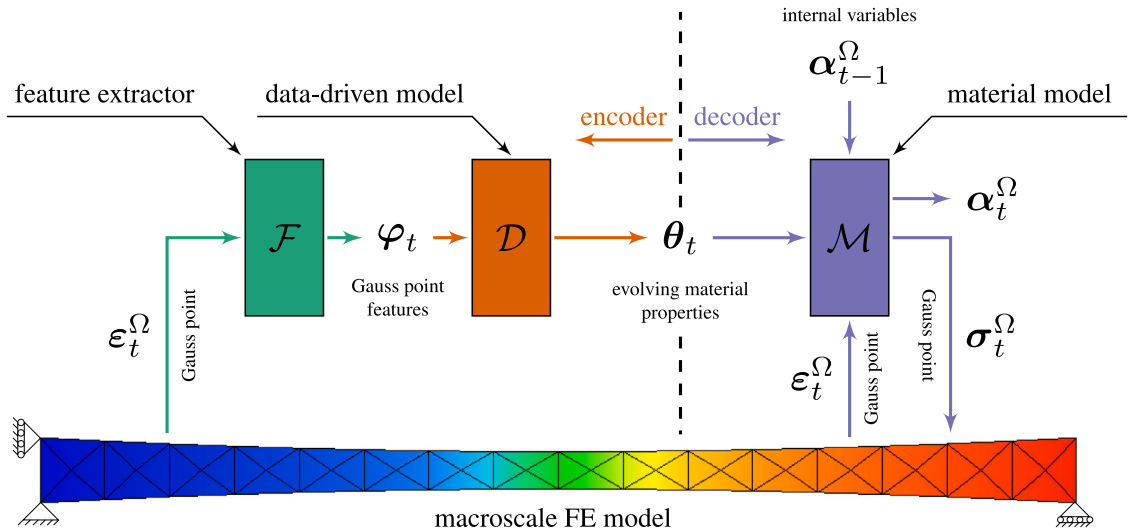


Fig. 1. The hybrid surrogate combining a data-driven encoder for material parameters and a physics-based material model decoder.

variables $\boldsymbol{\alpha}$, we can write the constitutive update for the microscale domain as:

$$\mathcal{M}^{\omega} \begin{cases} \boldsymbol{\alpha}_t^{\omega} = \mathcal{A}\left(\boldsymbol{\varepsilon}_t^{\omega}, \boldsymbol{\alpha}_{t-1}^{\omega}, \boldsymbol{\theta}^{\omega}\right) \\ \boldsymbol{\sigma}_t^{\omega} = \mathcal{S}\left(\boldsymbol{\varepsilon}_t^{\omega}, \boldsymbol{\alpha}_t^{\omega}, \boldsymbol{\theta}^{\omega}\right) \end{cases} \tag{3}$$

where $\boldsymbol{\theta}^{\omega}$ are the material parameters of the microscopic constituents, the operators $\mathcal{A}$ and $\mathcal{S}$ can be split into an arbitrary number of blocks with different models (*e.g.* elasticity, elastoplasticity, damage) for the different material phases, and $\boldsymbol{\alpha}^{\omega}$ is a concatenation of the internal variables of every microscopic Gauss point and therefore fully describes the path-dependent state of the microscopic problem.

In order to determine the strains $\boldsymbol{\varepsilon}^{\Omega}$ that serve as boundary conditions for the micromodels, a macroscopic small-strain equilibrium problem is solved:

$$\nabla \cdot \boldsymbol{\sigma}^{\Omega} = \mathbf{0} \qquad \boldsymbol{\varepsilon}^{\Omega} = \frac{1}{2}\left(\nabla \mathbf{u}^{\Omega} + \left(\nabla \mathbf{u}^{\Omega}\right)^{\mathrm{T}}\right) \tag{4}$$

but this time no constitutive assumptions are adopted. Macroscale stresses are instead directly homogenized from the microscopic response:

$$\boldsymbol{\sigma}^{\Omega} = \frac{1}{|\omega|}\int_{\omega} \boldsymbol{\sigma}^{\omega} \mathrm{d}\omega \tag{5}$$

which couples the macroscopic strain $\boldsymbol{\varepsilon}^{\Omega}$ with the microscopic solution. Since Eq. (1) also couples the solutions in the opposite direction, a bidirectional coupling is formed which requires the two-scale equilibrium problem to be solved iteratively.

2.2. Data-driven surrogate modeling

The coupled problem of Section 2.1 is extremely computationally demanding. The lower-scale domain ω usually features complicated geometric features and must therefore be modeled with dense FE meshes in order to ensure accuracy. Worse yet, an independent microscopic problem must be solved at every integration point in Ω for every iteration of every time step of the simulation. This nested nature quickly forms a computational bottleneck.

Since the bulk of the computational effort lies in solving the micromodels, a popular approach to make multiscale analysis viable for practical applications is to substitute the microscopic FE models by data-driven surrogates. The idea is to perform a number of micromodel simulations under representative boundary conditions and use the resulting stress–strain pairs to train a machine learning model to be deployed when performing the actual two-scale simulations of interest. Naturally, the approach tacitly assumes that the number of offline micromodel computations required to train the model is much smaller than the number of times the microscopic behavior will be computed online. In the following, we use a simple example to demonstrate a number of difficulties associated with training such a model to reproduce path-dependent material behavior.

2.3. Example: A one-dimensional RNN surrogate

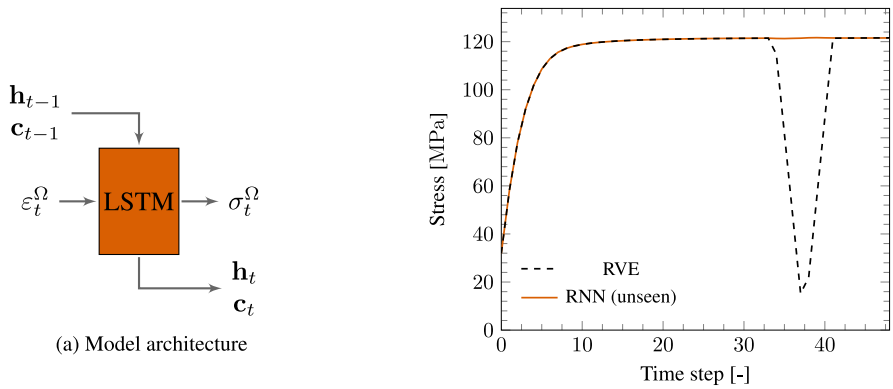
For this short demonstration, we train a Long Short-term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) to reproduce one-dimensional (single stress/strain component) elastoplasticity. The goal is not to provide a comprehensive investigation on constitutive modeling with RNNs, but rather to clearly identify parallels with our proposed approach and, more importantly, highlight how the two approaches differ from each other.

The architecture of the model is shown in Fig. 2(a) and is implemented in PyTorch (Paszke et al., 2019). In order to minimize the risk of overfitting, a pragmatic model selection procedure is performed by first training the model with several non-monotonic strain paths and gradually increasing cell size until reasonable accuracy is obtained. This leads to a parsimonious model with a single LSTM cell with 5 latent units.

At this point it is interesting to draw a parallel between the network and the micromodel whose behavior is being reproduced: the concatenation of the hidden state $\mathbf{h}$ and cell state $\mathbf{c}$ of the LSTM cell can be seen as a lower-dimensional surrogate for the set of microscopic internal variables $\boldsymbol{\alpha}^{\omega}$ of Eq. (3). However, in contrast to the variables in $\boldsymbol{\alpha}$, the latent variables $\mathbf{h}$ and $\mathbf{c}$ have no physical interpretation and evolve purely according to heuristic memory mechanisms that mimic patterns inferred during training.

First, we train the LSTM using only monotonic data. Since only one strain component is being modeled, this initial dataset is composed simply of one strain path in tension and one in compression. The trained model is then used to predict a tension path with one unloading–reloading cycle. Having never seen unloading during training, the network reverses course and unloads on top of its loading path (Fig. 2(b)). This result is hardly surprising, but sheds light on the potentially deceiving nature of the training procedure: even though we are only concerned with a single strain component, predictions actually take place in an augmented space that describes strain paths in time which can be arbitrarily high-dimensional (as paths can be arbitrarily long).

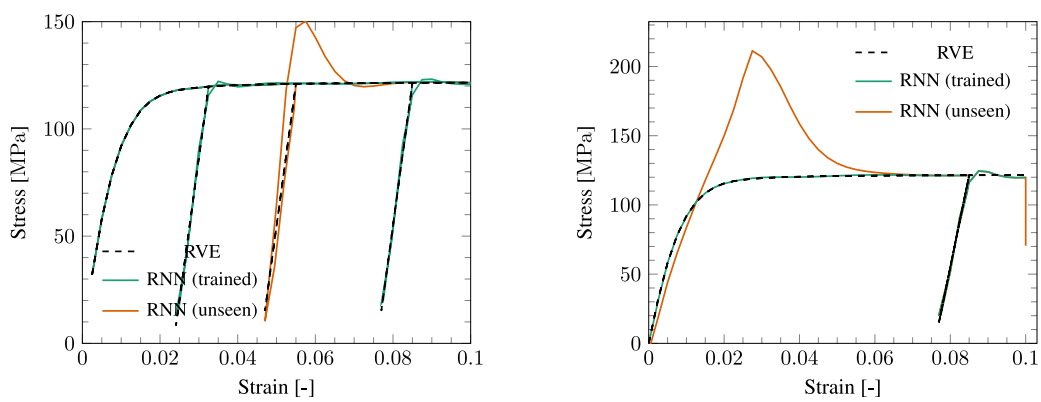
We can further demonstrate this manifestation of the *curse of dimensionality* with the two additional examples of Fig. 3. In Fig. 3(a)



(a) Model architecture

(b) Failure to predict unseen unloading

Fig. 2. An LSTM recurrent neural network as surrogate for 1D path-dependent material behavior trained with only monotonic data.



(a) Unloading at new strain level

(b) Stopping for 5 steps at $\varepsilon = 0$ and $\varepsilon = 0.1$

Fig. 3. 1D LSTM surrogate trained with unloading/reloading and used to predict unseen unloading paths.

we train the network with two unloading paths and it fails to predict a third one at an intermediate strain level. Here it can be deceiving to assume the third path can be interpolated from the other two: in the 48-dimensional space of strain paths (we use paths with 48 time steps each) the network is actually operating far away from training data. In Fig. 3(b) the network tries to reproduce a path seen during training but we first let the material rest at zero strain for five time steps before loading starts and for another five time steps at the end of the path. With purely data-driven latent dynamics, the initial rest disturbs the memory structure of the network and causes large deviations for a large portion of the path. For the rest at the end of the path, we see that the surrogate fails to predict the characteristic that the stress does not change upon constant deformation.

Training data-driven models to accurately reproduce path dependency is therefore not straightforward: their latent representations of material state are not interpretable and even phenomena as trivial as resting at zero strain must be learned from data. At the core of successful applications of RNNs to this task are either extensive datasets obtained with carefully crafted sampling strategies (Wu et al., 2020; Logarzo et al., 2021) or highly tailored datasets for specific macroscopic problems (Ghavamian and Simone, 2019). Alternatively, active learning frameworks may be used to skip offline training altogether (Knap et al., 2008; Rocha et al., 2021), but at the cost of producing slower surrogates.

3. A hybrid surrogate model

In this work we attempt to avoid the curse of dimensionality by relegating to a physics-based material model some of the tasks the RNN of Section 2.3 has to explicitly learn from data. In this section, we further formalize the hybrid approach of Fig. 1 by looking at the roles of each model component and their dependencies in time. We then particularize the model for the case of a feedforward neural network (FNN) encoder and discuss feature selection and numerical stabilization strategies.

3.1. Evolving material parameters

Physics-based material models are traditionally formulated with a fixed set of parameters $\boldsymbol{\theta}$ either directly computed from a specific set of (numerical) experiments or indirectly from stress–strain measurements in a Maximum Likelihood Estimation (MLE) approach.[3] Here we start from the premise that letting (part of) $\boldsymbol{\theta}$ evolve in time increases flexibility and allows the model to capture more complex material behavior. Conversely, keeping the remainder of the model intact improves interpretability and provides physics-based bias to the data-driven model tasked to learn this evolution.

In Fig. 4, the hybrid model of Fig. 1 is unrolled in time for a number of consecutive time steps and represented as a graph showing the dependencies between variables. Filled and hollow nodes represent observed and latent variables, respectively, and are color coded to represent the different model components in Fig. 1. Similar to the

[3] The parameters $\boldsymbol{\theta}$ can also be estimated through Bayesian inference and would therefore be described by a multivariate probability density instead of a fixed set of values. Regardless, that density would still be stationary in time.

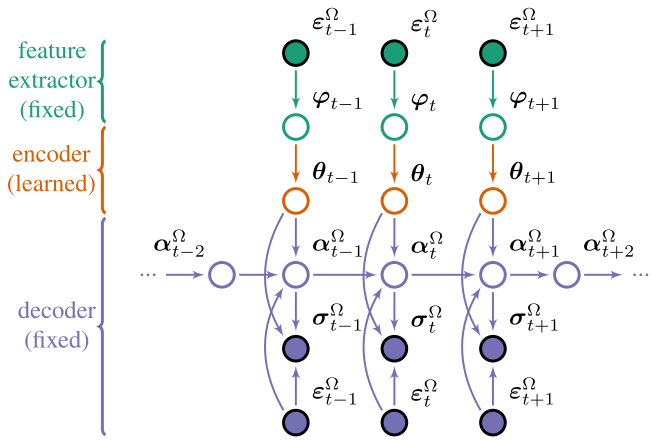


Fig. 4. Graph representation of the hybrid model architecture combining a data-driven encoder and a physics-based decoder. Filled circles represent observable variables and hollow circles represent latent variables.

microscale models of Eq. (3), we assume the constitutive behavior at the macroscale is given by a physics-based material model:

$$\mathcal{M}^{\Omega} \begin{cases} \boldsymbol{\alpha}_t^{\Omega} = \mathcal{A}\left(\boldsymbol{\varepsilon}_t^{\Omega}, \boldsymbol{\alpha}_{t-1}^{\Omega}, \boldsymbol{\theta}_t^{\Omega}\right) \\ \boldsymbol{\sigma}_t^{\Omega} = \mathcal{S}\left(\boldsymbol{\varepsilon}_t^{\Omega}, \boldsymbol{\alpha}_t^{\Omega}, \boldsymbol{\theta}_t^{\Omega}\right) \end{cases} \tag{6}$$

but now with time-dependent parameters $\boldsymbol{\theta}_t$. Note that the model response at time t depends on the material state at time $t-1$ through a set of internal variables $\boldsymbol{\alpha}_{t-1}^{\Omega}$ (Fig. 4). This gives the model a recurrent nature not unlike that of the RNN of Fig. 2(a) with its state variables $\mathbf{c}$ and $\mathbf{h}$. The advantage here is that $\boldsymbol{\alpha}$ has clear physical interpretation (plastic strains, damage variables, etc.) and its evolution is handled by the fixed operator $\mathcal{A}$ composed of clearly interpretable algorithmic steps grounded in physics and/or classical material phenomenology (*e.g.* a return mapping algorithm).

On the encoder side, we let the material properties $\boldsymbol{\theta}$ evolve according to an evolution operator $\mathcal{D}$ whose shape is learned from data:

$$\boldsymbol{\theta}_t = \mathcal{D}\left(\boldsymbol{\varphi}_t\right) \tag{7}$$

as a function of a set of input features $\boldsymbol{\varphi}$ that are themselves obtained from the macroscopic strains through a feature extractor $\mathcal{F}$:

$$\boldsymbol{\varphi}_t = \mathcal{F}\left(\boldsymbol{\varepsilon}_t^{\Omega}\right) \tag{8}$$

where $\boldsymbol{\varphi}_t$ could be simply the strains themselves or other quantities derived from it. This further split of the encoder architecture into $\mathcal{F}$ and $\mathcal{D}$ helps clearly delimitate where the purely data-driven part of the hybrid model begins. More importantly, note that $\boldsymbol{\theta}_t$ depends only on the current features $\boldsymbol{\varphi}_t$ and we therefore assume $\mathcal{D}$ is not recurrent (Fig. 4). This choice effectively limits the flexibility of $\mathcal{D}$ and makes the hybrid surrogate fully rely on the more robust model $\mathcal{M}^{\Omega}$ to explain path-dependent phenomena, helping counter the curse of dimensionality associated with sampling strain paths. For instance, it opens up the possibility to train the surrogate exclusively with monotonic data, as we will demonstrate in the examples of Section 4.

In the following sections, we particularize the model for the case of $\mathcal{D}$ being a fully-connected neural network and for specific choices of $\mathcal{F}$ and $\mathcal{M}$. Nevertheless, the general architecture of Figs. 1 and 4 is meant to be as flexible as possible:

- The nature and dimensionality of $\boldsymbol{\varphi}$ is not tied to that of $\boldsymbol{\varepsilon}^{\Omega}$ since strains are also given directly to $\mathcal{M}^{\Omega}$;
- Other machine learning models for regression can also be used as $\mathcal{D}$, and it could in principle be split into different models handling the evolution of different subsets of $\boldsymbol{\theta}$. Any number of model parameters may also be left out of $\boldsymbol{\theta}$ and either fixed as constants or optimized to constant values during training;
- No assumption is made on the form of $\mathcal{M}^{\Omega}$ or the nature or dimensionality of $\boldsymbol{\alpha}^{\Omega}$. Instead of a single model, it could also for instance be a mixture of physics-based models combined with analytical homogenization techniques.

3.2. Feature extractors

A pragmatic choice for $\mathcal{F}$ is to simply assume $\boldsymbol{\varphi}$ is the macroscopic strain vector $\boldsymbol{\varepsilon}^{\Omega}$ itself. It is also a familiar one, as we can then relate the resulting model to conventional surrogates mapping strains to stresses. However, since macroscopic strains are also directly passed on to the decoder, the architecture gives us the freedom to experiment with different input features.

Fig. 5 shows the two model architectures we explore in this work. For the two variants in Fig. 5(a) we either use $\boldsymbol{\varepsilon}^{\Omega}$ itself or a set of small-strain invariants of the macroscopic strain tensor of increasing dimensionality:

$$\mathbf{I}_{\varepsilon}^{\Omega} = \left[I_1^{\varepsilon}\right] \quad \text{or} \quad \mathbf{I}_{\varepsilon}^{\Omega} = \left[I_1^{\varepsilon} \quad I_2^{\varepsilon}\right] \tag{9}$$

where the variants are given by the well-known expressions:

$$I_1^{\varepsilon} = \operatorname{tr}(\boldsymbol{\varepsilon}), \quad I_2^{\varepsilon} = \frac{1}{2}\left(\operatorname{tr}(\boldsymbol{\varepsilon})^2 - \operatorname{tr}\left(\boldsymbol{\varepsilon}^2\right)\right) \tag{10}$$

Additionally, since the current study focus on elastoplasticity, it is also interesting to explore input feature spaces including invariants from the deviatoric strain tensor:

$$\mathbf{I}_{\varepsilon}^{\Omega} = \left[J_2^{\varepsilon}\right] \quad \text{or} \quad \mathbf{I}_{\varepsilon}^{\Omega} = \left[I_1^{\varepsilon} \quad J_2^{\varepsilon}\right] \tag{11}$$

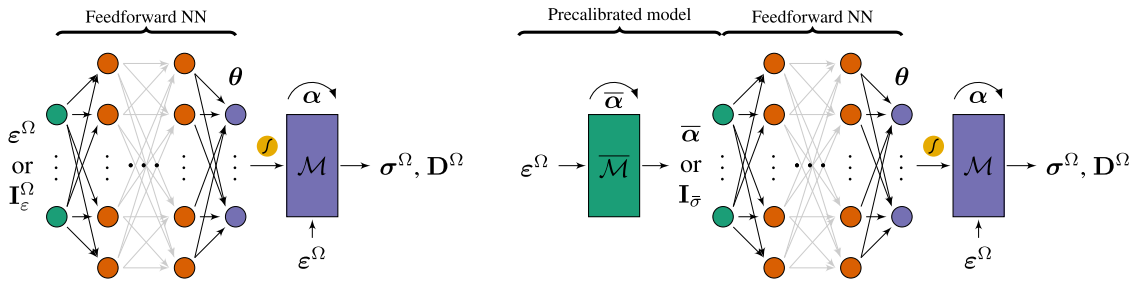
where:

$$J_2^{\varepsilon} = \frac{1}{3}\left(I_1^{\varepsilon}\right)^2 - I_2^{\varepsilon} \tag{12}$$

By using features based on invariants, the resulting surrogate can be made to naturally inherit frame invariance as long as the behavior of the physics-based decoder is also frame invariant. This stands in contrast with traditional black-box surrogates mapping strains to stresses. Furthermore, opting for invariant-based features can be seen as a physics-based dimensionality reduction operation that can potentially reduce the amount of data needed to train the hybrid model.

We also investigate the possibility of extracting features from the outputs of a precalibrated material model $\overline{\mathcal{M}}$ subjected to the same strain path seen at the macroscale (Fig. 5(b)). This $\overline{\mathcal{M}}$ is a physics-based material model, similar to $\mathcal{M}$, with the difference that its properties are not trained to reproduce training data but are fixed *a priori* (*i.e.* it is *precalibrated*). Note that this specific architecture introduces an additional recurrent component to the model through the set $\overline{\boldsymbol{\alpha}}$ of internal variables of $\overline{\mathcal{M}}$. From a machine learning perspective, the role of $\overline{\mathcal{M}}$ would be analogous to that of a temporal convolution operator or an RNN cell appended to the encoder. The key difference, however, is that $\overline{\mathcal{M}}$ is fixed *a priori* and therefore should not require extra sampling effort with respect to the more straightforward extractor in Fig. 5(a).

Naturally, different choices for $\overline{\mathcal{M}}$ yield models with distinct learning capabilities, and we therefore assume $\overline{\mathcal{M}}$ encapsulates relevant information about not only the current values of $\boldsymbol{\varepsilon}^{\Omega}$ but also of its history. In the present scenario where the data is coming from micromodel computations, we opt for the intuitive choice of having $\overline{\mathcal{M}}$ be one of the known constitutive models used to describe the microscopic material phases. We can therefore conceptually see $\overline{\mathcal{M}}$ as an imaginary representative material point at the microscale that is always subjected to the average micromodel strain. We then use either a subset of its internal variables $\overline{\boldsymbol{\alpha}}$ or a set of invariants $\mathbf{I}_{\overline{\sigma}}$ of its stress outputs as features.



(a) Macroscopic strains or invariants directly used as features (b) Features extracted from an informative microscopic constitutive model

Fig. 5. The two types of FNN-based model architectures explored in this work, with different feature extraction steps.

3.3. Neural network encoder

For simplicity, we opt for modeling the evolution of $\boldsymbol{\theta}$ using classical feedforward neural networks with fully-connected layers. As both architectures in Fig. 5 ultimately compute macroscopic stresses given macroscopic strains, we can use supervised learning to train the model with a straightforward Maximum Likelihood approach. Gathering the complete set of network weights in a vector $\mathbf{w}$ and seeing the complete surrogate as a monolithic model that computes an approximation $\hat{\boldsymbol{\sigma}}$ for stresses, we adopt the following observation model for the snapshot stresses $\boldsymbol{\sigma}$:

$$\boldsymbol{\sigma} = \hat{\boldsymbol{\sigma}}\left(\boldsymbol{\varepsilon}, \mathbf{w}\right) + \xi, \quad \xi \sim \mathcal{N}\left(\xi | \mathbf{0}, \beta^{-1}\mathbf{I}\right) \tag{13}$$

where the superscript Ω is dropped for convenience, $\mathbf{I}$ is an identity matrix, and ξ is an additive Gaussian noise.[4] Under the assumption of a squared loss, maximizing the likelihood of a training dataset with N observations amounts to minimizing the loss function (Bishop, 2006):

$$L = \frac{1}{2}\sum_{n=1}^{N}\left\|\boldsymbol{\sigma}_n - \hat{\boldsymbol{\sigma}}\left(\boldsymbol{\varepsilon}_n, \mathbf{w}\right)\right\|^2 \tag{14}$$

with the variance of the noise that explains data misfit being simply $\beta = N/2L$. The resulting loss function is the same one used for conventional data-driven surrogates and is therefore straightforward to implement.

Nevertheless, it is worth noting that since we cannot directly observe $\boldsymbol{\theta}$, computing the gradients of L with respect to $\mathbf{w}$ involves backpropagating derivatives through the decoder $\mathcal{M}$. Furthermore, since $\mathbf{w}$ affects the evolution of the internal variables $\boldsymbol{\alpha}$, backpropagation in time becomes necessary. Starting from Eq. (14) and walking back through the graph of Fig. 4, the gradient of the loss at time step t of a given strain path is given by:

$$\begin{aligned}\frac{\partial L_t}{\partial \mathbf{w}} = \frac{\partial L}{\partial \hat{\boldsymbol{\sigma}}_t}\Bigg\{&\frac{\partial \hat{\boldsymbol{\sigma}}_t}{\partial \boldsymbol{\theta}_t}\frac{\partial \boldsymbol{\theta}_t}{\partial \mathbf{w}} + \frac{\partial \hat{\boldsymbol{\sigma}}_t}{\partial \boldsymbol{\alpha}_t}\frac{\partial \boldsymbol{\alpha}_t}{\partial \boldsymbol{\theta}_t}\frac{\partial \boldsymbol{\theta}_t}{\partial \mathbf{w}} \\ &+ \frac{\partial \hat{\boldsymbol{\sigma}}_t}{\partial \boldsymbol{\alpha}_t}\sum_{\tilde{t}=t-1}^{1}\left[\left(\prod_{\tilde{t}=t}^{\tilde{t}+1}\frac{\partial \boldsymbol{\alpha}_{\tilde{t}}}{\partial \boldsymbol{\alpha}_{\tilde{t}-1}}\right)\frac{\partial \boldsymbol{\alpha}_{\tilde{t}}}{\partial \boldsymbol{\theta}_{\tilde{t}}}\frac{\partial \boldsymbol{\theta}_{\tilde{t}}}{\partial \mathbf{w}}\right]\Bigg\}\end{aligned} \tag{15}$$

where the remaining gradient chain $\partial\boldsymbol{\theta}/\partial\mathbf{w}$ is computed with conventional backpropagation through the network. If $\mathcal{M}$ is implemented in a code base that allows for automatic differentiation (*e.g.* in PyTorch), these time dependencies are naturally taken into account as long as a persistent gradient tape is used within each strain path.[5] In this work we instead implement network training directly into an existing FE code, and therefore opt for the pragmatic approach of computing all partial derivatives of quantities derived from $\mathcal{M}$ using finite differences.

Finally, in order to enforce upper and lower bounds for $\boldsymbol{\theta}$ and avoid unphysical parameter values (*e.g.* negative elasticity moduli), we apply sigmoid activation to the final layer of the network and scale the parameters back from a $[0, 1]$ range using predefined bounds:

$$\theta_i = \theta_i^{\text{low}} + \theta_i^{\sigma}\left(\theta_i^{\text{upp}} - \theta_i^{\text{low}}\right) \tag{16}$$

3.4. Material decoders

As previously mentioned, any constitutive model can in principle be used as $\mathcal{M}$. For the present study we focus on reproducing elastoplasticity and therefore narrow our choices down to the following set of potential decoders with increasing levels of complexity. The simplest one is a linear-elastic isotropic material with no internal variables:

$$\sigma_{ij} = D_{ijkl}\varepsilon_{kl} \quad \text{with} \quad D_{ijkl} = G\left(\delta_{ij}\delta_{kl} + \delta_{il}\delta_{jk}\right) + \left(K - \frac{2}{3}G\right)\delta_{ij}\delta_{kl} \tag{17}$$

where index notation is used for convenience. For this model, $\boldsymbol{\theta}$ comprises only the bulk and shear moduli K and G, or equivalently the Young's modulus E the Poisson's ratio ν.

The second decoder option is a simple plasticity model with J_2 (von Mises) flow. The stress update in this case becomes:

$$\sigma_{ij} = D_{ijkl}\left(\varepsilon_{ij} - \varepsilon_{ij}^{\text{p}}\right) \tag{18}$$

where strain is additively decomposed into elastic and plastic ($\boldsymbol{\varepsilon}^{\text{p}}$) contributions. The yield criterium and plastic flow rule are given by:

$$\phi = \sqrt{3J_2^{\sigma}} - \sigma_{\text{y}} \leq 0 \quad \text{and} \quad \Delta\varepsilon_{ij}^{\text{p}} = \Delta\gamma\sqrt{\frac{3}{2}}\frac{S_{ij}}{\left\|S_{ij}\right\|_{\text{F}}} \tag{19}$$

where $\mathbf{S}$ is the deviatoric part of the stresses, γ is a plastic multiplier, σ_{y} is a yield stress parameter and we write the Frobenius norm as $\|\cdot\|_{\text{F}}$. In order to keep the model as simple as possible, we assume σ_{y} is a material constant and therefore end up with a perfectly-plastic model with associative flow. The internal variables of this model are components of the plastic strain vector $\boldsymbol{\varepsilon}^{\text{p}}$ and the only new material parameter is the yield stress σ_{y}.

Finally, we also consider the more complex pressure-dependent, non-associative plasticity model proposed by Melro et al. (2013). Stress update is the same as in Eq. (18), but yield surface and plastic flow are given by:

$$\begin{aligned}&\phi = 6J_2^{\sigma} + 2I_1^{\sigma}\left(\sigma_{\text{c}} - \sigma_{\text{t}}\right) - 2\sigma_{\text{c}}\sigma_{\text{t}} \leq 0 \quad \text{and} \\ &\Delta\varepsilon_{ij}^{\text{p}} = \Delta\gamma\left(3S_{ij} + \frac{1 - 2\nu_{\text{p}}}{1 + \nu_{\text{p}}}I_1^{\sigma}\delta_{ij}\right)\end{aligned} \tag{20}$$

where δ_{ij} is the Kronecker delta, σ_{t} and σ_{c} are yield stresses in tension and compression, respectively, and ν_{p} is a new parameter controlling plastic contraction and allowing for compressible plastic flow. Hardening can be described by making the yield stresses general functions of $\boldsymbol{\varepsilon}^{\text{p}}$, but when used as a decoder we assume σ_{t} and σ_{c} do not depend on $\boldsymbol{\varepsilon}^{\text{p}}$ and instead let the decoder $\mathcal{D}$ describe their evolution.

[4] Even though our observations come from a computer model and can be considered noiseless, the surrogate $\hat{\boldsymbol{\sigma}}$ is in general not arbitrarily flexible and the random variable ξ is therefore still necessary to explain why the model does not exactly fit every single observation in the dataset.

[5] This is already the case for RNNs, so switching from RNNs to the present model should require little to no changes to the way training is performed.

The model by Melro et al. (2013) is also the one used to describe the microscopic material phase responsible for the nonlinear behavior observed when homogenizing micromodel response, and can therefore be seen as the natural choice for $\mathcal{M}$. Nevertheless, the other two decoders can provide interesting insights on the effect of introducing different levels of bias to the hybrid model.

3.5. Online predictions and inherited stability

The architecture of Fig. 1 is developed to be minimally intrusive and allow for existing material models to be used as decoders with minimum effort. We therefore implement the online routine of the model as a wrapper around an existing implementation of $\mathcal{M}$. The basic structure of the wrapper can be seen in Algorithm 1. The hybrid nature of the model allows for a robust approach that ensures the numerical stability of the original model $\mathcal{M}$ is inherited by the surrogate. This is achieved by only updating $\boldsymbol{\theta}$ at the end of each time step, after the global implicit Newton–Raphson scheme converges. Material properties are therefore fixed while the global solver is iterating, and that means the tangent stiffness $\mathbf{D}$ comes directly from $\mathcal{M}$ and inherits its stability features.

Algorithm 1: Material wrapper implementing the online component of the hybrid surrogate.

Input: strain $\boldsymbol{\varepsilon}^{\Omega}_{new}$ at macroscopic Gauss point
Output: stress $\boldsymbol{\sigma}^{\Omega}$ and stiffness $\mathbf{D}^{\Omega}$ at macroscopic Gauss point

use nested model with converged parameters and internal state: $\left(\boldsymbol{\sigma}^{\Omega}, \mathbf{D}^{\Omega}, \boldsymbol{\alpha}_{new}\right) \leftarrow \mathcal{M}\left(\boldsymbol{\varepsilon}^{\Omega}_{new}, \boldsymbol{\alpha}_{old}, \boldsymbol{\theta}\right)$;
if *global solver has converged* **:**
  store latest converged strain: $\boldsymbol{\varepsilon}_{old} \leftarrow \boldsymbol{\varepsilon}_{new}$;
  commit material history: $\boldsymbol{\alpha}_{old} \leftarrow \boldsymbol{\alpha}_{new}$;
  compute new features: $\boldsymbol{\varphi}_{new} \leftarrow \mathcal{F}\left(\boldsymbol{\varepsilon}_{new}\right)$;
  update model parameters for the upcoming time step: $\boldsymbol{\theta} \leftarrow \mathcal{D}\left(\boldsymbol{\varphi}_{new}\right)$;
if *first global iteration of time step* **and** *Gauss point is unstable* **:**
  stabilize encoder: $\mathcal{D} \leftarrow \texttt{stabilizeNetwork}\left(\boldsymbol{\varepsilon}^{\Omega}_{new}\right)$;
  recompute features: $\boldsymbol{\varphi}_{old} \leftarrow \mathcal{F}\left(\boldsymbol{\varepsilon}^{\Omega}_{old}\right)$;
  recompute model parameters for the current time step: $\boldsymbol{\theta} \leftarrow \mathcal{D}\left(\boldsymbol{\varphi}_{old}\right)$;
return $\boldsymbol{\sigma}^{\Omega}, \mathbf{D}^{\Omega}$

As an example, the J_2 plasticity model of Eq. (19) is unconditionally stable as long as its hardening modulus $h \geq 0$ for any $\left(\boldsymbol{\varepsilon}^{\Omega}_t, \boldsymbol{\alpha}^{\Omega}_t\right)$, which is the case for the perfectly-plastic version we consider here. It then follows that any hybrid surrogate with J_2 decoder is also unconditionally stable. Note that this is only possible because strains are directly passed on to the decoder and would therefore not be an option for conventional surrogates (*e.g.* the RNN of Fig. 3). For those surrogates, the tangent stiffness would come directly from the jacobian of a highly-flexible data-driven model, often at the cost of numerical stability.

3.6. Numerical stabilization

Nevertheless, the decoder $\mathcal{M}$ may be inherently unstable even with fixed material constants. This is for instance the case for the model by Melro et al. (2013): the non-associative flow rule of Eq. (20) can cause the tangent stiffness $\mathbf{D}^{\Omega}$ to lose positive definiteness under certain strain conditions and for certain combinations of model parameters. To accommodate such a scenario and open up the possibility for online model adaptivity in other contexts, we propose a scheme for updating the encoder $\mathcal{D}$ on the fly in order to enforce extra constraints locally.

Back to Algorithm 1, at the beginning of a new time step we keep $\boldsymbol{\theta}$ fixed to the one obtained with converged strains from the previous step and let the solver make a first strain prediction. After this first iteration, a stability criterion is checked and used to define a new loss function that can be used to update network weights in case instability is detected. Here we employ the determinant of the acoustic tensor $\mathbf{Q}$:

$$\mathbf{Q} = \mathbf{n}_d^{\mathrm{T}} \mathbf{D}^{\Omega} \mathbf{n}_d \tag{21}$$

where $\mathbf{n}_d$ is the vector normal to the strain localization direction creating the instability, which we find through an angle sweep procedure as in Van Der Meer and Ke (2022): we sample the space of possible localization angles densely (a one-dimensional search for 2D strain simulations) with 100 equally-spaced angles, compute $\mathbf{Q}$ and its determinant for each corresponding normal vector $\mathbf{n}_d$ and pick the angle that leads to the lowest value for the determinant. We use $\det(\mathbf{Q})$ as a metric of stability and trigger a retraining procedure in case a negative value is detected. We therefore introduce a new loss function:

$$L_{\mathrm{Q}} = -\frac{\langle \det(\mathbf{Q}) \rangle_-}{\det\left(\mathbf{Q}_0\right)} \tag{22}$$

where $\langle \cdot \rangle_-$ extracts the negative part of its operand and $\mathbf{Q}_0$ is a reference value for the acoustic tensor computed at the start of the simulation. We minimize this new loss at every unstable point for a small number of epochs with low learning rate, and to discourage significant drifts from the original model[6] we finish the stabilization procedure by updating the network using the original loss of Eq. (14) for a single minibatch. Finally, $\boldsymbol{\theta}$ is updated using the retrained model and is kept fixed for the remaining iterations.[7] Note that the local constraint of Eq. (22) is therefore only enforced in a soft way and remaining instabilities might still cause the global solver to diverge, in which case we cancel the current increment, go back to the beginning of the time step and allow for the procedure to be triggered again.

4. Numerical examples

The proposed model was implemented in an in-house Finite Element code developed using the open-source C++ numerical analysis library Jem/Jive (Nguyen-Thanh et al., 2020). In order to allow for seamless online retraining, network training was also implemented within the same code. We start this section by describing the datasets and model selection strategies used to build the surrogates. We then investigate the performance of the approach under several choices of encoders and decoders. Finally, we use the model within an FE^2 simulation and demonstrate the online stabilization procedure of Section 3.5. All simulations are performed on cluster nodes equipped with Xeon E5-2630V4 processors and 128 GB RAM running CentOS 7.

4.1. Data sampling and model selection

Models are trained to reproduce the behavior of the fiber-reinforced composite micromodel shown in Fig. 6. Fibers are modeled as linear-elastic and the matrix is described by the pressure-dependent non-associative elastoplastic model by Melro et al. (2013) (Section 3.4). Microscale material properties are adopted from Van der Meer (2016). The microscopic geometry shown in Fig. 6 results from an RVE study performed in Van der Meer (2016) and is therefore considered representative. Following the discussion in Section 3, our aim is to investigate up to which extent it is possible to circumvent the curse of dimensionality associated with path dependency by training surrogates exclusively on monotonic strain paths and having time-dependent behavior arise

[6] The two loss functions of Eqs. (14) and (22) are never optimized together within the same minibatch, which prevents a scenario in which the two objectives have conflicting gradients and cause training to stall. Nevertheless, optimizing them in alternation might lead to the model gradually losing accuracy on the original training dataset.

[7] Changing $\mathcal{D}$ and therefore $\boldsymbol{\theta}$ after every iteration would not work in favor of improving stability, but rather have the opposite effect.

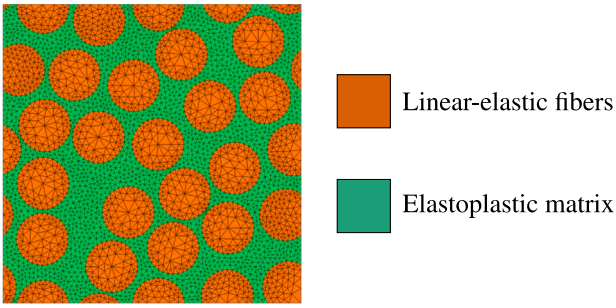


Fig. 6. The micromodel used in the examples of this work.

naturally from a physics-based decoder. We therefore limit ourselves to monotonic paths for training. For consistency, we also employ exclusively monotonic data to perform model selection.

For efficiency, we limit the present investigation to 2D simulations (*i.e.* three strain components) in the plane perpendicular to the fibers, but nevertheless expect the discussion and conclusions to generalize to 3D simulations as long as appropriate orthotropic decoders are employed. Datasets with 2000 monotonic strain paths are generated under both plane strain and plane stress assumptions. Fig. 7 shows the complete plane strain dataset, with a similar one also being generated for plane stress. Each path is generated with an FE^2 simulation of a single macroscopic element under displacement control along a fixed direction in strain space sampled from a uniform distribution. To circumvent convergence issues, we employ an adaptive time stepping technique that progressively reduces time step size when the simulation does not converge and gradually increases it back for subsequent increments. The simulations are stopped once a strain norm of 10 % is reached. As the adaptive scheme leads to paths with different numbers of time increments, we balance the dataset by ensuring every path is composed of 30 steps with strain norms as equally spaced as possible.

To keep model selection straightforward and avoid the need for cumbersome k-fold cross validation or bootstrapping, we train a preliminary model with enough flexibility and an extensive training dataset and gradually increase the size of the validation set until the validation error converges to a good estimate of the expected prediction error (Hastie et al., 2009). This results in validation sets with 500 paths selected at random from the original datasets, leaving 1500 paths to be used for training. We then perform model selection by gradually increasing the complexity of our FNN encoders until the validation error stabilizes. From experimenting with different architectures, we find that encoders with 5 hidden layers of 50 units each with Scaled Exponential Linear Unit (SELU) (Klambauer et al., 2017) activation provide enough flexibility for all the examples treated here. To ensure enough regularization when computing learning curves with small datasets, we employ Bernoulli dropout layers with a rate of 1 % after every hidden layer. Networks are trained for 20 000 epochs using the Adam optimizer with recommended parameters (Kingma and Ba, 2017) and 32-path minibatches. The model with lowest historical validation error is kept after training, further reducing the risk of overfitting on small datasets.

To assess the capabilities of the trained surrogates, we compute an additional test dataset comprising 50 monotonic, 50 unloading–reloading and 50 slow cycling paths, examples of which are shown in Fig. 8. To keep the comparisons fair, none of these paths are used to perform model selection and are therefore only considered after the surrogates are trained. We will use example curves like those from Fig. 8 for visual inspection of the model performance, but also the complete sets of 50 curves each for more rigorous statistical analysis. A summary of the models considered in the following sections can be seen in Table 1.

Table 1
Summary of the hybrid models used in the numerical examples.

Features	Encoder	Latent space	Decoder	Section
$\varepsilon_{xx}, \varepsilon_{yy}, \gamma_{xy}$ $\boldsymbol{\varepsilon} \to I_1^{\varepsilon}, I_2^{\varepsilon}$ $\boldsymbol{\varepsilon} \to \bar{I}_1^{\varepsilon}, \bar{I}_2^{\varepsilon}$	5 × 50 SELU	E, ν	Elastic	4.2
$\boldsymbol{\varepsilon} \to I_1^{\varepsilon}, I_2^{\varepsilon}$	5 × 50 SELU	σ_y	J_2 plasticity	4.3
$\varepsilon_{xx}, \varepsilon_{yy}, \gamma_{xy}$ $\boldsymbol{\varepsilon} \to I_1^{\varepsilon}, I_2^{\varepsilon}$ $\boldsymbol{\varepsilon} \to I_1^{\varepsilon}, J_2^{\varepsilon}$ $\boldsymbol{\varepsilon} \to \mathcal{M} \to \varepsilon_{xx}^{\mathrm{p}}$ $\boldsymbol{\varepsilon} \to \tilde{\mathcal{M}} \to I_1^{\tilde{\sigma}}, J_2^{\tilde{\sigma}}$	5 × 50 SELU	$\sigma_t, \frac{\sigma_c}{\sigma_t}, \nu_p$	(Melro et al., 2013)	4.4, 4.5

4.2. Elastic decoder

It is interesting to first consider the simple linear-elastic decoder of Eq. (17), as it has no internal variables and therefore leads to a surrogate model comparable in nature to a conventional FNN trained on stress–strain pairs. As we will demonstrate, however, the limited physical bias provided by such simple model already proves advantageous. Here we let both elastic properties be controlled by the learned encoder:

$$\boldsymbol{\theta} = \begin{bmatrix} E & \nu \end{bmatrix} \tag{23}$$

where the bounds $10^1 < E < 10^5$ and $0 < \nu < 0.5$ are enforced as described in Eq. (16).

We first perform a feature selection study and investigate how efficiently the model learns as the size of the dataset is increased. From the original plane strain training dataset of 1500 monotonic strain paths, we draw datasets with sizes ranging between 1 and 150 paths without replacement and use them to train networks with different encoder features. To get a reliable estimate of the expected prediction error, we repeat this process 50 times for each dataset size and encoder type, and for comparison we also do the same for conventional FNNs trained directly on stress targets (keeping the same architecture but going directly from the final hidden layer to stresses). This amounts to a total of 3400 trained networks from which we can compute an estimate of the prediction error by averaging $\|\boldsymbol{\sigma} - \hat{\boldsymbol{\sigma}}\|$ over the 500 paths left for validation.

Fig. 9(a) plots averages of the validation error over the 50 training datasets used for each size. Although the hybrid architecture does not show an advantage over the FNN when the encoder is trained on strain features, there is a clear gain in learning speed when using only the two first strain invariants as features. Apart from accelerating learning and resulting in lossless dimensionality reduction, using invariants also results in a surrogate which is frame invariant under small strains. For comparison, we also train a conventional FNN on the same set of features, but those are unsurprisingly not enough to describe general strain states and much of the material response is interpreted by the FNN as observation noise. We zoom into the first part of the learning curves in Fig. 9(b), this time also showing single standard deviation uncertainty bands coming from the variance among the 50 training datasets. The hybrid network outperforms conventional FNNs in the low data regime and tends to be less sensitive to changes in dataset starting from about 20 training paths. Nevertheless, the extra flexibility of conventional FNNs allow them to achieve lower validation errors if significantly more training paths are used.

Training the invariant-based hybrid network with the complete dataset of 1500 curves leads to surrogates with validation errors of about 4 MPa, accurately representing the monotonic behavior of the original micromodel. Fig. 10 shows representative predictions of this model for paths from the test set. It is also interesting to plot how the latent space of the hybrid model evolves within a single strain path. This is done for the monotonic path of Fig. 10a in Fig. 11

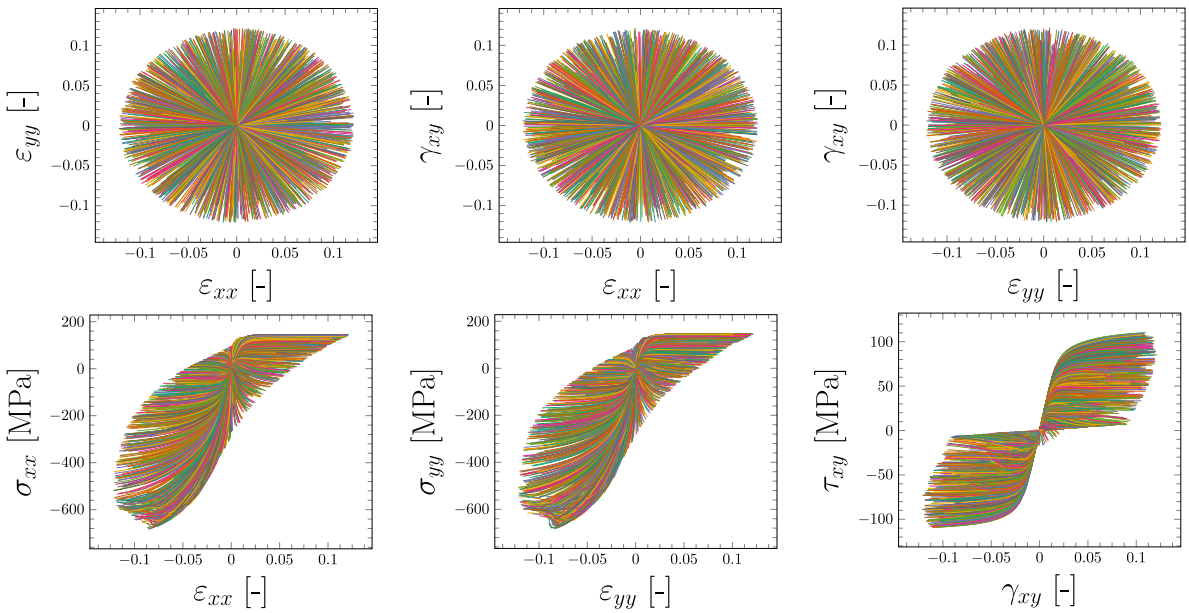
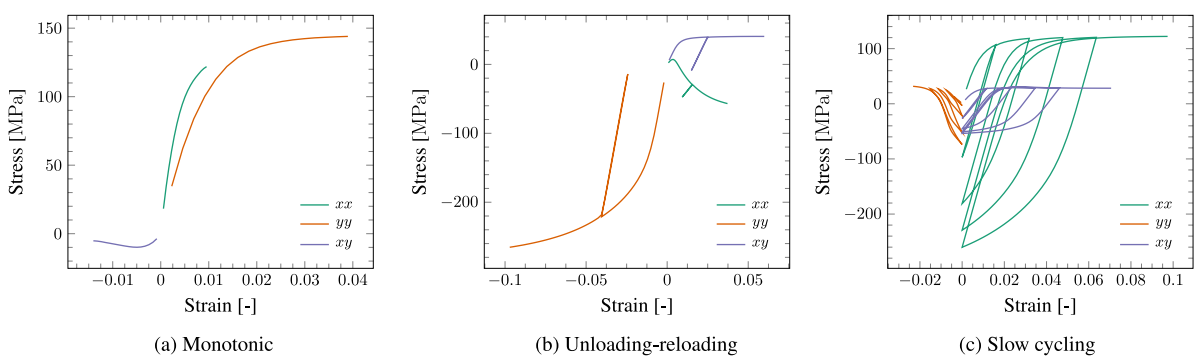
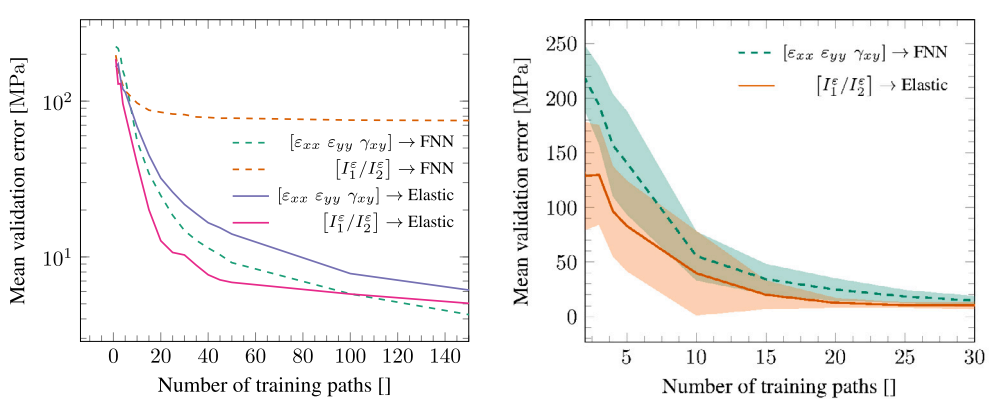


Fig. 7. The complete plane strain dataset used to train the surrogates, comprising 2000 monotonic strain–stress paths. A similar dataset is generated under plane stress conditions.



(a) Monotonic (b) Unloading-reloading (c) Slow cycling

Fig. 8. Examples from a test dataset with 50 paths of each type. They are not used to train any of the networks or perform model selection.



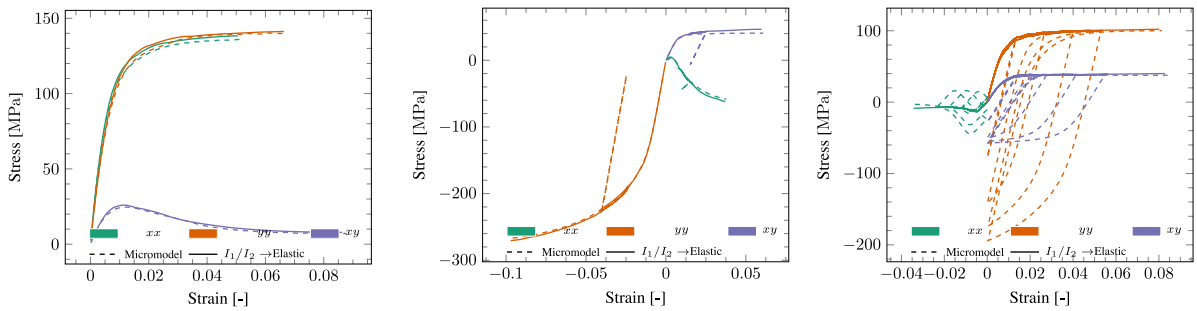
(a) Expectations over 50 datasets of each size (b) Detailed comparison including standard deviations

Fig. 9. Learning curves of models with elastic decoders and conventional FNN models. Mean error over the 500 validation monotonic paths.

(solid lines). In order to reproduce exponential hardening followed by perfect plasticity, the value of E decays asymptotically to zero while ν stabilizes at a relatively high level.

As can be seen in Fig. 10, this surrogate with no internal variables is not capable of predicting non-monotonic strain paths, and effectively behaves like a hyperelastic material model just as the conventional FNN would. Nevertheless, the flexible and interpretable encoder–decoder architecture of Fig. 1 allows for new creative approaches in feature selection. As a demonstration, we keep the trained network of Fig. 10 intact and only modify its feature extractor to introduce a simple path-dependent mechanism:

$$\boldsymbol{\varphi}_T \equiv \begin{bmatrix} \overline{I}_1^\varepsilon & \overline{I}_2^\varepsilon \end{bmatrix}_T = \underset{0<t<T}{\operatorname{argmax}} \left(\left(I_1^\varepsilon\right)_t^2 + \left(J_2^\varepsilon\right)_t \right) \tag{24}$$



(a) Monotonic (b) Unloading-reloading (c) Slow cycling

Fig. 10. Performance of the elastic decoder model for different test scenarios.

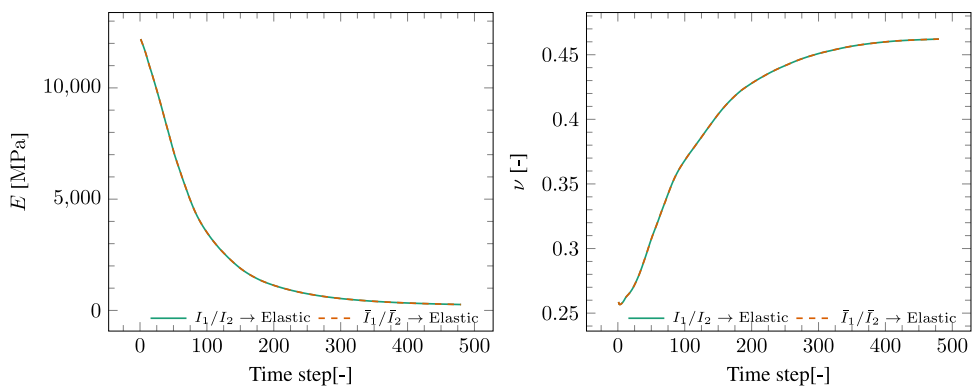


Fig. 11. Evolution of latent space properties of elastic decoder models for a monotonic test path.

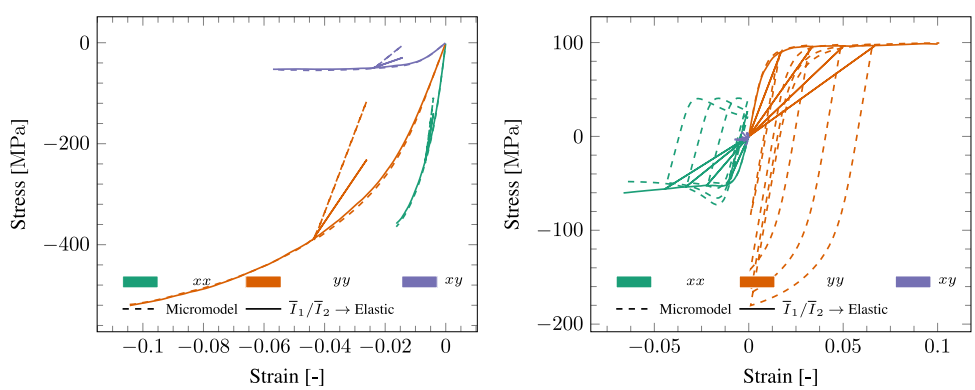


Fig. 12. Predicting unloading with a linear-elastic decoder through history-aware feature extraction.

which freezes the evolution of $\boldsymbol{\theta}$ if the path becomes non-monotonic. Note that the network does not need to be retrained and this modification can be employed exclusively when making online predictions, as the new features reduce to the original ones for the monotonic paths used for training (*c.f.* Fig. 11, dashed line).

We plot two representative non-monotonic paths predicted by the modified model in Fig. 12. From the hyperelastic behavior of Fig. 10, the modified surrogate now behaves as a damage model: the non-linear material behavior is explained by a loss of stiffness which is made persistent by the history-aware feature extractor. This path-dependent behavior can also be observed by looking at the evolution of material properties during the slow cycling path of Fig. 12b, which we plot in Fig. 13. With the original encoder (solid lines), properties are allowed to recover during unloading, while the modified feature extractor (dashed lines) keeps them constant until the path is fully reloaded. Nevertheless, although an improvement to the original model, it is unreasonable to expect the physical bias introduced by a purely elastic model to reliably represent an elastoplastic micromodel. We therefore move to decoders with more relevant physics.

4.3. J_2 decoder

In this section we choose as decoder $\mathcal{M}$ the elastoplastic model of Eq. (19) with J_2 plastic flow. Standing on its own, the model is *a priori* perfectly plastic (constant σ_y),[8] but here we let its yield stress be controlled by the data-driven encoder:

$$\boldsymbol{\theta} = [\sigma_y] \tag{25}$$

[8] We also experimented with models with linear hardening but since the encoder is in principle a universal approximator, no additional flexibility is obtained by assuming more complex hardening behavior. There are also no gains to be booked in terms of numerical stability, as a perfectly-plastic J_2 model is already unconditionally stable.

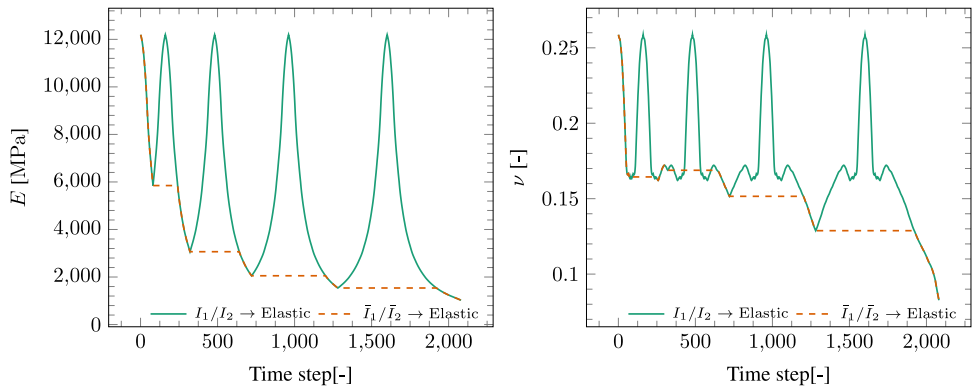


Fig. 13. Evolution of latent space properties of elastic decoder models for a slow cycling test path.

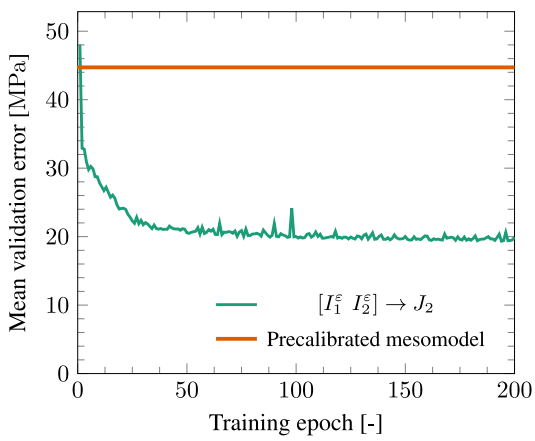


Fig. 14. Evolution of the mean validation loss for the first 200 training epochs of a network with J_2 decoder. Single dataset with 1500 monotonic paths.

while enforcing $10^1 < \sigma_y < 10^3$ and keeping the Young's modulus and Poisson's ratio fixed to values obtained from a single linear micromodel simulation. In contrast to the model with elastic decoder of the previous section, we now employ prior knowledge of the micromodel behavior and assume that all non-linearity should be explained by plasticity and do not let the elastic properties be dictated by the encoder. Still, the assumption of isotropic and incompressible plastic flow is a departure from the more complex pressure-dependent and non-associative behavior shown by the micromodel. Here we are therefore concerned with the effect of trading the flexibility of an elastic decoder for significantly more physical bias from a lower-fidelity representation of material behavior.

At this point it is interesting to compare the performance of the hybrid surrogate with predictions coming from the state-of-the-art mesoscale material model for polymer composites proposed by Vogler et al. (2013). It is an orthotropic elastoplastic model with pressure-dependent non-associative flow precalibrated with a small number of monotonic uniaxial and biaxial stress–strain curves obtained from simulations on the exact same micromodel of Fig. 6 (see Van der Meer, 2016 for details on the calibration procedure). For this section, we switch to a dataset in plane stress, allowing the J_2 model to describe richer nonlinear behavior under biaxial strain states.

Fig. 14 shows the evolution of the validation set loss when training the J_2-decoded model with 1500 plane stress training paths. The error quickly stabilizes at around 20 MPa, significantly lower than the 44 MPa average prediction error obtained with the precalibrated mesomodel. The added flexibility with respect to the original perfectly-plastic J_2 model can be seen in the test set curves plotted in Fig. 15: the data-driven encoder leads to correct predictions of nonlinear hardening (Fig. 15(a)) and pressure-dependent plastic flow (Fig. 15(b)). The figures also highlight the inability of the mesomodel to predict the behavior in certain regions of the strain space, particularly under compression-dominated scenarios.

The minimum validation error attained by the model is, however, nevertheless significantly higher than the 4 MPa obtained with the elastic decoder of the previous section. This result is not entirely surprising, as the elastic decoder introduces much less bias into the model and therefore allows for a greater degree of flexibility when fitting monotonic data. On the other hand, what cannot be directly gleaned from Fig. 14 is that the J_2 decoder benefits from having physics-based memory coming from its internal variables that allows for making predictions of non-monotonic behavior based solely on our assumption that nonlinearities come from plastic strain and therefore without ever having to see it during training.

In Fig. 16 we plot predictions of the J_2 surrogate for two different unloading–reloading paths from the test dataset. The model predicts unloading very well without being trained for it. Nevertheless, as Fig. 14 suggests, the model struggles to predict monotonic behavior under a number of different scenarios, from which it follows that any non-monotonic predictions along the same directions will also be inaccurate. Fig. 17 shows three examples of this behavior.

The choice of decoder therefore involves a tradeoff between bias and flexibility that can be deceiving to base solely on validation error computed on monotonic data. Indeed, the decoder used in the next section outperforms J_2-based decoders in most situations, but nevertheless a choice for the simpler decoder might still be justified — *e.g.* if the unconditional numerical stability of a J_2 decoder is desirable.

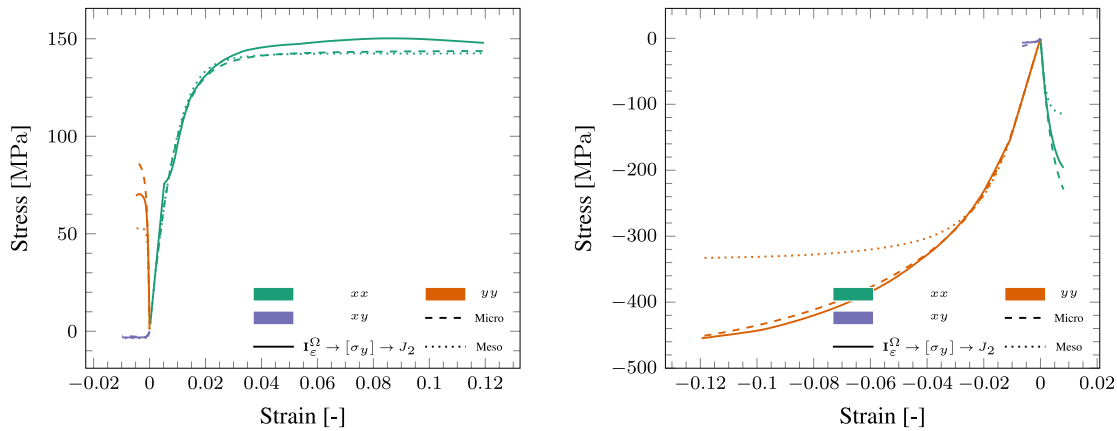
4.4. Non-associative pressure-dependent elastoplastic decoder

As one final exploration on model selection, we use as decoder the same elastoplastic model by Melro et al. used to describe the matrix material at the microscale (Melro et al., 2013). As mentioned in Section 3.4, this model is the natural choice for $\mathcal{M}$, as it attempts to explain the observed microscopic non-linear behavior with the same model from which the behavior arises. As before we keep the elastic properties of the model intact and let only the yield stresses and the plastic Poisson's ratio change in time:

$$\boldsymbol{\theta} = \begin{bmatrix} \sigma_t & \frac{\sigma_c}{\sigma_t} & \nu_p \end{bmatrix} \tag{26}$$

where $10^1 < \sigma_t < 10^4$, $0 < \nu_p < 0.5$ and $1 < \frac{\sigma_c}{\sigma_t} < 100$. We opt for the ratio $\frac{\sigma_c}{\sigma_t}$ instead of simply σ_c in order to also enforce $\sigma_c > \sigma_t$.

We expand upon the feature selection study of Fig. 9 by looking at several feature extractors coming both directly from strains and from the output of a precalibrated Melro model $\overline{\mathcal{M}}$ with the same properties used at the microscale (Fig. 5(b)). As mentioned in Section 3.2, this



(a) Predicting nonlinear hardening

(b) Predicting pressure dependency

Fig. 15. Predictions from the network with J_2 decoder. Letting the yield stress evolve extends the model to more complex plasticity behavior.

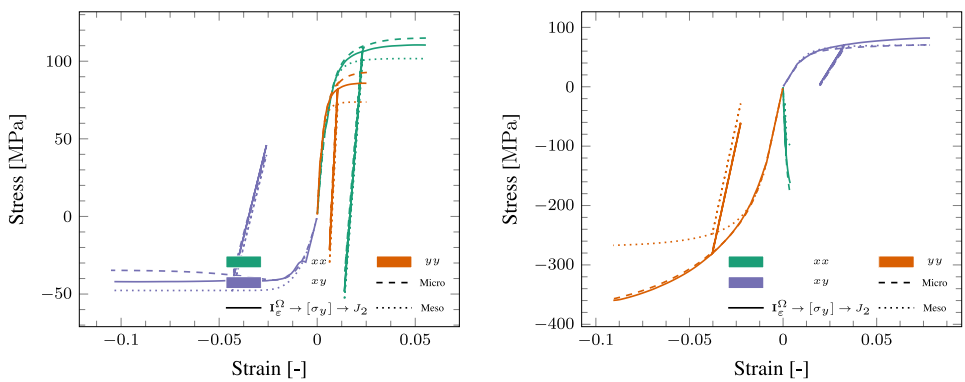


Fig. 16. Network predictions with J_2 decoder for unloading paths after being trained exclusively with monotonic paths.

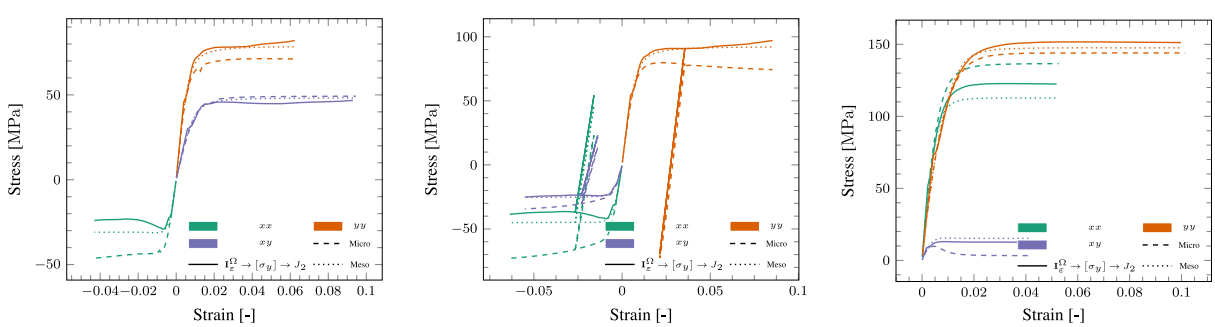


Fig. 17. Examples of strain paths not well predicted by the J_2 decoded model.

precalibrated model has fixed material properties and does not directly perform macroscopic stress predictions but rather only acts as a time convolution operator, converting the complete strain history seen by the surrogate into a single set of input features $\boldsymbol{\varphi}$ for the encoder. By encapsulating the same physics that generates the nonlinear material behavior present in training snapshots, it can therefore be seen as a physics-rich alternative to *e.g.* a network of 1D convolution layers parsing the complete history of strains from $t = 0$ until the time step at which predictions are sought.

Aside from the familiar choice of strain features ($\left[\varepsilon_{xx}\ \varepsilon_{yy}\ \gamma_{xy}\right] \to$ *Melro*), we look into invariants of the strain tensor ($\left[I_1^\varepsilon\ I_2^\varepsilon\right] \to$ *Melro*), combinations including invariants of the deviatoric strain tensor ($\left[J_2^\varepsilon\right] \to$ *Melro*, $\left[I_1^\varepsilon\ J_2^\varepsilon\right] \to$ *Melro*), plastic strain internal variables coming from the precalibrated feature extractor ($\left[\bar{\varepsilon}_{xx}^{\mathrm{p}}\ \bar{\varepsilon}_{yy}^{\mathrm{p}}\ \bar{\gamma}_{xy}^{\mathrm{p}}\right] \to$ *Melro*) and stress invariants coming from the extractor ($\left[I_1^{\bar{\sigma}}\ J_2^{\bar{\sigma}}\right] \to$ *Melro*). We also include predictions from the precalibrated mesomodel by Vogler et al. (2013) and selected curves from Fig. 9(a) for comparison purposes. As before, we train 50 networks of each type for each size of dataset ranging from 1 to 150 paths drawn from the original dataset with 1500 paths. Each trained network is then used to compute the validation error over the 500 monotonic validation paths and the 150 test paths (50 extra monotonic paths, 50 paths with unloading–reloading and 50 slow cycle paths). This results in an extensive study comprising 6800 trained networks and over one million test set simulations.

Results are summarized in Fig. 18, with each point in a curve being the average over 50 networks. Once again using invariants as features proves beneficial, leading to lossless dimensionality reduction and frame invariant surrogates. All tested models perform better than the precalibrated mesomodel, with a gap of more than one order of

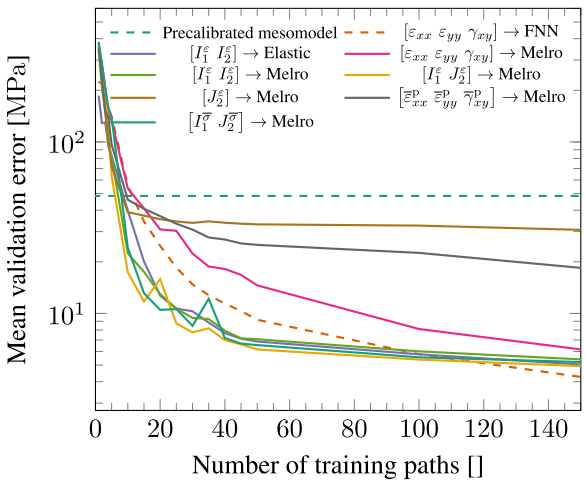


Fig. 18. Expected validation errors for Melro-decoded surrogates with different feature extractors (averages over 50 datasets).

magnitude for the best performing surrogates. Interestingly, models with Melro-based decoders seem to learn as fast and be as flexible as models with elastic decoders, already for the monotonic curves in the validation dataset. This suggests that the new decoder does not impose extra undesirable bias in learning the specific material behavior treated here other than the assumptions that had already been introduced by elasticity (*e.g.* symmetries and couplings encoded by the elastic stiffness tensor). Any benefits reaped when extrapolating to non-monotonic paths, as we will see in the following, are therefore obtained at a negligible price in terms of monotonic behavior accuracy. This stands in contrast with the discussion on the J_2 decoder of the previous section.

Although Fig. 18 is not enough to discern between several of our encoder choices, it is interesting to take a closer look at the two clearly underperforming options. Fig. 19 shows predictions from $\left[J_2^\varepsilon\right] \to$ *Melro* and $\left[\bar\varepsilon^{\mathrm{p}}_{xx}\ \bar\varepsilon^{\mathrm{p}}_{yy}\ \bar\gamma^{\mathrm{p}}_{xy}\right] \to$ *Melro* for the same monotonic test path. The model with a single feature struggles to predict the entirety of the path, indicating that further reducing the dimensionality of the feature space is not possible for this dataset. The oscillatory stress predictions make this model unsuitable for online stress evaluation in a multiscale setting. For the model with plastic strain features, the feature extractor shows no plastic strains until high stress levels while in the micromodel plasticity starts much earlier, forcing the surrogate to remain in the elastic regime until a sudden jump brings it back to the expected path.

Moving to unloading–reloading paths, we compare the performance of different feature sets by plotting the average test error over the 50 unloading–reloading paths in Fig. 20(a). Here an interesting observation can be made: even the surrogate $\left[I_1^\varepsilon\ I_2^\varepsilon\right] \to$ *Elastic* — which cannot predict unloading at all — attains a lower test error than the precalibrated mesomodel. This apparent contradiction can be explained by plotting in Fig. 20(b) the average error computed only at unloading or reloading time steps: use of an elastic decoder — and therefore of a conventional FNN or an RNN trained with insufficient data — excels at predicting monotonic response but is consistently inaccurate for non-monotonic paths and shows little improvement when more monotonic paths are added to the training dataset.

In contrast, the best-performing Melro models are consistently more accurate than the precalibrated mesomodel even when trained on very little data. We plot in Fig. 21 selected representative unloading paths from the test dataset for four of the surrogates. Unloading is once again well captured without having been seen during training, and since it emerges from a purely physical mechanism, it is reasonable to expect unloading at different points along the path to yield comparable results (*c.f.* Fig. 3(a)). Nevertheless, relatively small differences in unloading slope can still lead to large differences in stress at the end of the unloading branches. Furthermore, the model can struggle with tension–compression switches and predict spurious hysteresis loops.

Indeed, we observe a consistent inability by the models to properly predict switches between tension and compression within the same path. This becomes clear when looking at slow cycling test paths composed of several of these switches (Fig. 8(c)). We plot learning curves for the test error on slow cycling paths in Fig. 22, for complete paths as well as exclusively for the non-monotonic branches of the paths. In contrast with results up until now, here we see larger differences in performance for different feature sets. As expected, elastic decoders are once again shown to be unsuitable to predict non-monotonic paths, and the difference here is even more pronounced than in for single-unloading paths (*c.f.* Fig. 20) as most of the path is composed of unloading/reloading branches. The model encoded with stress invariants coming from an elastoplastic feature extractor performs best among the models we test. But crucially, none of the surrogates manages to surpass the precalibrated mesomodel in this case.

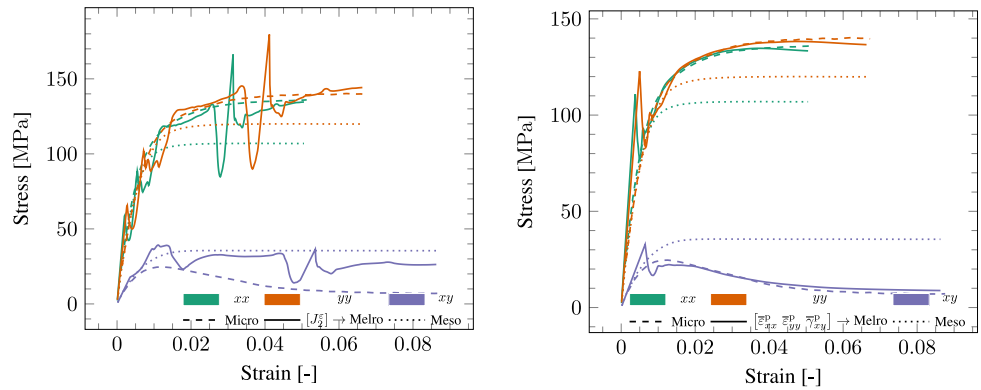
As a demonstration, we select a representative path from the test dataset and plot predictions made with four different feature sets in Fig. 23. As expected, larger errors are observed for more pronounced tension–compression switches as models either over- or undershoot the stress levels at compression–tension switch points. Interestingly, most models manage to converge back to the correct stress path after reloading, since hardening behavior is completely dictated by their non-recurrent data-driven encoders. The exception is the model with stress invariant features ($\left[I_1^{\bar\sigma}\ J_2^{\bar\sigma}\right] \to$ *Melro*) which we plot in more detail in Fig. 24 for each stress component separately, performing significantly better than the rest but showing a number of undesired oscillations in stress response due to the (physically) recurrent nature of its features forcing its neural network encoder to operate in extrapolation.

4.5. FE^2 example

We conclude our discussion with an FE^2 demonstration using the proposed hybrid surrogate. We model the tapered macroscopic bar with geometry and boundary conditions shown in Fig. 25. The model is meshed with 1620 linear triangles with a single Gauss point each and is loaded in tension until plastic strain localization takes place. The combination of the tapered geometry with the several circular voids along the model result in a complex range of stress states throughout the model. In contrast to the cases considered so far, this example also covers non-proportional strain paths. To facilitate convergence, the substepping approach proposed in Somer et al. (2009) is employed and an adaptive stepping algorithm is used at the macroscale that automatically reduces time step size and recomputes the current increment if either the micro- or macroscopic Newton–Raphson solver fails to converge.

We use the $\left[I_1^{\bar\sigma}\ J_2^{\bar\sigma}\right] \to$ *Melro* model of the previous section as surrogate, trained on the complete set of 1500 monotonic training strain paths. The global load–displacement curve at the right edge of the model is plotted for the full-order FE^2 solution and using the hybrid surrogate in Fig. 26(a). Since we update decoder properties in an explicit fashion (*i.e.* once per time step, see Algorithm 1), we use a displacement increment $\Delta u = 3.5 \times 10^{-3}$ mm for the approximate model, 10 times smaller than the one used for the full-order model.

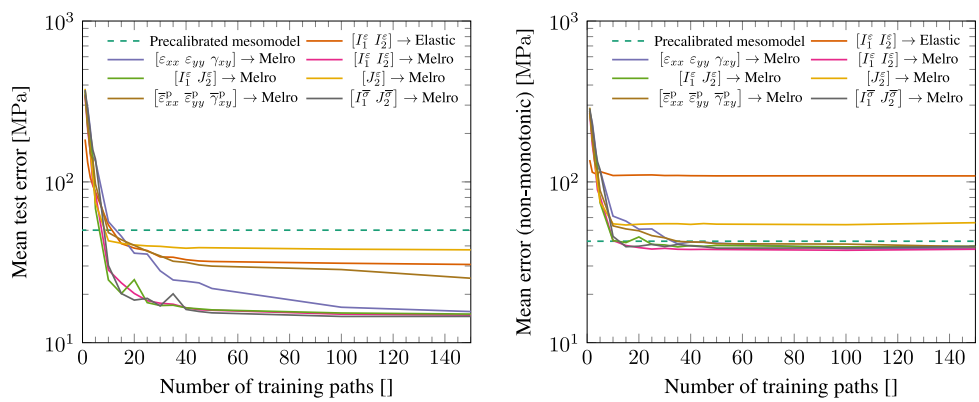
As mentioned in Section 3.5, the model by Melro et al. can suffer from numerical stability issues even with fixed material properties, and it is reasonable to expect these issues to become worse when letting properties evolve with time. Indeed, with no additional stabilization the model using the network fails to converge at the point marked in Fig. 26(a). In contrast, the stabilization procedure of Section 3.5 allows for a complete path to be obtained. For this first result, we stabilize the network for 5 epochs with a learning rate of 1×10^{-5} for the stabilization loss (Eq. (22)) and 1×10^{-9} for retraining on a single monotonic training path selected at random.



(a) Model with second deviatoric strain invariant as feature

(b) Model with plastic strain features

Fig. 19. Monotonic test set predictions from feature-deficient Melro models (complete training dataset with 1500 paths).



(a) Errors for complete paths

(b) Errors including only unloading/reloading steps

Fig. 20. Learning curves for unloading–reloading test errors of Melro-decoded surrogates (averages of 50 datasets).

We also consider a model with an unloading/reloading switch after the onset of macroscopic plasticity. Results are shown in Fig. 26(b). The surrogate approximates the full-order behavior fairly accurately[9] and several orders of magnitude faster than the full-order model.

It is worthwhile to further comment on the obtained levels of acceleration. On average, the computation of a single monotonic training path takes approximately 250 s, and training a surrogate for 20 000 epochs with the complete 1500-path dataset takes approximately 51 h on a single computing core. This would amount to a total offline computation time of 155 h, higher than the required for the FE^2 simulations of Fig. 26. This would however not be a consistent comparison. The hybrid nature of the proposed approach allows for training accurate surrogates with only a small fraction of this complete dataset. Indeed, Fig. 18 suggests a surrogate trained on 20 paths would offer comparable performance, which would amount to a total offline computation time (dataset generation and training) of approximately 2 h. Furthermore, this offline cost does not scale with the density of the macroscopic mesh being considered or with the number of macroscopic simulations being performed — *e.g.* in a parametric study or design optimization procedure.

[9] The obtained global load–displacement behavior suggests the macroscopic plastic localization mechanism has been correctly captured by the surrogate model. Nevertheless, we cannot guarantee the complete macroscopic plastic strain field has been correctly predicted by the surrogate, and since our FE^2 model does not provide homogenized values for macroscopic plastic strains, a more thorough comparison is left for future investigations.

We now look closer on the performance of the proposed online stabilization approach. We empirically find that retraining the network until every violating material point is fully stabilized is not strictly necessary in order to achieve convergence, and therefore opting for a small number of stabilization epochs proves to be an efficient approach. It is nevertheless interesting to investigate the impact of the number of stabilization epochs and of the subsequent retraining minibatch on the original dataset. We solve the monotonic example of Fig. 26(a) with different numbers of stabilization/retraining epochs ranging from 2 to 100 and compute the validation loss (on the 500-path validation set used for model selection) at the end of every macroscopic time increment in order to keep track of how much the stabilized network deviates from its original pretrained state.

Results are shown together with the corresponding load–displacement curves in Fig. 27. All curves remain stable at first, as stabilization is only triggered when the first unstable points are detected. From that point, models which do not undergo retraining after stabilization lose accuracy at a rate proportional to the number of stabilization epochs. However, this unintuitively does not lead to improved global stability: the loss of accuracy by the surrogate leads to spurious global softening (*c.f.* Fig. 27(b)) which in turn leads to further need for stabilization. Models stabilized for 50 and 100 epochs continuously fail to converge and we opt for terminating the simulation after 100 canceled time increments. On the other hand, models retrained with as little as a single strain path (out of the original 1500) after each stabilization epoch are able to maintain the original model accuracy while offering enough stability gains to allow the simulation to converge until the final step, with little change in global behavior for different stabilization regimes.

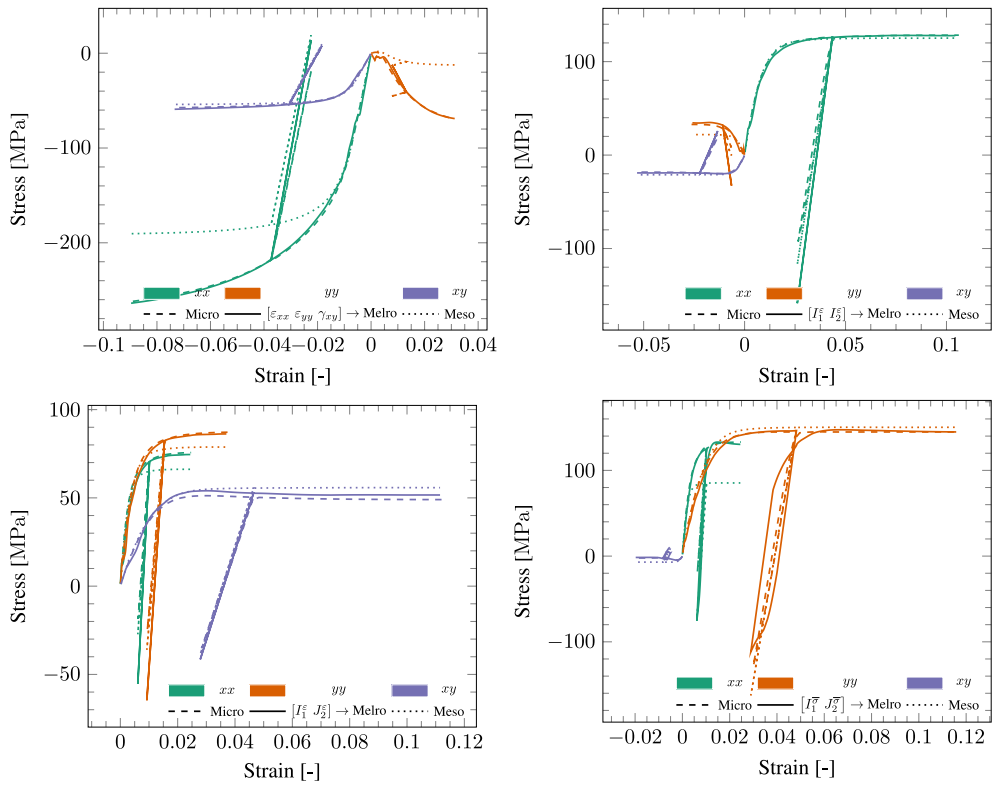
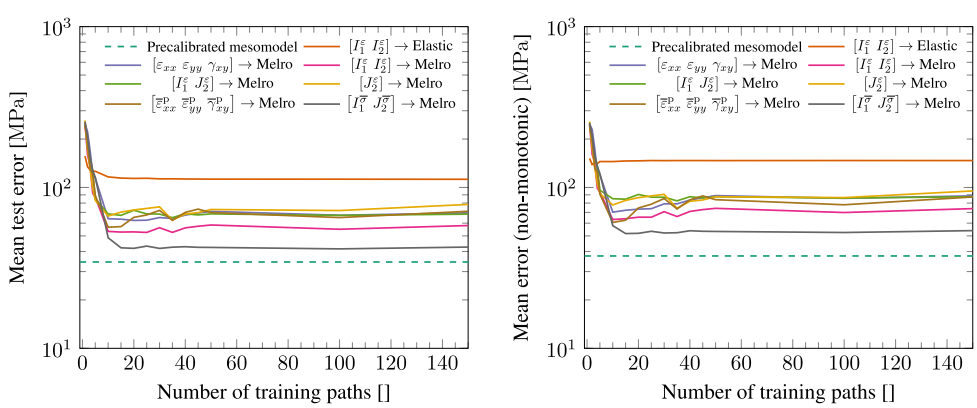


Fig. 21. Response of Melro-decoded surrogates with different features for selected unloading/reloading test paths (1500 monotonic training paths).



(a) Errors for complete paths

(b) Errors including only unloading/reloading steps

Fig. 22. Slow cycling test errors for Melro-decoded surrogates (averages of 50 datasets for each size).

More insight can be obtained on the different stabilization strategies by plotting the cumulative execution time of the simulation and the cumulative number of detected unstable strain states with time increments for different numbers of stabilization epochs. Results can be seen in Fig. 28. In general, simulations without retraining tend to run faster and result in improved stability, although any gains are quickly overshadowed by losses in accuracy (*c.f.* Fig. 27). Stabilizing for more epochs results in a reduction in the total number of unstable points detected, but beyond 5 epochs this does not result in an overall reduction in the computational cost of the simulation given the increased effort spent on individual stabilization operations.

As one final result, we run the monotonic simulation with the hybrid surrogate for different time step sizes. As previously mentioned, the hybrid approach allows for explicit update of $\boldsymbol{\theta}$ within an implicit simulation by obtaining the tangent stiffness matrix directly from the decoder. This however introduces a time step size dependency whose impact merits investigation. We plot in Fig. 29 predictions with step sizes spanning four orders of magnitude, including the same one used to obtain the full-order response. The combination of the explicit property update with the online stabilization procedure indeed introduces an upper bound for time step size for this specific problem. It stands to reason that the sensitivity to time step size also depends on the choice of decoder and on which material properties are included in $\boldsymbol{\theta}$. Further investigation into the matter in future works is therefore warranted.

5. Conclusions

In this paper, we propose a hybrid surrogate modeling architecture

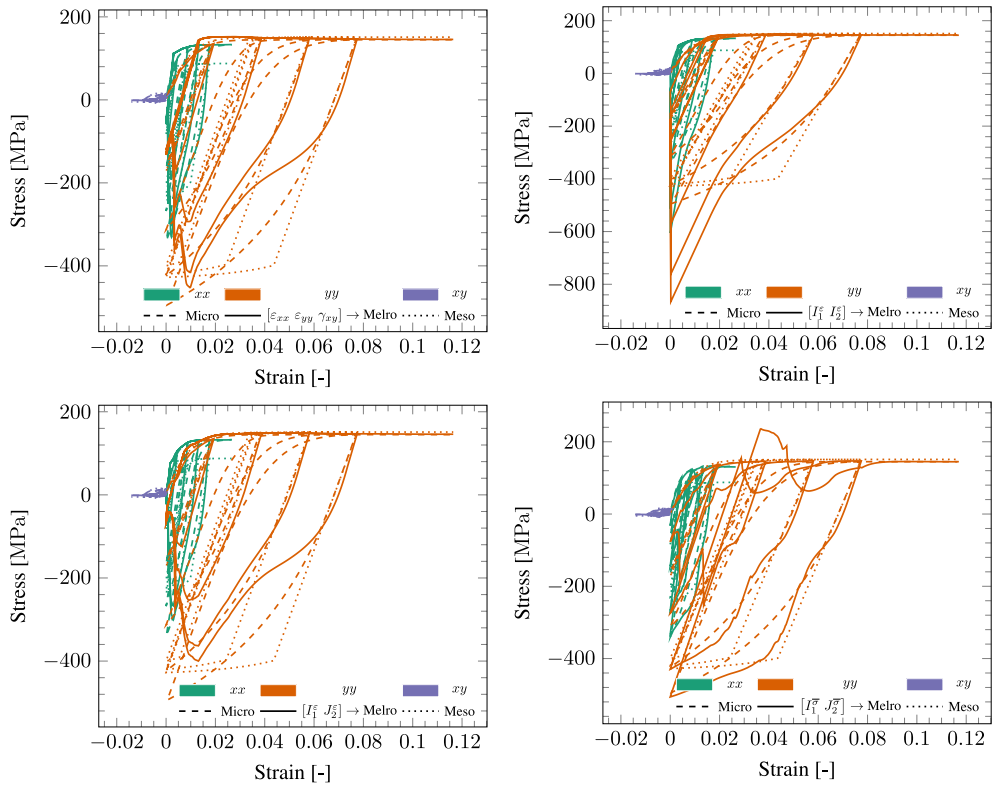


Fig. 23. Response of Melro-decoded surrogates with different features for selected slow cycling test paths (1500 monotonic training paths).

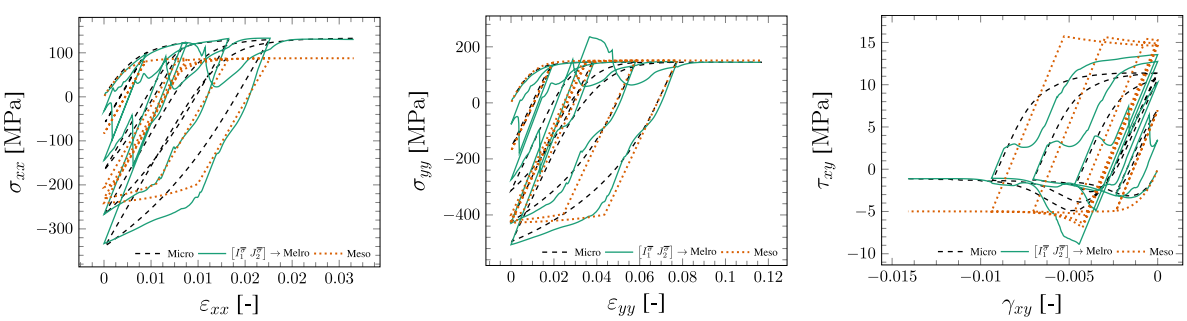


Fig. 24. Slow cycling response of a Melro-decoded surrogate with stress invariant features, different stress components plotted separately.

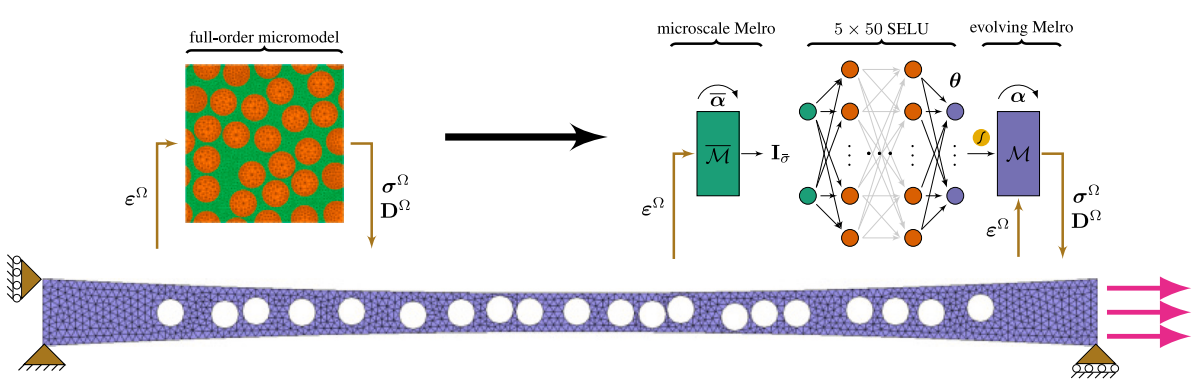
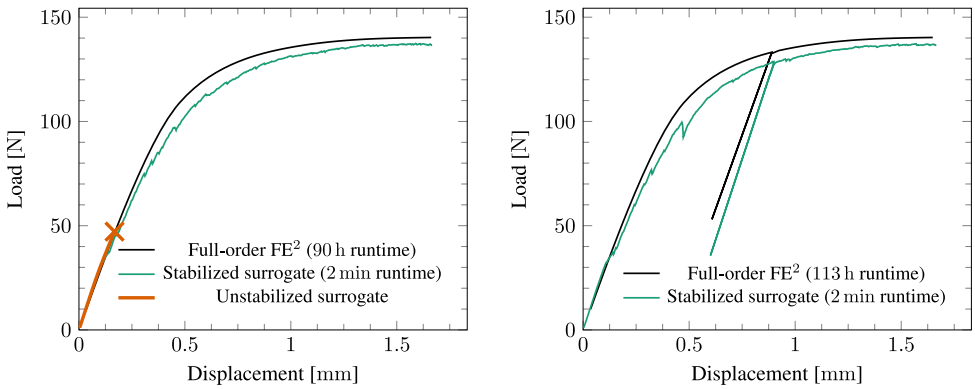


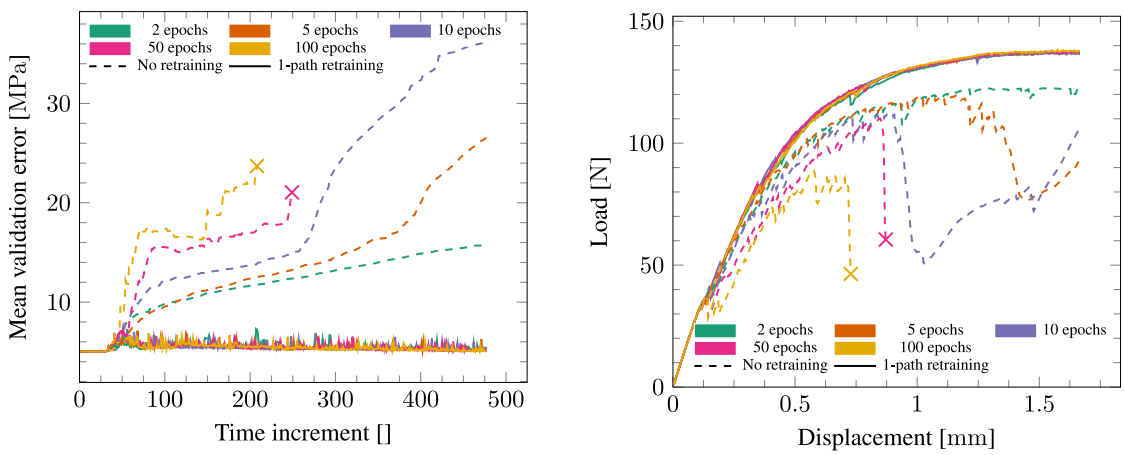
Fig. 25. FE^2 example: geometry, mesh and boundary conditions. Full-order (left) and surrogate-based (right) FE^2 simulations are compared.

for multiscale modeling of heterogeneous materials. The model is composed of a data-driven encoder for material properties and a physics-based decoder that computes stresses. In the resulting architecture, the encoder increases the flexibility of existing material models by letting their properties evolve in time, while the decoder provides beneficial bias and interpretability to the model. The model is conceived with flexibility in mind, allowing existing implementations of physics-based material models to be used with no extra modifications.



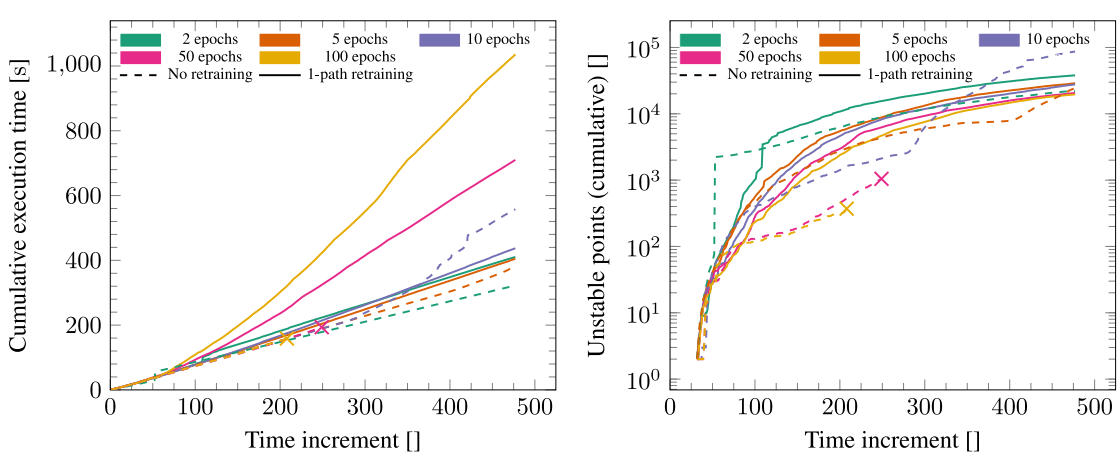
(a) Monotonic (b) Unloading/reloading

Fig. 26. FE^{2}example: load–displacement curves with and without online stabilization, compared to the ground-truth solution.



(a) 500-path validation loss after stabilization (b) Load-displacement curves

Fig. 27. Performance of the surrogate model for stabilization strategies of varying intensities with and without retraining after stabilization.



(a) Computational overhead due to stabilization (b) Number of detected unstable strain states

Fig. 28. Impact of stabilization regime on execution time and number of unstable points throughout the simulation.

Furthermore, by letting the decoder directly receive strain inputs, the encoder architecture is highly flexible and allows for preservation of frame independence. A semi-explicit online prediction algorithm is also proposed that allows for imposing extra constraints to model behavior in a semi-supervised way.

We demonstrate the architecture by reproducing pressure -dependent elastoplastic behavior coming from homogenized fiber-reinforced composite micromodels. The simple model with a linear-elastic decoder learned faster than conventional data-driven surrogates, allowed for lossless feature space dimensionality reduction through the use of strain invariants, and was able to approximate path-dependent

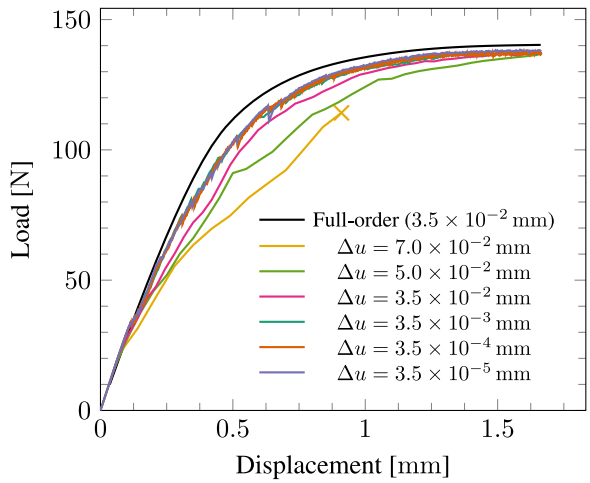


Fig. 29. FE^2 example: Effect of time step size on surrogate predictions.

behavior through a simple history-aware feature extractor. Models with perfectly-plastic J_2 decoders were shown to successfully learn nonlinear hardening and pressure dependency and predict unloading–reloading while being trained exclusively on monotonic data, outperforming a state-of-the-art mesomodel for composites in accuracy for arbitrary loading directions. Employing as decoder the same plasticity model used at the microscale led to highly-accurate monotonic response and fairly accurate extrapolation to unloading/reloading behavior. Finally, the model was used to solve a complex FE^2 model and the benefit of the online stabilization procedure was demonstrated.

We find the approach to be a promising new way to build hybrid surrogates which therefore merits further research on a number of fronts. The current architecture is not by construction concerned with enforcing unconditional thermodynamic consistency or other physical constraints of interest. Although we do find empirically that well-trained surrogates with thermodynamically consistent decoders tend to perform well, some constitutive models might not be suitable for having their properties evolve in time. Fortunately, the framework can cope with extra constraints without necessarily giving up on its flexibility, by enforcing them locally through online retraining. Although training exclusively on monotonic paths already allows for path dependency to be fairly well captured, some decoders might perform better in extrapolation if trained with a (small) number of extra non-monotonic and non-proportional strain paths — for instance when encoder and decoder can each explain the same phenomenon on their own (*e.g.* pressure dependency in the model by Melro et al.). We also foresee combining the present approach with the one in Maia et al. (2023) into a unified family of flexible hybrid surrogates with a range of possible combinations of feature extractors for physics-rich time convolution, fixed-property models with learned strain distributions and evolving material models.

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 code used in this work is publicly available at: https://gitlab.tudelft.nl/ibarceloscarne/evolving-material-models.

Acknowledgments

The authors gratefully acknowledge the TU Delft AI Initiative for their support through the SLIMM AI Lab. FM also acknowledges financial support from the Netherlands Organization for Scientific Research (NWO) under Vidi grant nr. 16464.

References

Abueidda, D.W., Koric, S., Sobh, N.A., Sehitoglu, H., 2021. Deep learning for plasticity and thermo-viscoplasticity. Int. J. Plast. 136, 102852. http://dx.doi.org/10.1016/j.ijplas.2020.102852.

Bessa, M.A., Bostanabad, R., Liu, Z., Hu, A., Apley, D.W., Brinson, C., Chen, W., Liu, W.K., 2017. A framework for data-driven analysis of materials under uncertainty: countering the curse of dimensionality. Comput. Methods Appl. Mech. Engrg. 320, 633–667. http://dx.doi.org/10.1016/j.cma.2017.03.037.

Bessa, M.A., Glowacki, P., Houlder, M., 2019. Bayesian machine learning in metamaterial design: Fragile becomes supercompressible. Adv. Mater. 31 (48), 1904845. http://dx.doi.org/10.1002/adma.201904845.

Bishop, C.M., 2006. Pattern Recognition and Machine Learning. In: Information science and statistics, Springer, New York.

Borkowski, L., Sorini, C., Chattopadhyay, A., 2022. Recurrent neural network-based multiaxial plasticity model with regularization for physics-informed constraints. Comput. Struct. 258, 106678. http://dx.doi.org/10.1016/j.compstruc.2021.106678.

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.

Daniel, T., Casenave, F., Akkari, N., Ryckelynck, D., Rey, C., 2022. Uncertainty quantification for industrial numerical simulation using dictionaries of reduced order models. Mech. Ind. http://dx.doi.org/10.1051/meca/2022001.

Ferreira, B.P., Andrade Pires, F.M., Bessa, M.A., 2022. Adaptivity for clustering-based reduced-order modeling of localized history-dependent phenomena. Comput. Methods Appl. Mech. Engrg. 393, 114726. http://dx.doi.org/10.1016/j.cma.2022.114726.

Feyel, F., 1999. Multiscale FE2 elastoviscoplastic analysis of composite structures. Comput. Mater. Sci. 16 (1), 344–354. http://dx.doi.org/10.1016/S0927-0256(99)00077-4.

Flaschel, M., Kumar, S., De Lorenzis, L., 2021. Unsupervised discovery of interpretable hyperelastic constitutive laws. Comput. Methods Appl. Mech. Engrg. 381, 113852. http://dx.doi.org/10.1016/j.cma.2021.113852.

Flaschel, M., Kumar, S., De Lorenzis, L., 2022. Discovering plasticity models without stress data. Npj Comput. Mater. 8 (1), 1–10. http://dx.doi.org/10.1038/s41524-022-00752-4.

Furtado, C., Pereira, L.F., Tavares, R.P., Salgado, M., Otero, F., Catalanotti, G., Arteiro, A., Bessa, M.A., Camanho, P.P., 2021. A methodology to generate design allowables of composite laminates using machine learning. Int. J. Solids Struct. 233, 111095. http://dx.doi.org/10.1016/j.ijsolstr.2021.111095.

Gantenbein, S., Colucci, E., Käch, J., Trachsel, E., Coulter, F.B., Rühs, P.A., Masania, K., Studart, A.R., 2023. Three-dimensional printing of mycelium hydrogels into living complex materials. Nature Mater. 22 (1), 128–134. http://dx.doi.org/10.1038/s41563-022-01429-5.

Gantenbein, S., Mascolo, C., Houriet, C., Zboray, R., Neels, A., Masania, K., Studart, A.R., 2021. Spin-printing of liquid crystal polymer into recyclable and strong all-fiber materials. Adv. Funct. Mater. 31 (52), 2104574. http://dx.doi.org/10.1002/adfm.202104574.

Geers, M.G.D., Kouznetsova, V.G., Brekelmans, W.A.M., 2010. Multi-scale computational homogenization: Trends and challenges. J. Comput. Appl. Math. 234 (7), 2175–2182. http://dx.doi.org/10.1016/j.cam.2009.08.077.

Ghaboussi, J., Garrett, J.H., Wu, X., 1991. Knowledge-based modeling of material behavior with neural networks. J. Eng. Mech. 117 (1), 132–153. http://dx.doi.org/10.1061/(ASCE)0733-9399(1991)117:1(132).

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.

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

Hastie, T., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning: Data Mining, Inference and Prediction, second ed. Springer, New York.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Khajehtourian, R., Kochmann, D.M., 2021. Soft adaptive mechanical metamaterials. Front. Robotics AI 8.

Kingma, D.P., Ba, J., 2017. Adam: A method for stochastic optimization. arXiv:1412.6980.

Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S., 2017. Self-normalizing neural networks. http://dx.doi.org/10.48550/arXiv.1706.02515, arXiv:1706.02515.

Knap, J., Barton, N.R., Hornung, R.D., Arsenlis, A., Becker, R., Jefferson, D.R., 2008. Adaptive sampling in hierarchical simulation. Internat. J. Numer. Methods Engrg. 76 (4), 572–600. http://dx.doi.org/10.1002/nme.2339, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.2339.

Koeppe, A., Bamer, F., Selzer, M., Nestler, B., Markert, B., 2021. Explainable artificial intelligence for mechanics: physics-informing neural networks for constitutive models. http://dx.doi.org/10.48550/arXiv.2104.10683, Number: arXiv:2104.10683 arXiv:2104.10683 [cs].

Kouznetsova, V., Brekelmans, W.A.M., Baaijens, F.P.T., 2001. An approach to micro-macro modeling of heterogeneous materials. Comput. Mech. 27 (1), 37–48. http://dx.doi.org/10.1007/s004660000212.

Krauklis, A.E., Gagani, A.I., Echtermeyer, A.T., 2019. Prediction of orthotropic hygroscopic swelling of fiber-reinforced composites from isotropic swelling of matrix polymer. J. Composit. Sci. 3 (1), 10. http://dx.doi.org/10.3390/jcs3010010.

Kumar, S., Tan, S., Zheng, L., Kochmann, D.M., 2020. Inverse-designed spinodoid metamaterials. Npj Comput. Mater. 6 (1), 1–10. http://dx.doi.org/10.1038/s41524-020-0341-6.

Le, B.A., Yvonnet, J., He, Q.-C., 2015. Computational homogenization of nonlinear elastic materials using neural networks. Internat. J. Numer. Methods Engrg. 104 (12), 1061–1084. http://dx.doi.org/10.1002/nme.4953.

Lefik, M., Boso, D.P., Schrefler, B.A., 2009. Artificial neural networks in numerical modelling of composites. Comput. Methods Appl. Mech. Engrg. 198 (21), 1785–1804. http://dx.doi.org/10.1016/j.cma.2008.12.036.

Linka, K., Hillgärtner, M., Abdolazizi, K.P., Aydin, R.C., Itskov, M., Cyron, C.J., 2021. Constitutive artificial neural networks: A fast and general approach to predictive data-driven constitutive modeling by deep learning. J. Comput. Phys. 429, 110010. http://dx.doi.org/10.1016/j.jcp.2020.110010.

Liu, Z., 2021. Cell division in deep material networks applied to multiscale strain localization modeling. Comput. Methods Appl. Mech. Engrg. 384, 113914. http://dx.doi.org/10.1016/j.cma.2021.113914.

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

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., Vannucci, P., Maffi-Berthier, V., 2021. Thermodynamics-based artificial neural networks for constitutive modeling. J. Mech. Phys. Solids 147, 104277. http://dx.doi.org/10.1016/j.jmps.2020.104277.

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

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.

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

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. PyTorch: An imperative style, high-performance deep learning library. http://dx.doi.org/10.48550/arXiv.1912.01703, Number: arXiv:1912.01703[cs, stat].

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

Rocha, I.B.C.M., Kerfriden, P., van der Meer, F.P., 2020a. 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.B.C.M., Kerfriden, P., van der Meer, F.P., 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.B.C.M., van der Meer, F.P., Sluys, L.J., 2020b. An adaptive domain-based POD/ECM hyper-reduced modeling framework without offline training. Comput. Methods Appl. Mech. Engrg. 358, 112650. http://dx.doi.org/10.1016/j.cma.2019.112650.

Ryckelynck, D., 2005. A priori hyperreduction method: An adaptive approach. J. Comput. Phys. 202 (1), 346–366. http://dx.doi.org/10.1016/j.jcp.2004.07.015.

Scanff, R., Néron, D., Ladevèze, P., Barabinot, P., Cugnon, F., Delsemme, J.-P., 2022. Weakly-invasive LATIN-PGD for solving time-dependent non-linear parametrized problems in solid mechanics. Comput. Methods Appl. Mech. Engrg. 396, 114999. http://dx.doi.org/10.1016/j.cma.2022.114999.

Somer, D.D., de Souza Neto, E.A., Dettmer, W.G., Perić, D., 2009. A sub-stepping scheme for multi-scale analysis of solids. Comput. Methods Appl. Mech. Engrg. 198 (9), 1006–1016. http://dx.doi.org/10.1016/j.cma.2008.11.013.

Telgen, B., Sigmund, O., Kochmann, D.M., 2022. Topology optimization of graded truss lattices based on on-the-fly homogenization. J. Appl. Mech. 89 (6), http://dx.doi.org/10.1115/1.4054186.

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

Van Der Meer, F., Ke, L., 2022. A computational homogenization framework with enhanced localization criterion for macroscopic cohesive failure in heterogeneous materials. J. Theor. Comput. Appl. Mech. http://dx.doi.org/10.46298/jtcam.7707.

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

Vogler, M., Rolfes, R., Camanho, P.P., 2013. Modeling the inelastic deformation and fracture of polymer composites – Part I: Plasticity model. Mech. Mater. 59, 50–64. http://dx.doi.org/10.1016/j.mechmat.2012.12.002.

Wang, K., Sun, W., Du, Q., 2019. A cooperative game for automated learning of elasto-plasticity knowledge graphs and models with AI-guided experimentation. Comput. Mech. 64 (2), 467–499. http://dx.doi.org/10.1007/s00466-019-01723-1.

Wang, J.-J., Wang, C., Fan, J.-S., Mo, Y.L., 2022. A deep learning framework for constitutive modeling based on temporal convolutional network. J. Comput. Phys. 449, 110784. http://dx.doi.org/10.1016/j.jcp.2021.110784.

Woigk, W., Nagel, Y., Gantenbein, S., Coulter, F.B., Masania, K., Studart, A.R., 2022. Flax-based natural composites hierarchically reinforced by cast or printed carbon fibres. Compos. Sci. Technol. 226, 109527. http://dx.doi.org/10.1016/j.compscitech.2022.109527.

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

Xu, Y., Gan, Y., Chang, Z., Wan, Z., Schlangen, E., Šavija, B., 2022. Towards understanding deformation and fracture in cementitious lattice materials: Insights from multiscale experiments and simulations. Constr. Build. Mater. 345, 128409. http://dx.doi.org/10.1016/j.conbuildmat.2022.128409.