Self-consistency Reinforced minimal Gated Recurrent Unit for surrogate modeling of history-dependent non-linear problems: application to history-dependent homogenized response of heterogeneous materials

Ling Wu[a,*], Ludovic Noels[a]

[a] *University of Liege, Department of Aeronautics and Mechanical Engineering, Computational & Multiscale Mechanics of Materials, Allée de la découverte 9, B-4000 Liège, Belgium*

Abstract

Multi-scale simulations can be accelerated by substituting the meso-scale problem resolution by a surrogate trained from off-line simulations. In the context of history-dependent materials, Recurrent Neural Networks (RNN) have widely been considered to act as such a surrogate, since their hidden variables allow for a memory effect.

However, defining a data-set for the training which virtually covers all the possible strain-stress state evolution encountered during the online phase remains a daunting task. This is particularly true in the case in which the strain increment size is expected to vary by several orders of magnitude. Self-Consistent recurrent networks were thus introduced in [C. Bonatti, D. Mohr, On the importance of self-consistency in recurrent neural network models representing elasto-plastic solids, Journal of the Mechanics and Physics of Solids 158 (2022) 104697] to reinforce the self-consistency of the neural network with respect to the input increment size when acting as surrogate of an elasto-plastic material model.

When designing RNN to act as surrogate of meso-scale Boundary Value Problem (BVP) defined by Representative Volume Element (RVE) of complex micro-structures, the number of learnable parameters required for existing Recurrent Neural Network (RNN) to be accurate could remain high, impeding the training performance. In this work, we revisit and design alternative self-consistent recurrent units in order to limit the number of hidden variables required for the neural network to act as a composite material surrogate in multi-scale simulations. Although the RNNs based on the newly suggested self-consistency reinforced recurrent units have a reduced number of learnable parameters yielding good training performance, they remain accurate in the context of multi-scale simulations considering various strain increment sizes.
Preprint submitted to Computer Methods in Applied Mechanics and Engineering (C) 2024; Licensed under the Creative Commons (CC-BY-NC-ND); formal publication on: 10.1016/j.cma.2024.116881

Keywords: Artificial Neural Network, Recurrent Neural Network, Self-consistency, Surrogate, Multi-scale, Elasto-plasticity

1. Introduction

In the recent years, the application of artificial neural networks (ANNs) has been booming in many fields, because of their ability to reproduce non-linear processes. ANNs have increasingly attracted attention in the field of computational mechanics, and extensive research work has been carried out, especially in fields in which material behaviors are complex and/or intensive computation is required. ANNs can be used directly as a surrogate model to substitute, or partly substitute, the physical models in order to reduce the computational cost of numerical analyses.

*Corresponding author, Phone: +32 4 366 96 55, l.wu@ulg.ac.be

In the field of computational mechanics, ANNs have been used as an alternative to the development of constitutive non-linear models in the context of visco-plasticity [1, 2], for cyclic plasticity [3], or again for constitutive law of interface [4, 5], in which experiments or lower scale simulations are used to build or discover these laws. These ANNs can be integrated in a finite element simulation [2, 6–8] avoiding the resolution of material constitutive laws. AANs have also been used for parameters identification of an elasto-plastic material model [9, 10]. In particular, Bayesian inference of multi-scale model parameters of short fiber reinforced composites was conducted in [10], in which ANNs were adopted to substitute complex material homogenization constitutive laws during the random walk.

For history-dependent behaviors of material models, based on specially designed recurrent units, the so called Recurrent neural networks (RNNs) have been adopted. A RNN has a "memory" under the form of hidden variables $\mathbf{h}$ which play a comparable role to the internal or state variables, $\boldsymbol{Z}$, in the analysis of history-dependent physical processes [11]. The RNNs using Long-Short term Memory (LSTM) unit, see e.g. [12–15], and Gated Recurrent Unit (GRU), see e.g. [16–21], possibly in combination with order reduction [11, 22, 23], have widely been adopted as a surrogate of history-dependent responses of material models or micro-structure evolution prediction [24]. Recently, the comparison of different gated recurrent neural networks, a multilayered temporal convolutional network and an encoder–decoder architecture consisting of GRU layers with a modified attention mechanism, has been conducted on predicting the dynamic macro-scale deformation of shock loaded plates from the pressure temporal sequence [25]. We also refer to the reviews [26–28]. Physics-Informed Neural Networks (PINN) have been introduced in [29] and were used to substitute the Partial Differential Equation (PDE) for the resolution of nonlocal variables [30], or the radial return mapping of visco-plastic models [31]. However, currently, this PINN is only applicable for the PDE with constant coefficients.

In spite of the application of ANNs in computational mechanics being very successful, its shortcomings have also emerged. One of the most important one is that the accuracy of ANNs used as surrogate depends largely on the used training data, since data-based models have a weak ability of extrapolation. This requires the use of a large data-set, which is not always available or is expensive to be generated, although multi-fidelity strategies can be adopted [32] when it comes to using numerical results as a synthetic data-set. The concept of ANNs thus inspires the research on developing physical models with reduced order of computation. In [33–36], the process of back-propagation, which is used in ANNs training, has been used to extract the topology of microscopic Representative Volume Elements (RVEs), in order to achieve a micro-mechanics model of reduced computational cost. The derivatives of the free energy and their relationships with dissipation rate, stress and internal variables were represented by feed-forward NNWs in [37] in order to obtain a thermo-dynamically consistent surrogate of an elasto-plastic law. The application of this method to act as a surrogate of homogenized response of RVEs requires dimensionality reduction of the micro-scale problem internal variables [38]. In [39], the feed-forward neural network (FFW) is used to determine the strain apportion in a microscopic Representation Volume Element (RVE), and material models are introduced at some nodes of the FFW for macroscopic stress evaluation.

Nevertheless, supervised learning in terms of strain-stress relationship remains attractive as an efficient computational methods, e.g. to substitute costly RVE homogenization in multi-scale simulations [11, 13]. To substitute the history-dependent behavior of material models, the RNNs are equipped with hidden variables, $\mathbf{h}$, to keep the historical information, and need to be trained with sequential data. As having been pointed out in [40], the RNN, which was trained using sequences with moderate strain increments, may yield totally wrong predictions when it received sequences with tiny strain increments as input. The problem is exacerbated since the strain increment sizes of the training data are usually moderate or large, in order to reach wider strain ranges while avoiding extreme long sequential data: long sequential data will not only increase the computation cost of data preparation, but also make the training of NNWs heavier. However, if the tiny strain increments, such as the norm of strain increment being lower than 10^{-5} or 10^{-6}, is not included in the training data, RNNs' accuracy will decrease in this range of input size. Normally in implicit finite element analyses, extremely small strain increments would not be applied, but a reduced strain increment size may be required when the material response if highly non-linear like in the cases of damage and failure. Besides, the accuracy of RNNs for tiny strain increment sizes becomes a must in explicit finite element analyses as illustrated in [40, 41]. In practice, although this issue can be relieved by adding

such small strain increment sizes in the training data through a random data augmentation as detailed in this work, the best solution is to account for the information of increment size as a part of the input, as suggested by [40].

In [40], a Linearized Minimal State Cell (LMSC) was designed to reinforce the self-consistency of RNNs' predictions with the strain increment size. This RNN takes sequential strain increments as input, and uses the norm of the increment as an extra information. This concept was validated when substituting the constitutive model of an isotropic elasto-plastic material, with von Mises plasticity and isotropic hardening. Such a constitutive model uses totally 6 internal or state variables, but the RNN requires more hidden variables to remain accurate. The question of the number of required internal variables when the RNN substitutes homogenized response of heterogeneous micro-structures, or RVE, while ensuring high training performance of self-consistent RNN, will be investigated in this paper, and alternative recurrent units will be designed.

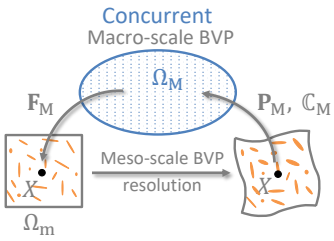


Figure 1: Computational homogenization-based multi-scale models with the coupled resolution of the meso-scale BVPs.

We refer to the computational homogenization based multi-scale analysis which is usually called FE^2 analysis [42, 43]. In such an approach, the macro-scale structure defines a Boundary Value Problem (BVP) which is solved by considering homogenized material properties extracted, at each (macro) material point of interest, from the coupled resolution of meso-scale BVPs defined by a RVE, see Fig. 1. Since the resolution of meso-scale RVEs in a coupled way at every macro-scale material integration points leads to a tremendous requirement of numerical resources in terms of time and memory, substituting this meso-scale resolution by efficient surrogate models becomes very meaningful in practice. In a direct finite element simulation of the meso-scale BVP, usually depending on the adopted RVE, hundreds to thousands of physical state or internal variables $\boldsymbol{Z}$ are involved. This means that numerous hidden variables, $\mathbf{h}$, need to be used in its RNN surrogate to account for the homogenized history-dependent behavior [11]. When the number of hidden variables increases, the total number of learnable parameters in a RNN grows quadratically, and the training process of this RNN can become very slow. Thus, using RNNs that can be trained efficiently or have a good training performance is important for the applications in which a high dimension of hidden variables is required.

It has been shown that GRU is a favorable choice to design RNNs being able to track the long input history when predicting the output [18]. Actually, the GRU can be replaced by its reduced version, the Simplified Minimal Gated Recurrent Unit (SMGRU) [44], which has much less learnable variables compared to the original GRU with the same amount of hidden variables. All these recurrent units, including LSTM, are originally designed for Nature Language Processing (NLP) and they can be further simplified when used in the design of RNNs as surrogate models for history-dependent physical processes.

In this work, considering the self-consistency of RNNs' predictions according to the variation of the input increment size, random data augmentation process is introduced and applied on training data to improve the prediction self-consistency of the RNNs from the aspect of training. Besides, based on the SMGRU proposed in [44], and the linearization assumption in [40], three recurrent units are designed. The Simplified Minimal Recurrent Unit (SMRU) has the simplest architecture among the presented units and can be used to replace GRU in the design of RNNs. Although the RNN based on SMRU is not self-consistent with the variation of the input increment size, it is still considered in view of providing results for a comparative study. The other two units are Self-Consistency reinforced Minimal Recurrent Units, SC-MRU-T for the

one using Total strain as input, and SC-MRU-I for the one using strain Increment as input. The RNNs, based on either of these three units, are compared from the aspects of learnable variables numbers and training efficiency, and are also compared to the Self-Consistency reinforced LMSC suggested in [40]. For the RNNs using the two newly designed recurrent units, i.e. SC-MRU-T and SC-MRU-I, their prediction accuracy and self-consistency are demonstrated by comparing the RNNs predictions to the direct finite element simulations conducted at the RVE level, where various input increment sizes are tested on RNNs. Finally the three designed RNNs, respectively using the SMRU, SC-MRU-T and SC-MRU-I as recurrent unit, are used as surrogate of the meso-scale BVP during FE^2 multi-scale simulations of an open hole sample subjected to multiple loading cycles, in which the self-consistency of the RNNs using the SC-MRU-T and SC-MRU-I as recurrent unit is exemplified.

Organization of the paper is as follows. Section 2 recalls the main concept of homogenization, with an emphasis put on the notion of internal or state variables in multi-scale analyses. Recurrent units with reinforced self-consistency for RNNs acting as surrogates of homogenized response of heterogeneous micro-structures are presented and designed in Section 3, and their performance in terms of training and accuracy are assessed in Section 4 on a composite RVE. Their applicability and accuracy in the context of multi-scale simulations are then demonstrated in Section 5. In particular, for the multi-scale simulation, the size of the strain increment is modified by 3 orders of magnitude in order to illustrate the self-consistent nature of the developed SC-MRU-T and SC-MRU-I as recurrent units.

2. Computational homogenization-based multi-scale simulation

In this section, we briefly summarize the main lines of the multi-scale simulations based on the concept of computational homogenization. In that context, the homogenized response of the meso-scale problem serves as a history-dependent constitutive model of the heterogeneous materials for the macro-scale analysis. In all generalities, we consider a finite-strain setting.

2.1. Homogenization theory

We recall the Boundary Value Problems (BVPs) that are defined at the two scales as well as the basic relations of the scale transition theory.

2.1.1. The macro-scale BVP

At the macro-scale, the linear momentum balance equation of a body Ω_{M} reads, assuming no dynamical effects,

$$\mathbf{P}_{\mathrm{M}}(\boldsymbol{X}_{\mathrm{M}}) \cdot \boldsymbol{\nabla}_{\mathrm{M0}} + \boldsymbol{b}_{\mathrm{M}} = \mathbf{0} \quad \forall \boldsymbol{X}_{\mathrm{M}} \in \Omega_{\mathrm{M}} \,, \tag{1}$$

where the subscript "M" refers to the macroscopic values, $\mathbf{P}_{\mathrm{M}}$ is the first Piola-Kirchhoff stress tensor, $\boldsymbol{\nabla}_{\mathrm{M0}}$ is the gradient operator with respect to the macro-scale reference configuration, and $\boldsymbol{b}_{\mathrm{M}}$ is the load per unit reference volume. The boundary conditions read

$$\boldsymbol{u}_{\mathrm{M}}(\boldsymbol{X}_{\mathrm{M}}) = \bar{\boldsymbol{u}}_{\mathrm{M}} \qquad \forall \boldsymbol{X}_{\mathrm{M}} \in \partial_D \Omega_{\mathrm{M}} \,, \text{ and} \tag{2}$$

$$\mathbf{P}_{\mathrm{M}}(\boldsymbol{X}_{\mathrm{M}}) \cdot \boldsymbol{N}_{\mathrm{M}} = \bar{\boldsymbol{T}}_{\mathrm{M}} \qquad \forall \boldsymbol{X}_{\mathrm{M}} \in \partial_N \Omega_{\mathrm{M}} \,, \tag{3}$$

where $\bar{\boldsymbol{T}}_{\mathrm{M}}$ is the surface traction, per unit reference surface, on the Neumann boundary $\partial_N \Omega_{\mathrm{M}}$ of outward unit normal $\boldsymbol{N}_{\mathrm{M}}$ expressed in the reference configuration, and $\bar{\boldsymbol{u}}_{\mathrm{M}}$ is the constrained displacement $\boldsymbol{u}_{\mathrm{M}}$ on the Dirichlet boundary $\partial_D \Omega_{\mathrm{M}}$.

To complete the macro-scale BVP, a history-dependent constitutive model of the heterogeneous material is defined from the homogenized response of the micro-scale behavior. This homogenized response is defined as the relationship between the macro-scale deformation gradient $\mathbf{F}_{\mathrm{M}} = \mathbf{I} + \boldsymbol{u}_{\mathrm{M}} \otimes \boldsymbol{\nabla}_0$, with $\mathbf{I}$ the second-order identity tensor, and the macro-scale stress tensor $\mathbf{P}_{\mathrm{M}}$, which is expressed in the general form

$$\mathbf{P}_{\mathrm{M}}\left(\boldsymbol{X}_{\mathrm{M}},\, t\right) = \mathbf{P}_{\mathrm{M}}\left(\mathbf{F}_{\mathrm{M}}\left(\boldsymbol{X}_{\mathrm{M}},\, t\right);\, \boldsymbol{Z}_{\mathrm{M}}\left(\boldsymbol{X}_{\mathrm{M}},\, \tau\right),\, \tau \in [0,\, t]\right) \,, \tag{4}$$

where the set of internal or state variables $\boldsymbol{Z}_{\mathrm{M}}$ is used to follow history-dependent processes. In the context of homogenization-based multi-scale simulation, Eq. (4) is the mathematical representation of the meso-scale BVP via the scale transition.

2.1.2. The meso-scale BVP

The meso-scale BVP is usually defined on parallelepipedic or rectangular Representative Volume Elements (RVEs) Ω_{m}, with planar boundary faces $\partial\Omega_{\mathrm{m}}$. It is assumed that the time for a stress wave to propagate in the RVE remains negligible and that the classical continuum mechanics equations hold. In the absence of dynamical effects, the equilibrium equations at a material point $\boldsymbol{X}_{\mathrm{m}} \in \Omega_{\mathrm{m}}$ read

$$\mathbf{P}_{\mathrm{m}} \cdot \boldsymbol{\nabla}_{\mathrm{m0}} = \mathbf{0} \qquad \forall \boldsymbol{X}_{\mathrm{m}} \in \Omega_{\mathrm{m}}\,, \tag{5}$$

$$\mathbf{P}_{\mathrm{m}} \cdot \boldsymbol{N}_{\mathrm{m}} = \boldsymbol{T}_{\mathrm{m}} \qquad \forall \boldsymbol{X}_{\mathrm{m}} \in \partial\Omega_{\mathrm{m}}\,, \tag{6}$$

where the subscript "m" refers to the microscopic local value, $\mathbf{P}_{\mathrm{m}}$ is the first Piola-Kirchhoff stress tensor, $\boldsymbol{\nabla}_{\mathrm{m0}}$ is the gradient operator with respect to the micro-scale reference configuration, and $\boldsymbol{T}_{\mathrm{m}}$ is the surface traction, per unit reference surface, on the boundary $\partial\Omega_{\mathrm{m}}$ of outward unit normal $\boldsymbol{N}_{\mathrm{m}}$ expressed in the reference configuration.

To complete the micro-scale problem, the local constitutive laws of the different material phases are defined and, at a given time t and material point $\boldsymbol{X}_{\mathrm{m}}$, they write

$$\mathbf{P}_{\mathrm{m}}\left(\boldsymbol{X}_{\mathrm{m}},\, t\right) = \mathbf{P}_{\mathrm{m}}\left(\mathbf{F}_{\mathrm{m}}\left(\boldsymbol{X}_{\mathrm{m}},\, t\right);\, \boldsymbol{Z}_{\mathrm{m}}\left(\boldsymbol{X}_{\mathrm{m}},\, \tau\right),\, \tau \in [0,\, t]\right)\,, \tag{7}$$

where the micro-scale deformation gradient tensor $\mathbf{F}_{\mathrm{m}}(\boldsymbol{X}_{\mathrm{m}}) = \mathbf{I} + \boldsymbol{u}_{\mathrm{m}} \otimes \boldsymbol{\nabla}_{\mathrm{m0}}$ is evaluated in terms of the micro-scale displacement $\boldsymbol{u}_{\mathrm{m}}$, and where $\boldsymbol{Z}_{\mathrm{m}}$ is a set of internal variables defined to account for history-dependent processes.

2.1.3. The scale transition

In homogenization theories, the scale transition is defined through the volume averaging processes of the respective microscopic deformation gradient tensor $\mathbf{F}_{\mathrm{m}}(\boldsymbol{X}_{\mathrm{m}})$ and stress tensor $\mathbf{P}_{\mathrm{m}}(\boldsymbol{X}_{\mathrm{m}})$ over the meso-scale volume element Ω_{m}, which read

$$\mathbf{F}_{\mathrm{M}}\left(\boldsymbol{X}_{\mathrm{M}},\, t\right) = \frac{1}{V(\Omega_{\mathrm{m}})}\int_{\Omega_{\mathrm{m}}} \mathbf{F}_{\mathrm{m}}(\boldsymbol{X}_{\mathrm{m}},\, t)\mathrm{d}\boldsymbol{X}_{\mathrm{m}}\,, \text{ and} \tag{8}$$

$$\mathbf{P}_{\mathrm{M}}\left(\boldsymbol{X}_{\mathrm{M}},\, t\right) = \frac{1}{V(\Omega_{\mathrm{m}})}\int_{\Omega_{\mathrm{m}}} \mathbf{P}_{\mathrm{m}}(\boldsymbol{X}_{\mathrm{m}},\, t)\mathrm{d}\boldsymbol{X}_{\mathrm{m}}\,, \tag{9}$$

where $V(\Omega_{\mathrm{m}})$ is the volume of the RVE Ω_{m}. These averaging quantities are completed by the constraint of energy consistency between the different scales, which corresponds to the Hill-Mandel condition:

$$\mathbf{P}_{\mathrm{M}} : \delta\mathbf{F}_{\mathrm{M}} = \frac{1}{V(\Omega_{\mathrm{m}})}\int_{\Omega_{\mathrm{m}}} \mathbf{P}_{\mathrm{m}} : \delta\mathbf{F}_{\mathrm{m}}\mathrm{d}\boldsymbol{X}_{\mathrm{m}}\,. \tag{10}$$

2.2. Computational homogenization

In the context of FE^2 simulations, the virtual material law (4) is implicitly defined through the scale transition formalism presented in Section 2.1.3. In practice, the micro-scale displacement field $\boldsymbol{u}_{\mathrm{m}}(\boldsymbol{X}_{\mathrm{m}})$ is written under the form

$$\boldsymbol{u}_{\mathrm{m}}(\boldsymbol{X}_{\mathrm{m}}) = (\mathbf{F}_{\mathrm{M}} - \mathbf{I}) \cdot (\boldsymbol{X}_{\mathrm{m}} - \boldsymbol{X}_{\mathrm{m}_0}) + \boldsymbol{u}'(\boldsymbol{X}_{\mathrm{m}})\,, \tag{11}$$

where $\boldsymbol{X}_{\mathrm{m}_0}$ is a reference point of Ω_{m}, and $\boldsymbol{u}'(\boldsymbol{X}_{\mathrm{m}})$ is the perturbation field. From the relations (11), the Hill-Mandel condition (10) is rewritten

$$\mathbf{P}_{\mathrm{M}} : \delta\mathbf{F}_{\mathrm{M}} = \mathbf{P}_{\mathrm{M}} : \delta\mathbf{F}_{\mathrm{M}} + \frac{1}{V(\Omega_{\mathrm{m}})}\int_{\Omega_{\mathrm{m}}} \mathbf{P}_{\mathrm{m}} : (\delta\boldsymbol{u}' \otimes \boldsymbol{\nabla}_{\mathrm{m0}})\,\mathrm{d}\boldsymbol{X}_{\mathrm{m}}\,. \tag{12}$$

According to the Hill-Mandel condition (12) and the strain averaging relation (8), the perturbation field must thus satisfy the conditions,

$$\frac{1}{V(\Omega_{\mathrm{m}})}\int_{\Omega_{\mathrm{m}}} \mathbf{P}_{\mathrm{m}} : (\delta\boldsymbol{u}' \otimes \boldsymbol{\nabla}_{\mathrm{m0}})\,\mathrm{d}\boldsymbol{X}_{\mathrm{m}} = 0\,, \tag{13}$$

and

$$\frac{1}{V(\Omega_{\mathrm{m}})}\int_{\Omega_{\mathrm{m}}}\boldsymbol{u}'(\boldsymbol{X}_{\mathrm{m}})\otimes\boldsymbol{\nabla}_{\mathrm{m}0}\mathrm{d}\boldsymbol{X}_{\mathrm{m}}=\frac{1}{V(\Omega_{\mathrm{m}})}\int_{\partial\Omega_{\mathrm{m}}}\boldsymbol{u}'\otimes\boldsymbol{N}_{\mathrm{m}}\mathrm{d}\boldsymbol{X}_{\mathrm{m}}=\mathbf{0}\,. \tag{14}$$

On the one hand, since the Hill-Mandel condition (13) corresponds to the weak form of the strong form (5-6) of the meso-scale BVP [45, 46], the virtual macroscopic material law (4) corresponds to the meso-scale BVP (13) resolution in the context of a concurrent multi-scale finite-element resolution. On the other hand, the condition (14) is satisfied by constraining specific boundary conditions on the RVE, such as such as Kinematic Uniform Boundary Conditions (KUBCs), Periodic Boundary Conditions (PBCs), Static Uniform Boundary Conditions (SUBCs), *etc.* We refer to [47] for a detailed implementation procedure.

The computational homogenization method is thus seen as a complex history-dependent constitutive representation of a heterogeneous material defined by the meso-scale responses $\mathbf{P}_{\mathrm{M}}$ and material operator $\mathbb{C}_{\mathrm{M}}=\frac{\partial\mathbf{P}_{\mathrm{M}}}{\partial\mathbf{F}_{\mathrm{M}}}$ of a given RVE subjected to a given deformation gradient $\mathbf{F}_{\mathrm{M}}$. In a FE^2 analysis, the meso-scale response is obtained through the finite element discretization of the weak form (13). In that context, the internal or state variables defining Eq. (4), are the set of internal or state variables of the RVE, i.e.

$$\boldsymbol{Z}_{\mathrm{M}}(\boldsymbol{X}_{\mathrm{M}},\,t)=\{\boldsymbol{Z}_{\mathrm{m}}(\boldsymbol{X}_{\mathrm{m}},\,t)\,:\,x\in\Omega_{\mathrm{m}}(\boldsymbol{X}_{\mathrm{M}})\}\,, \tag{15}$$

where the operator $\{\bullet\}$ is used to represent a set. Since the resolution of Eq. (13) is an iterative process that needs to be carried out at each Gauss point of the macro-scale finite element discretization, and since this iterative process has to be repeated for each macro-scale iteration, this implies a high computational cost.

3. Recurrent Neural Networks (RNNs) with self-consistent minimal recurrent unit

RNNs can be used as surrogates of computationally expensive history-dependent constitutive material models in general, and of the homogenized response (9) of the meso-scale BVP (13) in the particular context of a multi-scale analysis. Although, as detailed in the introduction, LSTM unit and GRU are now commonly used as a surrogate of history-dependent behavior of material models, they might suffer from inaccuracy when the strain increment is reduced by several orders. This so-called lack of self-consistency of RNNs has been studied in the work [40], in which the authors designed a Self-Consistency reinforced Linearized Minimal State Cell (SC-LMSC) as a recurrent unit serving as surrogate of a J2-isotropic elasto-plastic material model. In their work, the case studies show that a few transition layers (typically, ≥ 2) are required in order to obtain an optimal training time and accuracy of the designed RNN. Since the number of state variables or hidden variables in a RNN is associated with that of the physical internal variables [11], the number of hidden variables in the RNN will be high when the original physical model is characterized by a large amount of physical internal or state variables, which is the case in the context of computational homogenization, since for the meso-scale BVP described in Section 2.1.2 the physical internal or state variables correspond to the ones defined at the different integration points of the RVE following Eq. (15). The resulting high dimension of the hidden variables, together with the use of several transition layers, will lead to an increased number of learnable parameters in the RNN and therefore to a slow training process. Although the number of learnable parameters involved here is nothing compared to that of the current popular AI networks, we would like to have a RNN which can be trained efficiently with sequential data on a personal computer or a server. We thus aim at finding an architecture of RNN which can reduce or use comparable amount of learnable variables as that of SC-LMSC while reaching a better training performance.

In this section, we use respectively $\mathbf{u}_t$ and $\mathbf{v}_t$ $(t=0,1,...)$ to represent the sequential input and output of the RNNs, and $\underline{\mathbf{u}}_t$ and $\underline{\mathbf{v}}_t$ respectively represent their normalized values, when normalization is applied. Thus, the incremental form of the input reads $\Delta\mathbf{u}_t=\mathbf{u}_t-\mathbf{u}_{t-1}$ with $\Delta\mathbf{u}_0=\mathbf{u}_0$, and $\Delta\underline{\mathbf{u}}_t=\underline{\mathbf{u}}_t-\underline{\mathbf{u}}_{t-1}$ with $\Delta\underline{\mathbf{u}}_0=\underline{\mathbf{u}}_0$ after normalization. For a recurrent unit, $\mathbf{x}_t$ and $\mathbf{y}_t$ are used to represent its input and output, respectively.

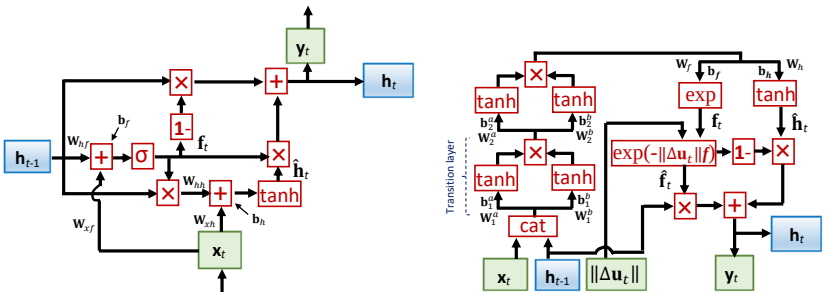


Figure 2: Two recurrent units: (a) Minimal Gated Recurrent Unit (MGRU); and (b) Self-Consistency reinforced Linearized Minimal State Cell (SC-LMSC).

3.1. Recurrent Units

3.1.1. Minimal Gated Recurrent Unit (MGRU)

In our previous work [18], it was found that GRU has a better performance than long short-term memory network (LSTM) when the RNN is designed to replace a history-dependent constitutive material model. LSTM uses two groups of hidden variables to keep the historical information, while GRU has only one group of hidden variables. It not only makes the structure of GRU conciser, but also closer to the way of passing historical information in physical models than LSTM. However, both LSTM and GRU are originally designed for the purpose of Natural Language Processing (NLP). Thus, some special information paths, which are designed for NLP, may not be necessary for the history-dependent non-linear mapping. It has been shown that, by removing some information paths, a simplified GRU, the Minimal Gated Recurrent Unit (MGRU) [48], can be used in the design of RNNs. The MGRU has the architecture and simple information paths detailed in Fig. 2(a), where the operation symbols are

- $\boxed{+}$: The element-wise sum operator on two vectors of same dimension, which can be expressed as $\mathbf{r} = \mathbf{a} + \mathbf{b}$;
- $\boxed{\times}$: The element-wise multiplication $\odot$, or Hadamard product, on two vectors, $\mathbf{a}$ and $\mathbf{b}$, of same dimension, which reads $r_i = a_i b_i$, no sum on i intended;
- $\boxed{\mathbf{1}-}$: The element-wise operator on a vector $\mathbf{a}$, which reads $\mathbf{r} = \mathbf{1} - \mathbf{a}$;
- $\boxed{\sigma}$: The element-wise non-linear activation sigmoid function, $\sigma(a) = \frac{1}{1+\exp(-a)}$, which returns values in the range 0 to 1; and
- $\boxed{\tanh}$: The element-wise non-linear activation hyperbolic tangent function.

In a RNN, a feed-forward neural network is usually applied to transit the original input, $\mathbf{u}_t$ (or $\underline{\mathbf{u}}_t$) to the input of the MGRU, $\mathbf{x}_t$. The information flow in the MGRU follows Fig. 2(a) with

$$\mathbf{f}_t = \sigma(\mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{xf}\mathbf{x}_t + \mathbf{b}_f)\,; \tag{16}$$

$$\hat{\mathbf{h}}_t = \tanh(\mathbf{W}_{hh}(\mathbf{f}_t \odot \mathbf{h}_{t-1}) + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_h)\,; \text{ and} \tag{17}$$

$$\mathbf{h}_t = (\mathbf{1} - \mathbf{f}_t) \odot \mathbf{h}_{t-1} + \mathbf{f}_t \odot \hat{\mathbf{h}}_t\,; \tag{18}$$

where $\mathbf{h}_{t-1}$, $\mathbf{h}_t$ and $\hat{\mathbf{h}}_t$ are respectively the previous, current and update hidden variables, $\mathbf{x}_t$ is the current input, "$\odot$" is the element-wise multiplication operator, and where the learnable parameters $\mathbf{W}$ and $\mathbf{b}$ are respectively the weights and biases of the linear operations. The output $\mathbf{y}_t$ is equivalent to the current hidden variables $\mathbf{h}_t$. Eventually, a feed-forward neural network is applied to transit $\mathbf{y}_t$ to the output of the RNN $\mathbf{v}_t$ (or $\underline{\mathbf{v}}_t$).

3.1.2. Self-Consistency reinforced Linearized Minimal State Cell (SC-LMSC)

The structure of the SC-LMSC proposed in the work [40] is illustrated in Fig. 2(b). In order to include the information of the increment size, the input variables are given in an incremental form, and are then converted into their direction, $\mathbf{x}_t = \Delta\mathbf{u}_t/\|\Delta\mathbf{u}_t\|$, and their Frobenius norm, $\|\Delta\mathbf{u}_t\|$. The additional operation symbols used in Fig. 2(b) are

- $\boxed{\text{cat}}$: The concatenate operation on two vectors; and
- $\boxed{\text{exp}}$: The element-wise exponential function.

The example of two non-linear transition layers, $N = 2$, is presented in Fig. 2(b), in which the information flow in layer "i" $(i = 1, ...N)$ is expressed as

$$\mathbf{O}_i = \tanh(\mathbf{W}_i^a \mathbf{I}_i + \mathbf{b}_i^a) \odot \tanh(\mathbf{W}_i^b \mathbf{I}_t + \mathbf{b}_i^b)\,, \tag{19}$$

where $\mathbf{I}_i$ and $\mathbf{O}_i$ are respectively the input and output of that layer. Afterwards, the remaining information flow reads,

$$\mathbf{f}_t = \exp(\mathbf{W}_f \mathbf{O}_N + \mathbf{b}_f)\,; \tag{20}$$

$$\hat{\mathbf{f}}_t = \exp(-\|\Delta\mathbf{u}_t\|\mathbf{f}_t)\,; \tag{21}$$

$$\hat{\mathbf{h}}_t = \tanh(\mathbf{W}_h \mathbf{O}_N + \mathbf{b}_h)\,; \text{ and} \tag{22}$$

$$\mathbf{h}_t = \hat{\mathbf{f}}_t \odot \mathbf{h}_{t-1} + (\mathbf{1} - \hat{\mathbf{f}}_t) \odot \hat{\mathbf{h}}_t\,. \tag{23}$$

The term $\exp(-\|\Delta\mathbf{u}_t\|\mathbf{f}_t)$ in Eq. (21) is obtained by solving an approximated linear differential equation in a small input step [40]. The exponential function $\exp(\bullet)$ is used as the non-linear activation function in Eq. (20), to ensure that $\mathbf{f}_t$ is positive. Therefore, the yield $\hat{\mathbf{f}}_t$ is in a range 0 to 1. The output $\mathbf{y}_t$, which is equivalent to $\mathbf{h}_t$, is transited to $\mathbf{v}_t$ (or $\underline{\mathbf{v}}_t$) through a feed-forward neural network.

3.1.3. The characteristics of the Recurrent Units

Comparing the two kinds of recurrent units presented in Sections 3.1.1 and 3.1.2, it can be seen that their information flow has two main branches: one branch determines the ratio vectors, respectively $\mathbf{f}_t$ for the MGRU and $\hat{\mathbf{f}}_t$ for SC-LMSC; and the other branch evaluates the update hidden variables $\hat{\mathbf{h}}_t$.

For both recurrent units, their ratio vectors are the outputs of the activation functions designed in order to ensure that their element values are in the range from 0 to 1. The current hidden variables, $\mathbf{h}_t$, are computed through element-wise linear combination of previous and update hidden variables, respectively $\mathbf{h}_{t-1}$ and $\hat{\mathbf{h}}_t$, with their ratio vectors being respectively $\mathbf{1} - \mathbf{f}_t$ and $\mathbf{f}_t$ for the MGRU, and $\hat{\mathbf{f}}_t$ and $\mathbf{1} - \hat{\mathbf{f}}_t$ for the SC-LMSC, see respectively Eqs. (18) and (23). For the MGRU, a sigmoid function, $\sigma(\bullet)$ is adopted in Eq. (16) to compute $\mathbf{f}_t$, while for the SC-LMSC the function $\exp(-\exp(\bullet))$ is used to compute $\hat{\mathbf{f}}_t$ –if we ignore the norm term $\|\Delta\mathbf{u}_t\|$– in Eq (21). Actually, the performance of the MGRU will not be affected by replacing the $\sigma(\bullet)$ term by $1.0 - \exp(-\exp(\bullet))$ in Eq. (16), since the response of these two functions is quite similar.

Comparing Eq. (17) and Eq. (22), it appears that the two recurrent units used different expressions for the update hidden variables, $\hat{\mathbf{h}}_t$. Actually, omitting the element-wise multiplication with $\mathbf{f}_t$ and using only $\mathbf{h}_{t-1}$ in Eq. (17) would not affect the performance of the MGRU for our history-dependent non-linear problem, see the training performance comparison provided in Appendix A.

Besides, in order to increase the flexibility of the RNNs so that it can be adapted to the complexity of the physical problem being studied, i.e. the simulation of non-linear RVEs, we point out that

- The non-linear transition is usually applied as a preliminary information process;
- The original input variables should pass though a feed-forward neural network (Fw) before being sent to the MGRU; and
- Non-linear transition layers are applied on both the input and hidden variables in the SC-LMSC, see Fig. 2(b).

3.2. Design of the Self Consistent Minimal Recurrent Units (SC-MRU)

In this section, a Simplified Minimal Recurrent Unit (SMRU), with the current total value, $\underline{\mathbf{u}}_t$, and the norm of input increment, $\|\Delta\underline{\mathbf{u}}_t\|$, as input, is firstly presented. The self-consistency of this SMRU will be weakly reinforced by using $\|\Delta\underline{\mathbf{u}}_t\|$ and a dedicated training process. Based on this SMRU, a self-consistency reinforced recurrent unit is presented. Since this unit takes the previous total value $\underline{\mathbf{u}}_{t-1}$ and the incremental value $\Delta\underline{\mathbf{u}}_t$ as input, it is abbreviated as SC-MRU-T. Another self consistency reinforced recurrent unit, which only uses the incremental value $\Delta\mathbf{u}_t$ as input, is also designed and abbreviated as SC-MRU-I.

In order to reinforce the consistency of the RNN predictions for various increment sizes of the input, in the two self-consistent units, the double exponential function, see Eqs. (20) and (21) of the SC-LMSC [40], is adopted with as additional input the norm of the input increment. The training and prediction performance of the presented SMRU, SC-MRU-T and SC-MRU-I will be compared in Section 4 on RVE responses and in Section 5 on multi-scale simulations.

3.2.1. SMRU

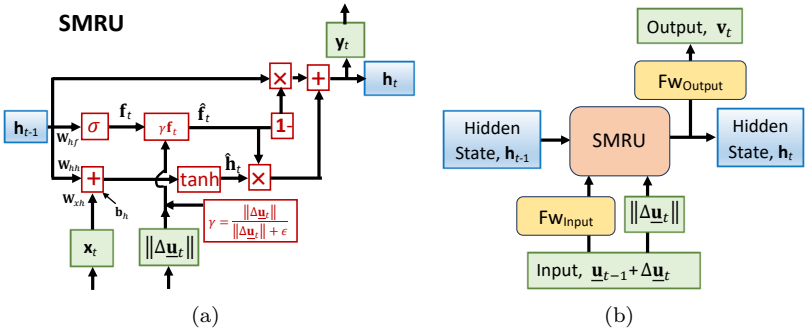


Figure 3: Design of the (a) Simplified Minimal Recurrent Unit (SMRU); and (b) Simplified RNN using as recurrent unit the SMRU and with the sum of the previous total value $\underline{\mathbf{u}}_{t-1}$ and of the incremental value $\Delta\underline{\mathbf{u}}_t$ as input (S-RNN-T).

In the work [44], three simplified versions of the MGRU were proposed by deleting either the term in $\mathbf{b}_f$ in Eq. (16), the terms in $\mathbf{x}_t$ and $\mathbf{b}_f$ in Eq. (16), or the terms in $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$ in Eq. (16), respectively. The performance of the second simplified version was found to be superior not only to that of the other two simplified versions, but also to that of the original MGRU. Besides, following the outcomes of the comparison study provided in Appendix A, we proposed a Simplified Minimal Recurrent Unit (SMRU), whose architecture is illustrated in Fig. 3(a) and whose corresponding information flow reads

$$\mathbf{f}_t = \sigma(\mathbf{W}_{hf}\mathbf{h}_{t-1})\,; \tag{24}$$

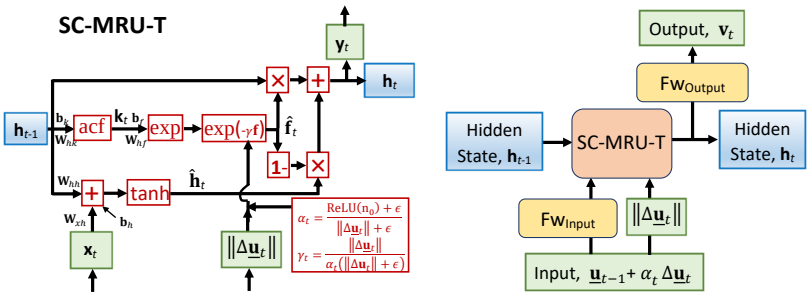
$$\hat{\mathbf{f}}_t = \frac{\|\Delta\underline{\mathbf{u}}_t\|}{\|\Delta\underline{\mathbf{u}}_t\| + \epsilon}\mathbf{f}_t\,; \tag{25}$$

$$\hat{\mathbf{h}}_t = \tanh(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h)\,; \text{ and} \tag{26}$$

$$\mathbf{h}_t = (1.0 - \hat{\mathbf{f}}_t) \odot \mathbf{h}_{t-1} + \hat{\mathbf{f}}_t \odot \hat{\mathbf{h}}_t\,. \tag{27}$$

In Eq. (25), ϵ is a small value, such as $1.0\,\mathrm{e}-20$, which is introduced in order to guarantee that the hidden variables will not be updated when the input stays same as in the previous step.

The corresponding architecture of the RNN is presented in Fig. 3(b). A feed-forward neural network of one layer, $\boxed{\mathrm{Fw}_{\mathrm{Input}}}$, is applied to lift the dimension of the input to that of the hidden variables, while the desired output is extracted from the hidden variables through a one-layer feed-forward neural network without activation function, $\boxed{\mathrm{Fw}_{\mathrm{Output}}}$.



(a) (b)

Figure 4: Design of the (a) Self-Consistent Minimal Recurrent Unit with the previous total value $\underline{\mathbf{u}}_{t-1}$ and the incremental value $\Delta\underline{\mathbf{u}}_t$ as input (SC-MRU-T); and (b) Self-Consistent RNN using as recurrent unit the SC-MRU-T and with the previous total value $\underline{\mathbf{u}}_{t-1}$ and the incremental value $\Delta\underline{\mathbf{u}}_t$ as input (SC-RNN-T).

3.2.2. SC-MRU-T

A self-consistent unit is designed based on the presented SMRU, as illustrated in Fig. 4(a). Besides replacing the sigmoid activation function by a double exponential function, a few modifications are applied to achieve a good performance of the recurrent unit. This SC-MRU-T takes both previous total value $\underline{\mathbf{u}}_{t-1}$ and incremental value $\Delta\underline{\mathbf{u}}_t$ as input. Totward this end, the RNN using as recurrent unit this SC-MRU-T, and as illustrated in Fig. 4(b), applies a feed-forward neural network of one layer, $\boxed{\text{Fw}_{\text{Input}}}$, to lift the dimension of the inputs to that of the hidden variables, yielding the SC-MRU-T input $\mathbf{x}_t$, following

$$\underline{\hat{\mathbf{u}}}_t = \underline{\mathbf{u}}_{t-1} + \alpha_t \Delta\underline{\mathbf{u}}_t\,, \text{ with } \alpha_t = \frac{\text{ReLU}(\mathsf{n}_0)+\epsilon}{\|\Delta\underline{\mathbf{u}}_t\|+\epsilon}\,; \tag{28}$$

$$\mathbf{x}_t = \text{acf}\left(\mathbf{W}_{ux}\underline{\hat{\mathbf{u}}}_t + \mathbf{b}_x\right), \tag{29}$$

where ReLU() refers to the "relu" activation function, "acf" is the other chosen activation function in the feed-forward network $\boxed{\text{Fw}_{\text{Input}}}$, and n_0 is a scalar learnable parameter. The information flow in the SC-MRU-T reads

$$\mathbf{f}_t = \exp(\mathbf{W}_{hf}\mathbf{k}_t + \mathbf{b}_f) \text{ with } \mathbf{k}_t = \text{acf}\left(\mathbf{W}_{hk}\mathbf{h}_{t-1} + \mathbf{b}_k\right); \tag{30}$$

$$\hat{\mathbf{f}}_t = \exp\left(-\frac{\|\Delta\underline{\mathbf{u}}_t\|}{\alpha_t(\|\Delta\underline{\mathbf{u}}_t\|+\epsilon)}\mathbf{f}_t\right)\,; \tag{31}$$

$$\hat{\mathbf{h}}_t = \tanh\left(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h\right)\,; \text{ and} \tag{32}$$

$$\mathbf{h}_t = \hat{\mathbf{f}}_t \odot \mathbf{h}_{t-1} + \left(1-\hat{\mathbf{f}}_t\right)\odot\hat{\mathbf{h}}_t\,. \tag{33}$$

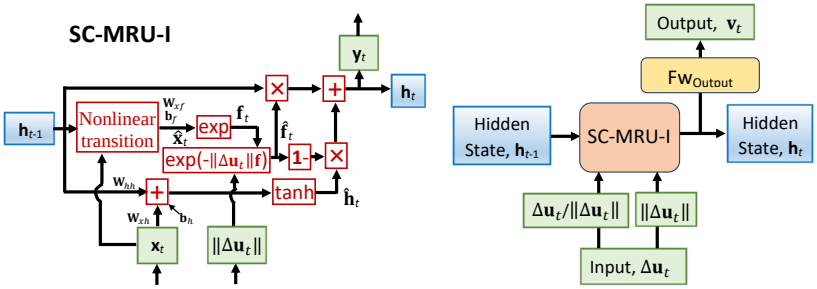
Eventually, the desired output of the recurrent neural network, see Fig. 4(b), is extracted from the hidden variables through a one-layer feed-forward neural network without activation function, $\boxed{\text{Fw}_{\text{Output}}}$.

3.2.3. SC-MRU-I

In this second self-consistent case, the incremental form of the input is considered, as in the SC-LMSC unit [40]: the incremental value $\Delta\mathbf{u}_t$ and its norm $\|\Delta\mathbf{u}_t\|$ are thus used as input.

In the previous Section 3.1, it was shown that there are two basic elements in the recurrent MGRU (respectively the recurrent SC-LMSC unit): the calculations of the ratio vector $\mathbf{f}_t$ (respectively $\hat{\mathbf{f}}_t$) and the use of the update hidden variables, $\hat{\mathbf{h}}_t$. Besides, in the RNNs, a preliminary non-linear transition is applied on the original input and/or previous hidden variable, $\mathbf{h}_{t-1}$, before the information flows into the two branches to calculate $\mathbf{f}_t$ (respectively $\hat{\mathbf{f}}_t$) and $\hat{\mathbf{h}}_t$. This means that these two information flow branches share the same input.

After performing some comparative training experiments, it was noticed that training the RNNs is slow when $\Delta\mathbf{u}_t$ is chosen as an input. However, the good performance of the simplified version of MGRU [44] suggests that better RNN performance could be achieved by using different input information when calculating the ratio vector $\mathbf{f}_t$ (respectively $\hat{\mathbf{f}}_t$) and update hidden variables, $\hat{\mathbf{h}}_t$. In other words, using separate non-linear transition layers for the preliminary information process of vector $\mathbf{f}_t$ (respectively $\hat{\mathbf{f}}_t$) and $\hat{\mathbf{h}}_t$ is beneficial in practice.



(a) (b)

Figure 5: Design of the (a) Self-Consistency Reinforced Recurrent Unit with the incremental value $\Delta\underline{\mathbf{u}}_t$ as input (SC-MRU-I); and (b) Self-Consistent RNN using as recurrent unit the SC-MRU-I and with the incremental value $\Delta\underline{\mathbf{u}}_t$ as input (SC-RNN-I).

Keeping in mind that we want to use as few learnable variables as possible, we suggest a Self-Consistent Minimum Recurrent Unit using Incremental input (SC-MRU-I) based on different architectural designs and training attempts. The general architecture of the SC-MRU-I is presented in Fig. 5(a). For this recurrent unit, the inputs $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$ are used directly to calculate the update variable, $\hat{\boldsymbol{h}}_t$. Then, a non-linear transition block is applied on $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$ before calculating the ratio vector $\hat{\mathbf{f}}_t$. As a result, the information flow in the SC-MRU-I reads

$$\mathbf{x}_t, \mathbf{h}_{t-1} \rightarrow \boxed{\text{Non-linear transition}} \rightarrow \hat{\mathbf{x}}_t\,; \tag{34}$$

$$\mathbf{f}_t = \exp(\mathbf{W}_{xf}\hat{\mathbf{x}}_t + \mathbf{b}_f)\,; \tag{35}$$

$$\hat{\mathbf{f}}_t = \exp(-\|\Delta\mathbf{u}_t\|\mathbf{f}_t)\,; \tag{36}$$

$$\hat{\mathbf{h}}_t = \tanh(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_h)\,; \text{ and} \tag{37}$$

$$\mathbf{h}_t = \hat{\mathbf{f}}_t \odot \mathbf{h}_{t-1} + (\mathbf{1} - \hat{\mathbf{f}}_t) \odot \hat{\mathbf{h}}_t\,. \tag{38}$$

The input $\mathbf{x}_t$ of this SC-MRU-I is obtained from the incremental value following $\mathbf{x}_t = \Delta\mathbf{u}_t/\|\Delta\mathbf{u}_t\|$, as illustrated in the RNN using as recurrent unit the designed SC-MRU-I, see Fig. 5(b). In this RNN, the output is also extracted from the hidden variables through a one-layer feed-forward neural network without activation function, $\boxed{\text{Fw}_{\text{Output}}}$. In this architecture, the input does not need to be normalized, contrarily to the ones presented in Sections 3.2.1 and 3.2.2, since they are divided by the vector norm before feeding the SC-MRU-I.

3.2.4. Discussion on self-consistency

From Eqs. (31-32) and (36-37), it can be seen that the self-consistency of the RNNs with respect to various input step sizes is reinforced by the following common characteristics of SC-MRU-T and SC-MRU-I.

- The update hidden variable, $\hat{\mathbf{h}}_t$, is evaluated based on the previous hidden variables $\mathbf{h}_{t-1}$ and on the direction of the current input whose step size information has been removed by normalization using $\|\Delta\mathbf{u}_t\|$, see respectively Eqs. (28) and (37).

- The ratio vector $\hat{\mathbf{f}}_t$ is computed by an exponential function with the information of the current input step size, $\|\Delta\mathbf{u}_t\|$, see respectively Eqs. (31) and (36). For a small input step size, e.g. $< 10^{-3}$, which is

commonly used in nonlinear finite element analyses to ensure numerical convergence, the exponential function is really close to a linear function. Thus, the current hidden variable $\mathbf{h}_t$ is computed by element-wise linear interpolation between the previous and update hidden variables according to the input step size, see Eqs. (33) and (38).

This interpolation between the previous and update hidden variables according to the input step size is not guaranteed in recurrent units which use only a sigmoid function to compute the ratio vector $\mathbf{f}_t$, such as in the GRU.

3.2.5. The non-linear transition block of the SC-MRU-I

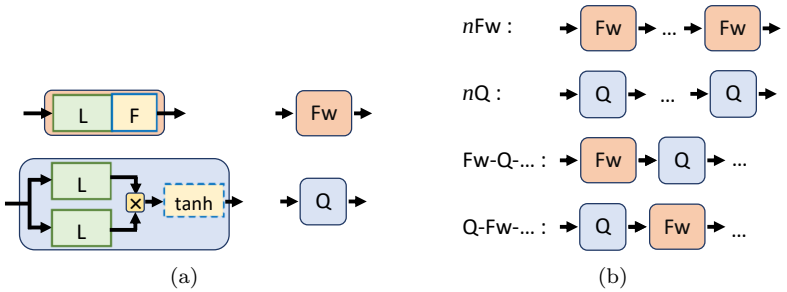


Figure 6: Non-linear transition block: (a) Basic operation units; and (b) Possible structures of the non-linear transition block.

The non-linear transition block is constructed using artificial neural layers. An artificial neural layer performs a linear operation on the input, and an activation function is then optionally applied on the result. To integrate two linear operations into one, the concatenation of the previous hidden variables and of the current input, $[\mathbf{h}_{t-1}, \mathbf{x}_t]$, is used as the input of this block. For simplicity, we represent

- The linear operation by $\boxed{\text{L}}$, standing for $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$, where $\mathbf{x}$, $\mathbf{y}$, $\mathbf{W}$ and $\mathbf{b}$ are respectively the input, output, weight and bias of the layer; and
- The adopted activation function by $\boxed{\text{F}}$.

In Fig. 6(a), two basic units are defined:

- $\boxed{\text{Fw}}$ is a feed-forward artificial neural layer which includes the linear operation and a non-linear active function; and
- $\boxed{\text{Q}}$ represents the quadratic operation, which has two linear operations followed by an element-wise multiplication and a hyperbolic tangent function being applied on the result.

Various non-linear transition blocks can then be constructed by connecting these basic units in different serial orders. In Fig. 6(b), "nFw" represents n layers of feed-forward neural network, "nQ" stands for n quadratic operations applied in a serial order, "Fw-Q-..." stands for a feed-forward neural layer followed by a quadratic operation etc., and "Q-Fw-..." stands for a quadratic operation followed by a feed-forward neural layer etc.

In a series of feed-forward neural networks, such as in "nFw" structure, the activation function $\boxed{\text{F}}$ can be either "ReLU", "LeakyReLU", "tanh" or "σ", etc. However, in a structure such as "nQ", a bounded activation function, such as "tanh" in Fig. 6(a), is required in order to avoid the problem of "Exploding Gradient" during the training. Nevertheless, when a single $\boxed{\text{Q}}$ is considered, the "tanh" function in the dashed box of Fig. 6(a) can be omitted without introducing that issue in training; the training performance of the RNN will not be affected either.

To simplify the design of the non-linear transition block, we consider that all its linear operations have the same output dimension, $\hbar$, which is also the dimension of the hidden variables.

3.2.6. Implementation of the RNNs

The presented recurrent units are not basic modules of the PyTorch library. Because of their recurrent nature, the RNN traning process would be rather slow if it were implemented in Python. Therefore, PyTorch C++ frontend, LibTorch [49], which is a pure C++ interface to the PyTorch machine learning framework, is used for the implementation of the designed RNNs.

All the detailed implementation is available in [50]. The basic training operations of the NNWs, such as computation and optimization of the loss function and update of the weights, follow the recommendations provided by PyTorch library [49] and are thus omitted in this paper.

Before comparing the training performance of RNNs using different non-linear transition blocks and numbers of hidden variables, the preparation of training and testing data needs to be clarified.

3.3. The training and testing data preparation for the RNN

Following Section 2.1.3, the homogenized stress tensor $\mathbf{P}_{\mathrm{M}}$ corresponding to a given macroscopic deformation gradients $\mathbf{F}_{\mathrm{M}}$ sequence, is required for solving the macroscopic BVP. We here detail the frame invariant input and output of the RNN. Then we summarize the data generation strategy presented in [18]. Eventually we present a process of random training data augmentation.

3.3.1. Input and output variables of the RNN

Since the resolution of the meso-scale BVP respects the frame invariance, the Green-Lagrange strain tensor $\mathbf{E}_{\mathrm{M}}$ and the 2$^{\mathrm{nd}}$ Piola-Kirchhoff stress tensor $\mathbf{S}_{\mathrm{M}}$ are used to describe the stress-strain relationship, instead of using directly $\mathbf{F}_{\mathrm{M}}$ and $\mathbf{P}_{\mathrm{M}}$. This corresponds to eliminating the rigid rotation mode, with the following conversions

$$\mathbf{E}_{\mathrm{M}} = \frac{1}{2}\left({\mathbf{F}_{\mathrm{M}}}^{\mathrm{T}} \cdot \mathbf{F}_{\mathrm{M}} - \mathbf{I}\right), \text{ and} \tag{39}$$

$$\mathbf{P}_{\mathrm{M}} = \mathbf{F}_{\mathrm{M}} \cdot \mathbf{S}_{\mathrm{M}}\,. \tag{40}$$

In particular, for the 2D RVE under plane strain condition, at step t, the output variable of the RNN is $\mathbf{v}_t = \{S_{\mathrm{M}XX}, S_{\mathrm{M}YY}, S_{\mathrm{M}ZZ}, S_{\mathrm{M}XY}\}_t$, while the inputs, for the RNNs built with respectively the SMRU, SC-MRU-T and SC-MRU-I, are $\mathbf{u}_t$, $\{\mathbf{u}_{t-1}, \Delta\mathbf{u}_t\}$ and $\Delta\mathbf{u}_t$, where

$$\mathbf{u}_t = \{E_{\mathrm{M}XX}, E_{\mathrm{M}YY}, E_{\mathrm{M}XY}\}_t\,;\ \mathbf{u}_{t-1} = \{E_{\mathrm{M}XX}, E_{\mathrm{M}YY}, E_{\mathrm{M}XY}\}_{t-1}\,;\ \text{and} \tag{41}$$

$$\begin{aligned}\Delta\mathbf{u}_t &= \{\Delta E_{\mathrm{M}XX}, \Delta E_{\mathrm{M}YY}, \Delta E_{\mathrm{M}XY}\}_t\,;\\ &= \{E_{\mathrm{M}XX}, E_{\mathrm{M}YY}, E_{\mathrm{M}XY}\}_t - \{E_{\mathrm{M}XX}, E_{\mathrm{M}YY}, E_{\mathrm{M}XY}\}_{t-1}\,.\end{aligned} \tag{42}$$

3.3.2. Data collection of the meso-scale BVP simulations

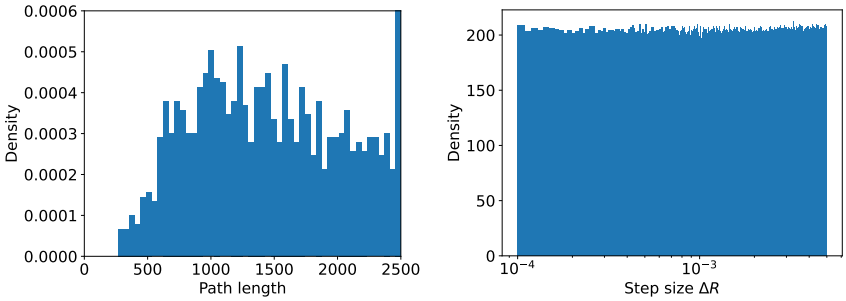
The training and testing data are provided by performing several computational homogenizations on the same RVE. $\mathbf{F}_{\mathrm{M}}$ is used to define the boundary condition applied on the fluctuation field $\boldsymbol{u}'$, see Eq. (11), on the RVE boundary. Since $\mathbf{F}_{\mathrm{M}}$ has a unique decomposition, $\mathbf{R}_{\mathrm{M}} \cdot \mathbf{U}_{\mathrm{M}}$, where $\mathbf{R}_{\mathrm{M}}$ is a rotation tensor, because of the frame invariance of the meso-scale BVP, this rotation can be defined arbitrarily and $\mathbf{R}_{\mathrm{M}} = \mathbf{I}$ is used.

Thus, the preparation of data for training and testing of the RNN is based on generating sequential $\mathbf{U}_{\mathrm{M}}$, as detailed in Section 3.3.3. The meso-scale BVP is defined according to $\tilde{\mathbf{F}}_{\mathrm{M}} = \mathbf{U}_{\mathrm{M}}$, while the resolution provides $\tilde{\mathbf{P}}_{\mathrm{M}} = \mathbf{U}_{\mathrm{M}} \cdot \mathbf{S}_{\mathrm{M}}$, where the notation $\tilde{\bullet}$ is used when removing the rotation tensor. Then, the RNN input and output are extracted through

$$\mathbf{E}_{\mathrm{M}} = \frac{1}{2}\left(\mathbf{U}_{\mathrm{M}}^2 - \mathbf{I}\right) \quad \text{and} \quad \mathbf{S}_{\mathrm{M}} = \mathbf{U}_{\mathrm{M}}^{-1} \cdot \tilde{\mathbf{P}}_{\mathrm{M}}\,. \tag{43}$$

3.3.3. Random and cyclic loading path

Since RNNs require sequential data for training, sequences of $\mathbf{U}_{\mathrm{M}}$ are generated. A random loading path $\{\mathbf{U}_{\mathrm{M}_t}\}$ is defined by a sequence of right stretch tensor, such as $\mathbf{U}_{\mathrm{M}_0}$, $\mathbf{U}_{\mathrm{M}_1}$, ..., $\mathbf{U}_{\mathrm{M}_N}$, where $\mathbf{U}_{\mathrm{M}_0} = \mathbf{I}$ is the starting stage of the loading path. The loading increment, $\Delta\mathbf{U}_{\mathrm{M}_t} = \mathbf{U}_{\mathrm{M}_t} - \mathbf{U}_{\mathrm{M}_{t-1}}$, is defined as a random vector which permits the loading path to change the loading direction at each step [18].



(a) (b)

Figure 7: Statistical information of 2000 generated random loading paths: (a) Distribution of the number of the increments in each path –ordinates are truncated to show the distribution of paths not reaching the bound of 2500 increments; and (b) Distribution of the increment ΔR.

The increment of the right stretch tensor, $\Delta\mathbf{U}_{\mathrm{M}}$ is a symmetric second-order tensor whose spectral decomposition form is expressed as,

$$\Delta\mathbf{U}_{\mathrm{M}} = \Delta\lambda_1 \boldsymbol{n}_1 \otimes \boldsymbol{n}_1 + \Delta\lambda_2 \boldsymbol{n}_2 \otimes \boldsymbol{n}_2 + \Delta\lambda_3 \boldsymbol{n}_3 \otimes \boldsymbol{n}_3 \,, \tag{44}$$

where $\boldsymbol{n}_1$, $\boldsymbol{n}_2$, and $\boldsymbol{n}_3$ are the eigenvectors of $\Delta\mathbf{U}_{\mathrm{M}}$ and $\Delta\lambda_1$, $\Delta\lambda_2$, and $\Delta\lambda_3$ are their corresponding eigenvalues. Therefore, $\Delta\mathbf{U}_{\mathrm{M}}$ can be obtained by the generation of a set of orthogonal vectors $\boldsymbol{n}_1$, $\boldsymbol{n}_2$, and $\boldsymbol{n}_3$, and eigen-values $\Delta\lambda_1$, $\Delta\lambda_2$ and $\Delta\lambda_3$. The random orthogonal vectors $\boldsymbol{n}_1$, $\boldsymbol{n}_2$ and $\boldsymbol{n}_3$ are obtained by generating three uniformly distributed angular random variables with realizations $\alpha \in [0,\, \pi)$, $\beta \in [0,\, 2\pi)$ and $\gamma \in [0,\, 2\pi)$[1], with

$$\begin{aligned}
\boldsymbol{n}_1 &= [\cos\gamma\cos\alpha - \cos\beta\sin\alpha\sin\gamma \;\; \cos\gamma\sin\alpha + \cos\beta\cos\alpha\sin\gamma \;\; \sin\gamma\sin\beta]^{\mathrm{T}}, \\
\boldsymbol{n}_2 &= [-\sin\gamma\cos\alpha - \cos\beta\sin\alpha\cos\gamma \;\; -\sin\gamma\sin\alpha + \cos\beta\cos\alpha\cos\gamma \;\; \cos\gamma\sin\beta]^{\mathrm{T}}, \\
\boldsymbol{n}_3 &= [\sin\beta\sin\alpha \;\; -\sin\beta\cos\alpha \;\; \cos\beta]^{\mathrm{T}} \,.
\end{aligned} \tag{45}$$

The three eigenvalues define a step increment $\sqrt{\Delta\lambda_1^2 + \Delta\lambda_2^2 + \Delta\lambda_3^2} = \Delta R$, which is randomly picked at each time step from a given range $[\Delta R_{\min},\ \Delta R_{\max}]$. The eigenvalues are obtained by generating two uniformly distributed angular random variables with realizations $\theta \in [0,\, \pi)$ and $\phi \in [0,\, 2\pi)$[2], with

$$\Delta\lambda_1 = \Delta R\sin\theta\cos\phi \quad , \Delta\lambda_2 = \Delta R\sin\theta\sin\phi \quad \text{, and } \Delta\lambda_3 = \Delta R\cos\theta \,. \tag{46}$$

Since considering too small steps may result in too long loading sequences, which means long computational times for the meso-scale BVP resolution, and since too large steps prevent numerical convergence of the non-linear RVE resolution, ΔR follows a uniform distribution in the range $[1.0 \times 10^{-4},\, 5.0 \times 10^{-3}]$. As a result, ΔR has a probability higher than 80% of being larger than 1.0×10^{-3}. Nevertheless, a sub-stepping may be needed in order to guarantee the numerical convergence of the micro-scale resolution in the case ΔR is too large. Finally, the random walk or path is stopped after a maximum number of loading steps,

[1] In the 2D case, one has realizations $\alpha \in [0,\, \pi)$, $\beta = 0$ and $\gamma = 0$

[2] In the 2D case, one has realizations $\theta = \pi/2$ and $\phi \in [0,\, 2\pi)$

$N_{\max} = 2500$, is reached or when a criterion on the reached strain is met: *e.g.* when the eigen-values of $\mathbf{U}_{\mathrm{M}\,N}$ are such that $\max_{i=1,2,3} \{|\lambda_i - 1|\} > R_{\max}$, where $R_{\max}$ is a given critical value. In this work, $R_{\max} = 0.11$ is used to generate the random loading paths. The statistical information of 2000 generated random loading paths is illustrated in Fig. 7. Most of the loading paths reach the bound of $N_{\max} = 2500$ and Fig. 7(a) is thus truncated in its ordinates in order to depict the distribution of the shorter paths. The step increment ΔR, which corresponds to the L_2-norm of the stretch tensor increment, is observed to follow a uniform distribution in Fig. 7(b).

Besides random loading paths, proportional cyclic loading paths with randomized loading direction and reversal times are also used to create the training data. Indeed, some strain state histories might not be covered by the random loading paths, and proportional loading paths can help the training data to reach a better coverage of the strain space.

Details on the generation of the loading paths can be found in [18].

3.4. Data normalization and padding/trimming

Because the input/output features may not have the same scale, and to prevent the input of the activation functions to be out of their active range, all the input and output features must be normalized. A linear operation is used to map all the features to the range $[-1, 1]$. However, the normalization is not applied on the inputs with the format $\Delta\mathbf{E}_{\mathrm{M}}$, Eq. (42), since they will be normalized by their norm before feeding in the network, see Fig. 5(b). We note that when considering $\Delta\underline{\mathbf{E}}_{\mathrm{M}}$, see Figs. 3(b) and 4(b), we considered the difference of the normalized data and not the normalized difference.

Since the training data sequences are obtained through a random walking process which is terminated once a critical strain measure is reached, this results in different lengths of the data sequences. However, training a RNN with batches of data requires that each sequence within a training batch is of equal size. Therefore, on the one hand, for short sequences both zero padding at the beginning and repeatedly adding the last element at the ends of them are applied, and, on the other hand, long sequences are trimmed from their ends. Besides, zero padding at the beginning of the sequence is necessary in order to guarantee accurate prediction of the RNN for vanishing entries, e.g. at the beginning of multi-scale simulations for which parts of the structures are under zero deformation gradient.

More details on data normalization and padding/trimming can be found in [18].

3.5. Random training data augmentation

In practice, only moderate or large strain step sizes, ΔR, are used in the direct numerical resolution of the meso-scale BVP in order to balance the computational time and the strain ranges that could be reached at the end of each loading path. However, if we want to reinforce the consistency of the NNWs' prediction for various strain step sizes, different strain increment sizes, including smaller increments by several orders of magnitude, should be included in the training data. Therefore, a random data augmentation is applied on the training data.

Besides, when testing the self consistency of the trained NNWs, extra points will be inserted in the sequences of testing data, following the same data augmentation process.

Let us consider a sequential training data of "N" increments, with as inputs, $\mathbf{E}_{\mathrm{M}_0}$, $\mathbf{E}_{\mathrm{M}_1}, ..., \mathbf{E}_{\mathrm{M}_N}$ and output, $\mathbf{S}_{\mathrm{M}_0}$, $\mathbf{S}_{\mathrm{M}_1}, ..., \mathbf{S}_{\mathrm{M}_N}$. The random training data augmentation is detailed as follows

- A integer n_i $(i = 1, ..., N)$ is randomly picked from $[0, 1, ..., n_{\max}]$ and will be used as the number of insertion points for each data point interval; we will use the notation $n_{\max}^{\mathrm{training}}$ for the random training data augmentation and $n_{\max}^{\mathrm{testing}}$ for the random testing data augmentation;
- For each data point interval, n_i random variables β_k $(k = 1, ..., n_i)$ are generated following a uniform distribution on $(0, 1)$, and sorted in an ascending order;
- Then, n_i extra data points are inserted between $\mathbf{E}_{\mathrm{M}_{i-1}}$, $\mathbf{E}_{\mathrm{M}_i}$ and $\mathbf{S}_{\mathrm{M}_{i-1}}$, $\mathbf{S}_{\mathrm{M}_i}$, following a linear interpolation, as

$$\mathbf{E}_{\mathrm{M}_{i,k}} = \mathbf{E}_{\mathrm{M}_i} + \beta_k(\mathbf{E}_{\mathrm{M}_i} - \mathbf{E}_{\mathrm{M}_{i-1}})\,, \tag{47}$$

$$\mathbf{S}_{\mathrm{M}_{i,k}} = \mathbf{S}_{\mathrm{M}_i} + \beta_k(\mathbf{S}_{\mathrm{M}_i} - \mathbf{S}_{\mathrm{M}_{i-1}})\,, \tag{48}$$

with $(k = 1, ..., n_i)$.

In Eqs. (47) and (48), using the linear interpolation for the insertion points may not be accurate for non-linear problems. However, the original intervals, $[\mathbf{E}_{\mathrm{M}_i} - \mathbf{E}_{\mathrm{M}_{i-1}}]$, are already small with an order of magnitude 10^{-3}, since it was required to ensure the numerical convergence of the meso-scale BVP resolution. As a result, the first order approximation resulting from the linear interpolation of $\mathbf{S}_{\mathrm{M}_{i,k}}$ is accurate enough.

4. RNNs surrogate modeling of an elasto-plastic composite RVE

In this section, the RNNs presented in Section 3 are tested on a history-dependent elasto-plastic composite RVE. The material system used in [18] is reused in this work for comparison purpose. This material system is thus briefly recalled first before assessing the performance of the trained RNNs by comparing, for the testing paths, the results obtained by computational homogenization on the RVE with the RNN predictions .

4.1. Description of the meso-scale BVP

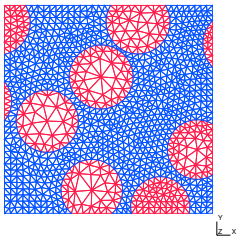


Figure 8: Finite element mesh of the micro-structural volume element of dimension 0.02mm×0.02mm used to generate the data-base.

A 2D RVE of a 39.9% continuous fiber reinforced elasto-plastic matrix material, as illustrated in Fig. 8, is used as the meso-scale BVP.

4.1.1. Fiber material model

The fibers obey a hyperelastic law based on an elastic potential

$$\psi_{\mathrm{f}}(\mathbf{C}) = \frac{K_{\mathrm{f}}}{2} \ln^2 J + \frac{\mu_{\mathrm{f}}}{4} (\ln \mathbf{C})^{\mathrm{dev}} : (\ln \mathbf{C})^{\mathrm{dev}} \ , \tag{49}$$

where $J = \det \mathbf{F}$ is the Jacobian, $\mathbf{C} = \mathbf{F}^{\mathrm{T}} \cdot \mathbf{F}$ is the right Cauchy strain tensor, $(\bullet)^{\mathrm{dev}}$ denotes the deviatoric part of an arbitrary second-order tensor $\bullet$, K_{f} and μ_{f} correspond respectively to the bulk and shear moduli of the material. The stress in the fiber phase is thus computed by

$$\mathbf{P} = \frac{\partial \psi_{\mathrm{f}}(\mathbf{F})}{\partial \mathbf{F}} = K_{\mathrm{f}} \mathbf{F}^{\text{-T}} \ln J + \mathbf{F}^{\text{-T}} \cdot \left[\mu_{\mathrm{f}} \ln \mathbf{C}^{\mathrm{dev}} \right] \ . \tag{50}$$

4.1.2. Matrix material model

A finite strain J_2 elasto-plastic constitutive model [51] is used for the matrix material. The deformation gradient $\mathbf{F}$ is decomposed into the reversible elastic part $\mathbf{F}^{\mathrm{e}}$ and the irreversible plastic part $\mathbf{F}^{\mathrm{p}}$ such that $\mathbf{F} = \mathbf{F}^{\mathrm{e}} \cdot \mathbf{F}^{\mathrm{p}}$.

The elastic potential energy reads,

$$\psi_{\mathrm{m}}(\mathbf{C}^{\mathrm{e}}) = \frac{K_{\mathrm{m}}}{2} \ln^2 J + \frac{\mu_{\mathrm{m}}}{4} (\ln \mathbf{C}^{\mathrm{e}})^{\mathrm{dev}} : (\ln \mathbf{C}^{\mathrm{e}})^{\mathrm{dev}} \ , \tag{51}$$

where $\mathbf{C}^{\mathrm{e}} = \mathbf{F}^{\mathrm{eT}} \cdot \mathbf{F}^{\mathrm{e}}$, and K_{m} and μ_{m} correspond respectively to the bulk and shear moduli of the matrix material. The first Piola-Kirchhoff stress tensor $\mathbf{P}$ derives from the elastic potential (51) following

$$\mathbf{P} = \frac{\partial \psi_{\mathrm{m}}\left(\mathbf{F}; \mathbf{F}^{\mathrm{p}}\right)}{\partial \mathbf{F}} = K_{\mathrm{m}} \mathbf{F}^{-\mathrm{T}} \ln J + \mu_{\mathrm{m}} \mathbf{F}^{\mathrm{e}} \cdot \left[\mathbf{C}^{\mathrm{e}\,-1} \cdot (\ln \mathbf{C}^{\mathrm{e}})^{\mathrm{dev}}\right] \cdot \mathbf{F}^{\mathrm{p}\,-\mathrm{T}} . \tag{52}$$

The elastic part $\mathbf{F}^{\mathrm{e}}$ and the plastic part $\mathbf{F}^{\mathrm{p}}$ of the deformation gradient are obtained through a J_2 plastic flow expressed in terms of the Kirchhoff stress, $\boldsymbol{\kappa} = \mathbf{P} \cdot \mathbf{F}^{\mathrm{T}}$. According to the J_2-plasticity theory, the von Mises stress criterion reads

$$f = \tau_{\mathrm{eq}} - \tau_{\mathrm{y}}^{0} - R(\gamma) \leq 0 , \tag{53}$$

where f is the yield surface, the equivalent von Mises stress is calculated through $\tau_{eq} = \sqrt{\frac{3}{2} \boldsymbol{\kappa}^{\mathrm{dev}} : \boldsymbol{\kappa}^{\mathrm{dev}}}$, τ_{y}^{0} is the initial yield stress, γ is the equivalent plastic strain and where the isotropic hardening stress $R(\gamma)$ takes the form

$$R(\gamma) = Y \left[1 - \exp(-k\gamma)\right] , \tag{54}$$

with Y and k material constants. Eventually, the evolution of $\mathbf{F}^{\mathrm{p}}$ is determined by the normal plastic flow theory following

$$\dot{\mathbf{F}}^{\mathrm{p}} = \dot{\gamma} \mathbf{N} \cdot \mathbf{F}^{\mathrm{p}} , \tag{55}$$

where $\mathbf{N}$ is the normal to the yield surface, see [51] for more details.

The material properties used in this work are listed in Table 1.

Table 1: Material properties for fibers and matrix.

Fiber		Matrix				
K_{f} [GPa]	μ_{f} [GPa]	K_{m} [GPa]	μ_{m} [GPa]	τ_{y}^{0}[MPa]	Y [MPa]	k [-]
16.67	12.50	2.50	1.15	100	20	30

4.2. Training of RNN

Around 15000 loading paths, among which 2/3 and respectively 1/3 are random paths and random cyclic loading paths, were used to generate the micro-scale finite simulations data-base. 75% of the data are randomly picked from the data-base for training, and the remaining 25% are kept for testing. The average length of these sequential data is around 1700, see Fig. 7(a), for the random paths and 130 for the cyclic loading paths. Both training and testing data are either padded or trimmed to a few lengths, which are noted by L_0, for different training purposes. Random data augmentation for both training and testing data is applied following Section 3.5, with the parameter $n_{\max}$, the original and final lengths of the training and testing sequential data being detailed later. The RNNs are trained with mini-batches, which include hundreds of sequential data, which are randomly picked from the training data. The backpropagation and learnable variables update are carried out at every epoch. Similarly, mini-batches of testing data are also used to evaluate the training process.

The Mean Square Error (MSE) is adopted for training and accuracy evaluation, and reads

$$L_{\mathrm{MSE}} = \frac{1}{n} \sum_{i=1}^{n} (\underline{v}_i - \underline{\hat{v}}_i)^2 , \tag{56}$$

where $\underline{v}_i$ and $\underline{\hat{v}}_i$ are respectively the actual and predicted normalized outputs. The learning rate is set to be 0.001, and the hidden variables are initialized by setting $\mathbf{h}_0 = \mathbf{0}$.

In the following section, the training performance of the RNNs developed in Section 3 is evaluated through the curves of training epochs vs. the corresponding MSEs. All the MSE plots report the MSE values at every 100 epochs only for a better visibility.

4.3. Selection of the hyper-parameters

For the RNNs developed in Section 3 and illustrated in Figs. 3(b), 4(b) and 5(b), the hidden variables number, $\hbar$, is the only hyper-parameter, besides the architecture of the non-linear transition block of the SC-MRU-I.

4.3.1. The number of hidden variables $\hbar$

The number of required hidden variables depends on the complexity of the corresponding physical problem [11], and needs to be determined on a case by case basis. Using the training data collected from direct numerical simulations on the composite RVE, the value of $\hbar$ is determined by examining the improvement of RNNs' training performance when the hidden variables number is increased.

Since the number of required hidden variables depends on the underlying physics, the effect of the number of hidden variables can be assessed by RNNs using only one type of recurrent unit. From the aspect of training speed, we consider the SMRU-based RNN, see Figs. 3(a) and 3(b), which uses the current total strain values as input and has the simplest architecture. A feed-forward neural network of one layer and the activation function "LeakyReLU" with negative slope 0.1 is adopted in $\boxed{\mathrm{Fw}_{\mathrm{Input}}}$, see Fig. 3(b).

According to our previous work [18], in which $\hbar = 100$ is used, $\hbar = 90,\ 120,\ 150$ are successively used in this study. The sequential data of original length $L_0 = 1600$ are used for the training, and random data augmentation with $n_{\max}^{\text{training}} = n_{\max}^{\text{testing}} = 2$ is applied, so that the final length of the training and testing sequential data is 3200. The training mini-batches containing 258 sequential data are changed every 30 epochs.

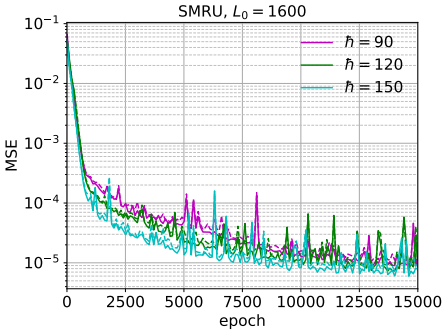


Figure 9: The evolution of the MSE vs. training epoch, in "solid line" for the training data and in "dashed line" for the testing data, for different numbers $\hbar$ of hidden variables and for the designed RNNs using SMRU as recurrent unit.

The evolution of the MSE during the training of the RNNs, using successively $\hbar = 90,\ 120,\ 150$, is plotted in Fig. 9. In general, for a given number of epochs, the MSE is lower when more hidden variables are used in the RNNs. In Fig. 9, it can be seen that the RNN with $\hbar = 150$ has reached a lower MSE than with $\hbar = 90\,, 120$, while the difference of MSEs between $\hbar = 90$ and 120 is not obvious. This could result from randomly picking the training mini-batches, which yields different training paths for the different trained RNNs. With the increase of training epochs (epoch > 12500 in Fig. 9), the advantage of using $\hbar = 120,\ 150$ becomes less obvious. It indicates that the effect of $\hbar$ begins to saturate at $\hbar = 120,\ 150$. On the one hand, using a high number of hidden variables increases the total number of RNNs' learnable parameters with an order of $\hbar^2$, see next Section 4.3.2; the time of one training epoch also increases and the training process of the RNNs is slowed down. On the other hand, for a given number of hidden variables, the MSE decreasing rate becomes low after 15000 epochs, a slight improvement in the MSE may require a few thousands more training epochs. Therefore, in order to reach a certain MSE, much more training epochs are needed when less hidden variables are defined in the RNN. The reason for requiring a reasonably high number of hidden variables results from the complex and different plastic deformation patterns (as illustrated in [11]) that develop on the adopted RVE, Fig. 8, under the various random loading paths that are considered in the training stage. Using a reasonably high number of hidden variables can help to capture the homogenized response of the

RVE accurately while decreasing the risk of over-fitting during the training of RNNs. Therefore, $\hbar = 120$ will be used for the comparative studies performed in the following sections.

4.3.2. RNNs' training performance comparison and effect of the non-linear transition block of the SC-MRU-I

Using $\hbar = 120$, the training performances of RNNs using different recurrent units are compared.

On the one hand, the RNN based on SMRU keeps the same architecture as for the hidden variables study in Section 4.3.1. The SC-MRU-T based RNN follows the information path as detailed in Eqs. (28-33), and uses "LeakyReLU" of 0.1 negative slope as activation function, meaning the same $\boxed{\text{Fw}_{\text{Input}}}$ as for the SMRU based RNN, see Figs. 3(b) and 4(b). For both RNNs, no activation function is applied for the output transition $\boxed{\text{Fw}_{\text{Output}}}$, see Figs. 3(b) and 4(b).

The learnable parameter n_0, in Eq. (28), is initialized to 0.05, which is close to the normalized value of the maximum loading step in the direct finite element simulations, see Sections 3.3.3 and 3.4.

On the other hand, two simple non-linear transition blocks, which can be expressed respectively as $\boxed{\text{Q}}$ and $\boxed{\text{2Fw}}$ according to Fig. 6(b), are successively adopted in the SC-MRU-I for the case of incremental input RNNs, see Fig. 5(a). The activation functions "tanh" and "LeakyReLU" with negative slope 0.1 are respectively adopted in $\boxed{\text{Q}}$ and in $\boxed{\text{2Fw}}$. For both cases, no activation function is applied for the output transition $\boxed{\text{Fw}_{\text{Output}}}$ in Fig. 5(b). The dimension $\hbar$ is chosen as the output dimension of all the linear operations in $\boxed{\text{Fw}_{\text{Input}}}$ and the non-linear transition blocks. The two blocks, $\boxed{\text{Q}}$ and $\boxed{\text{2Fw}}$, have a comparable amount of learnable parameters which can be computed by

$$N_{\text{lp}} = 2(\hbar + I_{\text{dim}} + 1) \times \hbar \quad \text{for} \quad \boxed{\text{Q}}, \tag{57}$$

$$N_{\text{lp}} = (\hbar + I_{\text{dim}} + 1) \times \hbar + (\hbar + 1) \times \hbar \quad \text{for} \quad \boxed{\text{2Fw}}, \tag{58}$$

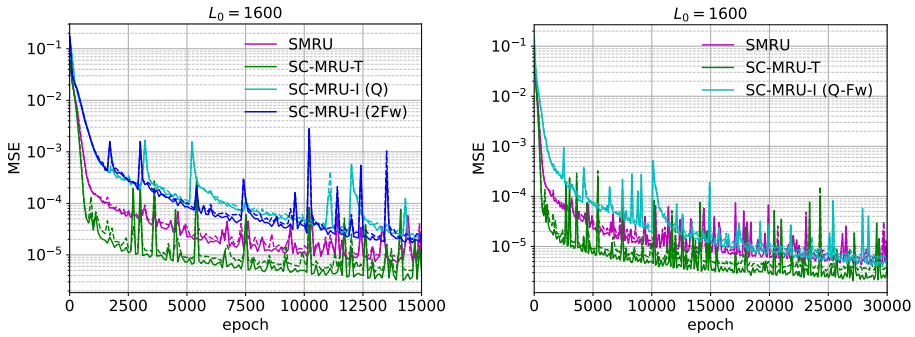
where N_{lp} is the number of learnable parameters, I_{dim} is the dimension of input variables ($I_{\text{dim}} = 3$ in our 2D problem), and where $(\hbar + I_{\text{dim}}) \times \hbar$ and $1 \times \hbar$ are the respective numbers of weights and biases in the first linear operation of the non-linear transition blocks. These choices of non-linear transition blocks in the SC-MRU-I yield a comparable amount of learnable parameters in the RNNs using self-consistent recurrent unit, see Table 2.

Table 2: The numbers of learnable parameters in the RNNs using recurrent unit with 120 hidden variables.

Recurrent unit	SMRU	SC-MRU-T	SC-MRU-I		
Transition block	-	-	Q	2Fw	Q-Fw
Learnable parameters	44284	58925	59644	59284	74164

The same training data and mini-batch shifting as in the hidden variable effect study are adopted. The evolution of the MSE during the training of the four RNNs is plotted in Fig. 10(a). From both aspects of training performance and number of learnable parameters, the results show that the RNN using the SMRU has a good training performance and much less learnable parameters than when using either the SC-MRU-T or the SC-MRU-I, see Table 2. The RNN using the SC-MRU-T shows the best training performance, while it has 30% more learnable parameters than when using the SMRU. For a given number of epochs, the RNNs using the total strain value as input exhibit a lower MSE than when using the incremental form of the strain as input. Therefore, in order to improve the training performances of the SC-MRU-I, different non-linear transition blocks are studied in Appendix B, motivating the use of $\boxed{\text{Q}} \rightarrow \boxed{\text{Fw}}$ as a non-linear transition block. At the cost of more learnable variables, see Table 2, its training performance improvement can be seen in Fig. 10(b).

The RNNs using the SC-LMSC architecture introduced in [40], and summarized in Section 3.1.2, are also trained with the strain-stress data collected from the composite RVE. The MSE evolution of the SC-LMSC using successively one and two non-linear transition layers is presented in Fig. 11, together with the



(a) (b)

Figure 10: The evolution of the MSE vs. training epoch, in "solid line" for the training data and in "dashed line" for the testing data, for the RNNs using either the SMRU, SC-MRU-T or SC-MRU-I with different non-linear transition blocks: (a) $\boxed{\text{Q}}$ and $\boxed{\text{2Fw}}$; and (b) $\boxed{Q} \rightarrow \boxed{\text{Fw}}$. The number of hidden variables is $\hbar = 120$.

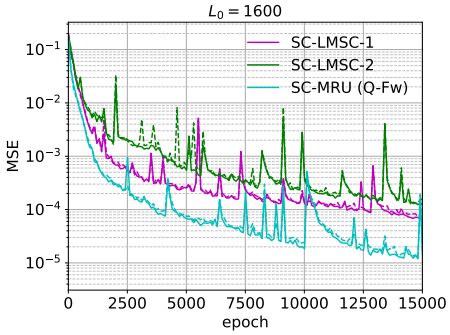


Figure 11: The evolution of the MSE vs. training epoch, in "solid line" for the training data and in "dashed line" for the testing data, for the RNNs using the SC-LMSC unit with either one or two non-linear transition layers and for the RNN using the SC-MRU-I unit with the non-linear transition block $\boxed{\text{Q}} \rightarrow \boxed{\text{Fw}}$. The number of hidden variables is $\hbar = 120$.

developed RNN using the SC-MRU-I with $\boxed{\text{Q}} \rightarrow \boxed{\text{Fw}}$ as a non-linear transition block. When considering a number of hidden variables $\hbar = 120$, the total numbers of learnable parameters are respectively 60748 and 91252 for the RNNs using one and two non-linear transition layers with the SC-LMSC unit, while the total number of learnable parameters is 74164 when using the SC-MRU-I.

Following the analysis of the MSE evolution plotted in Figs. 9-11, it can be seen that the MSEs for the training and testing data decrease with the same trend, which indicates that there is no over-fitting during the training. Therefore, more training epochs will yield a better accuracy of the RNNs.

Usually the random initialization of the learnable variables could have an impact on the training performance of the NNWs. Therefore, the three presented recurrent units, SMRU, SC-MRU-T and SC-MRU-I, have been trained from scratch for several times using training data of different lengths at the design stage. No obvious effect of the NNWs' initialization on their training performance has been observed. In Fig. 11, the SC-LMSC-1 and SC-LMSC-2 RNNs have been trained with four different initializations for each of them, and the training data showing the best performance have been reported in the Figure.

4.4. Effect of the input increment size and of the random data augmentation

According to the results of the conducted analysis, see Figs. 9-11, three RNNs are selected to study the effect of varying the input increment size of the testing data following the data augmentation described

in Section 3.5: the RNNs using SMRU, SC-MRU-T and the RNN using SC-MRU-I with the non-linear transition block $\boxed{\text{Q}} \rightarrow \boxed{\text{Fw}}$. We consider $\hbar = 120$ hidden variables.

4.4.1. RNN trained using random data augmentation with $n_{max}^{training} = 2$

Using the randomly augmented training data with $n_{\max}^{\text{training}} = 2$, the RNNs with $\hbar = 120$ were trained for 30000 epochs, for which the MSEs on the testing data with $n_{\max}^{\text{testing}} = 2$ were found to be around 5×10^{-6}, 3×10^{-6} and 5×10^{-6} respectively for the RNNs using the SMRU, SC-MRU-T and SC-MRU-I as recurrent unit, see Fig. 10(b).

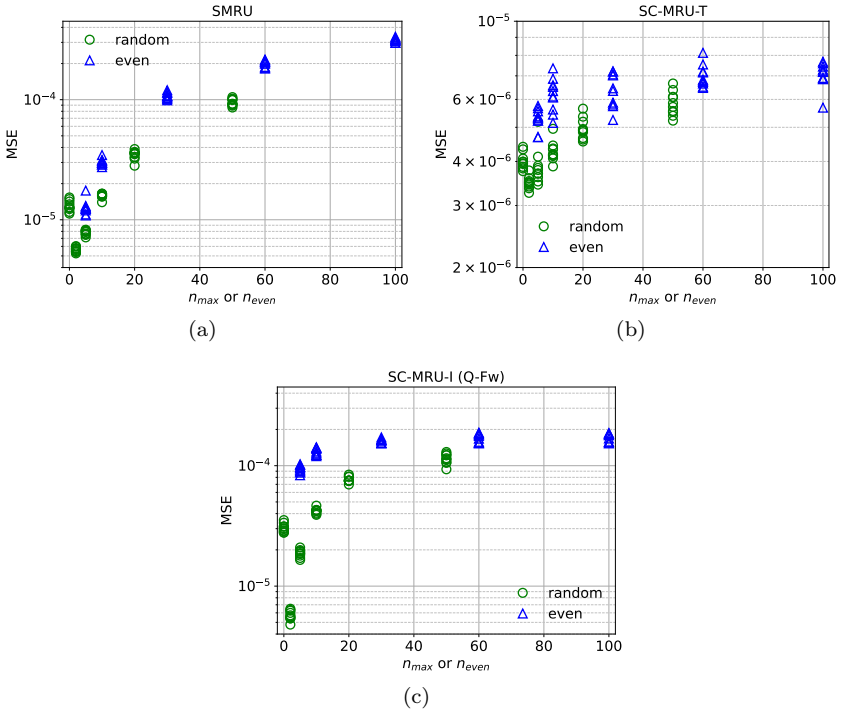


Figure 12: Testing MSE of the RNNs trained with randomly augmented data ($n_{\max}^{\text{training}} = 2$) on testing data vs. different random ($n_{\max}^{\text{testing}}$) and even ($n_{\text{even}}^{\text{testing}}$) testing data augmentation numbers: RNN using the (a) SMRU; (b) SC-MRU-T; and (c) SC-MRU-I (Q-Fw) as recurrent unit.

In order to assess the capability of the trained RNNs to predict the response for different input increment sizes, we successively consider a maximum number of randomly inserted points $n_{\max}^{\text{testing}} = 0, 2, 5, 10, 20, 50$ on the mini-batches randomly picked from the testing data, which has an original length $L_0 = 1600$. We also consider a data augmentation with evenly inserted points in the testing data, with $n_{\text{even}}^{\text{testing}} = 5, 10, 30, 60, 100$. Each batch includes more than 100 sequential data, and the MSEs of the RNNs predictions are computed for 10 different mini-batches at each value of $n_{\max}^{\text{testing}}$ and $n_{\text{even}}^{\text{testing}}$. The obtained MSEs are plotted in Fig. 12.

From Fig. 12(b), it can be seen that for a fixed random augmentation parameter $n_{\max}^{\text{training}} = 2$ applied on the training data, and then used to train the RNNs, the prediction accuracy of the RNN using the SC-MRU-T as recurrent unit is not sensitive to the number of inserted points ($n_{\max}^{\text{testing}}$ or $n_{\text{even}}^{\text{testing}}$) in the testing data. However, for the RNNs using the SMRU and SC-MRU-I (Q-Fw), the MSEs increase when introducing more points in the testing data than in the training data, in Figs. 12(a) and 12(c). In particular, when using the SMRU, the MSE keeps an increasing trend with the number of introduced points in the testing data, in Fig. 12(a), while the MSE of the RNN using the SC-MRU-I stops increasing after the introduced

points number reaches 30, see Fig. 12(c), which indicates an effect of the self-consistency reinforcement in the recurrent unit. One reason for these accuracy decreases might be that the training data does not include enough variation of the input increment size.

4.4.2. SMRU and SC-MRU-I based RNNs further trained using random data augmentation with various $n_{max}^{training}$

Random training data augmentation with $n_{\max}^{\text{training}} = 2$ provides training data with feasible sequential length and with various input increment sizes so that the RNNs can be trained within a reasonable time while reaching a targeted accuracy. However, the predictions of the RNNs using the SMRU and the SC-MRU-I as recurrent unit are not always accurate when the input increment size is decreased, as shown when considering random testing data augmentation with various $n_{\max}^{\text{testing}}$ and $n_{\text{even}}^{\text{testing}}$ numbers. Therefore, in order to improve the predictions of the RNNs using either the SMRU or the SC-MRU-I for a wide range of input increment sizes, we further train the RNNs through a warm-start using randomly augmented training data. In this further training process, the RNNs are trained with mini-batches whose random data augmentation parameter is successively picked among $n_{\max}^{\text{training}} = 0$, 5, 15 and 30.

- For the SMRU based RNN, the corresponding original sequential data lengths, L_0 are padded or trimmed to 2500, 600, 400 and 200, and the lengths after data augmentation are respectively around 2500, 2100, 3400 and 3200. The mini-batches include around 250 sequential data, and are changed after every 10 training epochs. The learning rate of 0.001 is kept.

- It has been found that the SC-MRU-I based RNN will not provide good prediction on long sequential data, if it has been further trained with short sequential data. Thus, for the SC-MRU-I based RNN, the corresponding original sequential data length, $L_0 = 1600$, is used and the lengths after data augmentation are respectively around 1600, 5500, 13000 and 26000. The mini-batches include respectively around 750, 150, 100 and 50 sequential data, and are changed after every 10 training epochs. A reduced learning rate of 0.0001 is adopted in this further so called fast-shifting training process.

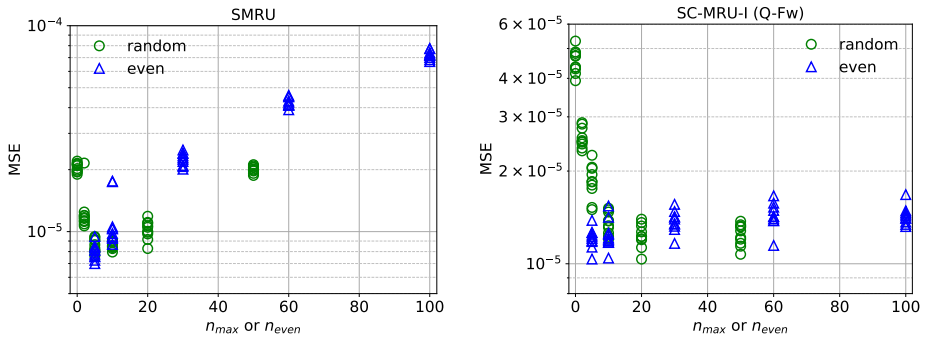
After around 4000 training epochs, the MSEs of the RNNs predictions are computed using the testing data sampled in the way described in Section 4.4.1, and are plotted in Figs. 13. Fig. 13(a) shows that the accuracy of the RNN using the SMRU improves by the described short reinforced training, especially for the testing data augmented with $n_{\max}^{\text{testing}}$ comparable to the adopted $n_{\max}^{\text{training}}$ in training. A similar improvement is also seen in Figs. 13(b) for the RNN using the SC-MRU-I, except for the MSE on testing data without augmentation, $n_{\max}^{\text{testing}} = 0$. The effect of self-consistency is still obvious, as the value of MSE becomes stable with the decrease of the input incremental size (increase of $n_{\max}^{\text{testing}}$ and $n_{\text{even}}^{\text{testing}}$ in testing data augmentation).

In Figs. 13, the lowest MSE, among all the tested $n_{\max}^{\text{testing}}$ and $n_{\text{even}}^{\text{testing}}$, is higher than that of the RNNs before the short reinforced training being applied. It indicates that it is difficult to reach a good prediction accuracy for a wide range of input incremental size with these two RNNs. Using more learnable parameters in these two RNNs, such as increasing the number of hidden variables, could be a solution. Besides, training them with data having the estimated range of input increment size as that in the targeted applications will provide the best prediction accuracy.

4.5. Comparisons of the RNNs predictions to the direct finite element simulations

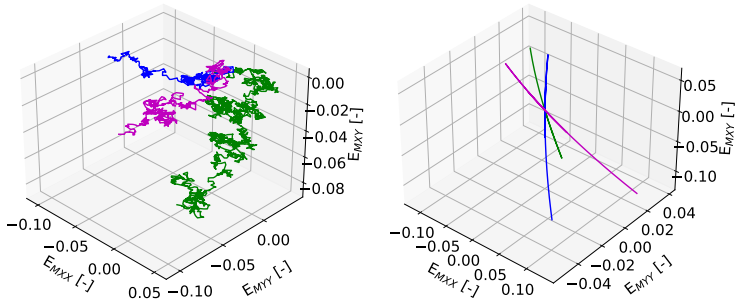
The accuracy of the RNNs using a self consistency reinforced recurrent unit is also assessed by comparing the RNNs predictions to the direct finite element simulations conducted at the RVE level, for loading cases which were not part of the training data. We successively consider three random loading paths and three cyclic loading paths as illustrated in Fig. 14 in terms of the $\{E_{\mathrm{M}_{XX}}, E_{\mathrm{M}_{YY}}, E_{\mathrm{M}_{XY}}\}$-trajectories.

The random data augmentation is applied on the green and blue loading paths, in Fig. 14 with $n_{\max}^{\text{testing}} = 5$ and 20, respectively, while an even data augmentation is applied on the magenta loading paths with $n_{\text{even}}^{\text{testing}} = 60$, in order to assess the capability of the RNNs to predict the solution with lower input increment sizes than seen during the training. The RNNs using the SC-MRU-T and SC-MRU-I (Q-Fw) as recurrent units are considered in this study. A further training with fast-shifting of mini-batches has been applied on the



(a) (b)

Figure 13: Testing MSE vs. n^{testing} of the RNNs using the (a) SMRU; and (b) SC-MRU-I (Q-Fw), after a warm-start training with fast-shifting of mini-batches and for different $n_{\max}^{\text{training}}$ numbers of the random training data augmentation.



(a) (b)

Figure 14: Randomly picked loading paths from the testing data: (a) Random loading paths; and (b) Cyclic loading paths.

RNN using the SC-MRU-I (Q-Fw) following Section 4.4.2. The number of hidden variables, $\hbar = 120$ is used for all the RNNs.

Table 3: The MSEs of the RNN predictions for the randomly picked loading paths from the testing data illustrated in Fig. 14.

Loading path		green	blue	magenta
Data augmentation		$n_{\max}^{\text{testing}} = 5$	$n_{\max}^{\text{testing}} = 20$	$n_{\text{even}}^{\text{testing}} = 60$
RNN using	Random path	$1.5\,\text{e}-5$	$6.6\,\text{e}-6$	$1.2\,\text{e}-5$
SC-MRU-T	Cyclic path	$1.3\,\text{e}-6$	$6.9\,\text{e}-7$	$1.9\,\text{e}-6$
RNN using	Random path	$4.2\,\text{e}-5$	$6.0\,\text{e}-6$	$5.1\,\text{e}-6$
SC-MRU-I	Cyclic path	$2.5\,\text{e}-6$	$4.6\,\text{e}-6$	$1.1\,\text{e}-5$

The MSEs of the RNNs predictions are reported in Table 3. It can be seen that both RNNs provide predictions with good accuracy. The stress evolution predicted by RNNs using the SC-MRU-T and the SC-MRU-I (Q-Fw) as recurrent unit are compared to the RVE direct numerical simulations in respectively Figs. 15 and 16. The accuracy and prediction consistency with respect to the input increment size are verified.

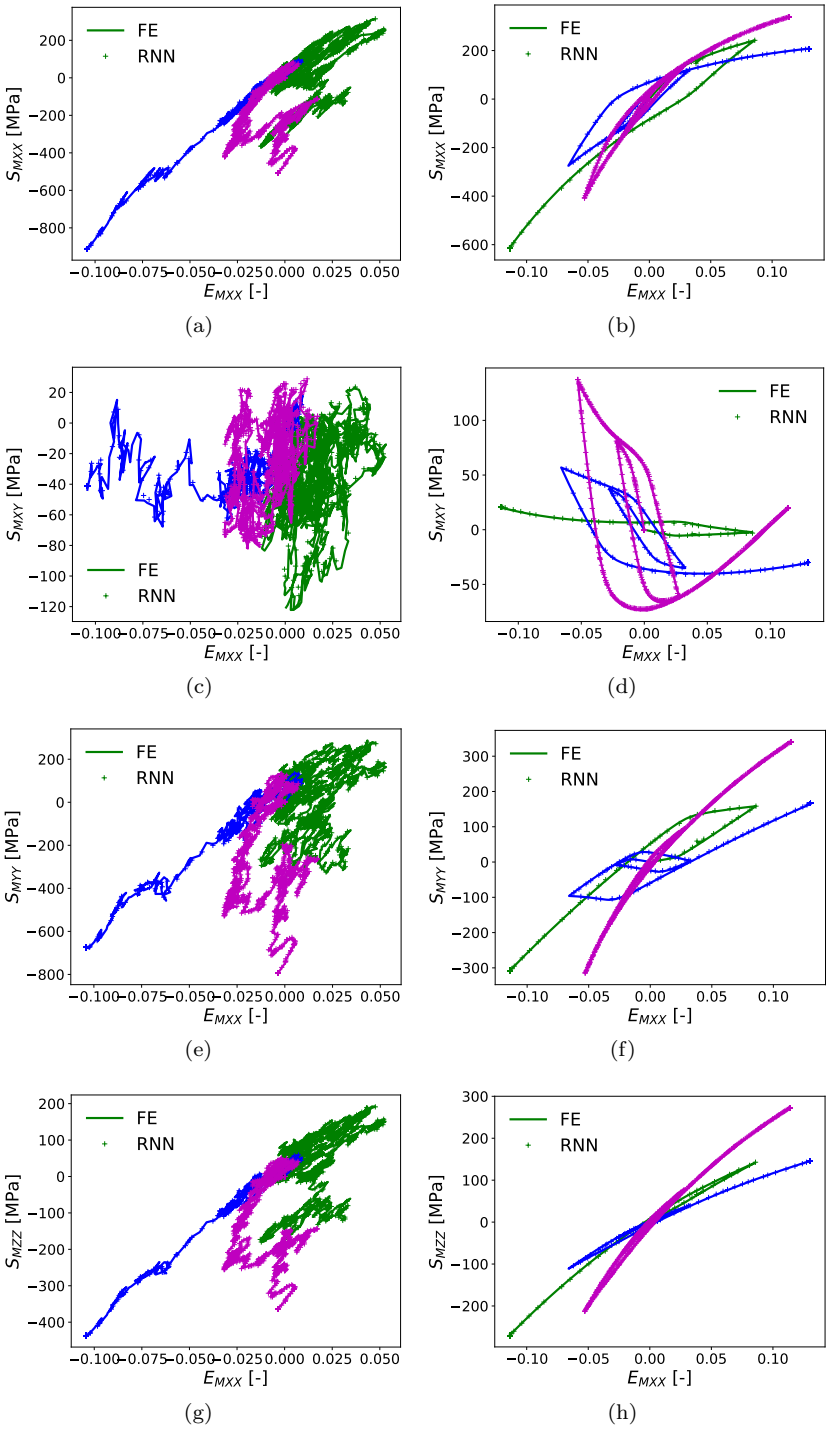


Figure 15: Comparison between the strain-stress curves obtained by direct finite element simulations conducted at the RVE level and by RNN predictions using the SC-MRU-T on the data augmented loading paths illustrated in Fig.14.

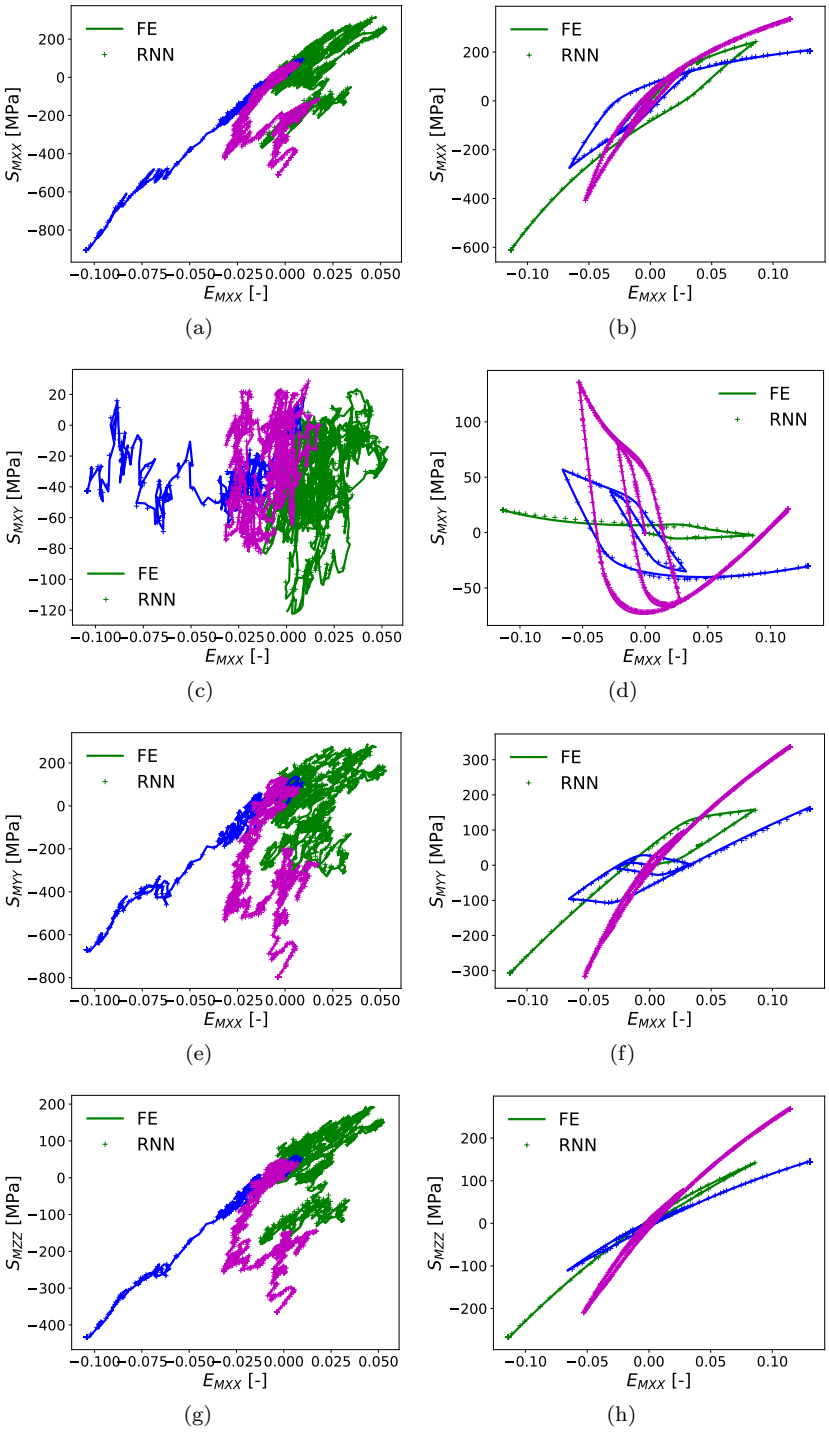


Figure 16: Comparison between the strain-stress curves obtained by direct finite element simulations conducted at the RVE level and by RNN predictions using the SC-MRU-I on the data augmented loading paths illustrated in Fig.14.

5. Multi-scale analysis of an open hole sample

In this section, the RNN is used as a surrogate of the material law (4), which represents the RVE response, so that the finite element analysis can be conducted at the sole macro-scale. An open hole sample subjected to several loading/unloading cycles is studied and the response obtained with a fully coupled FE^2 analysis serves as a reference. In particular, the effect of the loading step size on the accuracy of the surrogate model-based simulation is discussed.

The training with fast-shifting of mini-batches as described in Section 4.4.2 is adopted in the further training of the RNNs. In order to have better predictions in a wider input variables range, the data of original length $L_0 = 2500$ is used. We also use data trimmed to $L_0 = 200$ to improve the predictions in low output value ranges, i.e. at small strain. The RNNs are further trained for around 3000 epochs using training data with random data augmentation using $n_{\max}^{\text{training}} = 2$.

5.1. Implementation of RNN surrogate model in multi-scale analyzes

The implementation of the RNN as a surrogate model of the material law (4) follows the traditional implementation of a constitutive model in a finite element code, with the particularity that the trained RNN is stored as a function of the material law [18]. Different from the work in [18], the RNN is coded and trained with the C++ version of PyTorch, LibTorch [49]. In order to store it as a script module which can be loaded either from Python or from a compiled C++ code, the trained learnable parameters of the RNN are reloaded in their Python version to save the module, according to the available function of PyTorch.

In order to track the history-dependency, at each Gauss point, the hidden variables $\mathbf{h}$ of the RNN are saved as internal or state variables $\boldsymbol{Z}_{\mathrm{M}}(\boldsymbol{X}_{\mathrm{M}}, t)$ of Eq. (4), and updated when evaluating the RNN model.

Although the "Automatic Differentiation function" of NNWs can be used to compute the macro-scale material tensor $\mathbb{C}_{\mathrm{M}}$, which is defined by $\frac{\partial \mathbf{P}_{\mathrm{M}}}{\partial \mathbf{F}_{\mathrm{M}}}$, –in which case it is actually $\frac{\partial \mathbf{S}_{\mathrm{M}}}{\partial \mathbf{E}_{\mathrm{M}}}$ that is evaluated and that is transformed to get $\mathbb{C}_{\mathrm{M}}$–, this function not only requires to keep the gradient of all the torch tensors which demands much more computer memory, but also turns out to be slower than using perturbation. Thus, the perturbation method is adopted to compute the macro-scale material tensor.

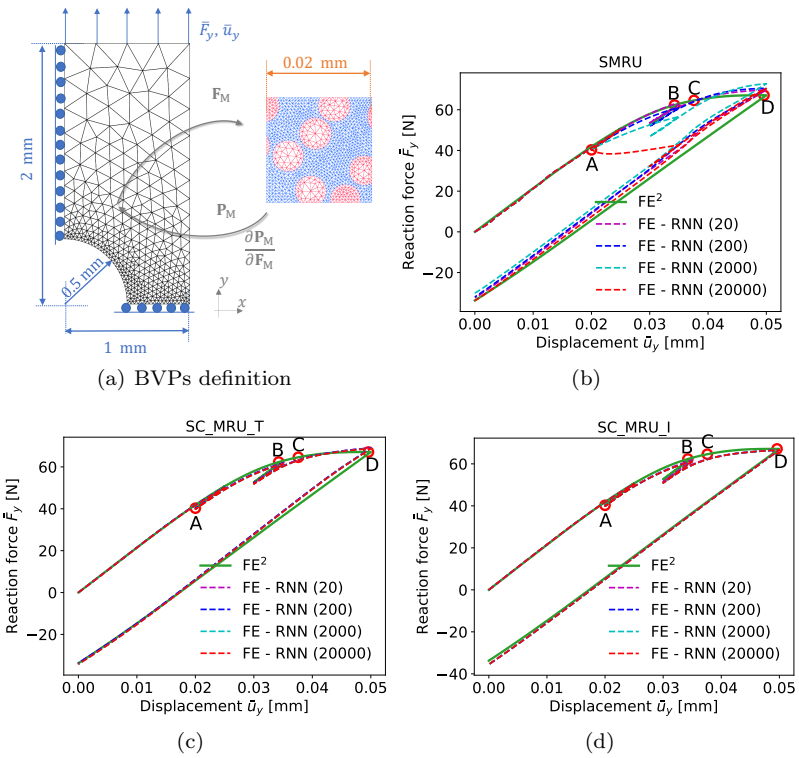
5.2. Numerical application on an open hole sample

At the macro-scale, the open hole sample illustrated in Fig. 17(a), is considered in this section, with the meso-scale BVP being defined by the RVE described in Section 4.1. The displacement boundary condition applied on the open hole sample is illustrated in Fig. 17(a): the sample is loaded on its top edge under controlled displacement, which increases from 0 to 0.05 mm with two partial unloading-reloading cycles in the middle and a final decrease to 0. Using the geometrical and loading symmetry, only one quarter of an open hole sample is simulated.

A concurrent multi-scale analysis, denoted by FE^2, is conducted in which the meso-scale BVP is solved using the finite element method in a concurrent way with the macro-scale BVP. The resulting reaction force vs. displacement curve serves as a reference in Figs. 17(b), 17(c) and 17(d). On the displacement-reaction force curve, four points are marked respectively by "A" for the end of the first unloading, "B" for the beginning of second unloading, "C" for the onset of the maximum strain, $E_{\mathrm{M}_{YY}}$, being out of the RNNs training data range as it will be further discussed, and by "D" for the maximum reached displacement.

In order to study the effect of the loading step size on the predictions of the multi-scale analysis using the RNN as surrogate model, 20, 200, 2000 and 20000 steps are successively used from loading points "A" to "B". The responses predicted by the three RNNs, using the SMRU, SC-MRU-T and SC-MRU-I (Q-Fw) as recurrent units are also reported in Figs. 17(b), 17(c) and 17(d), respectively.

In Fig. 17(b), when 20 steps are used between points "A" and "B", a good accuracy on the displacement-reaction force curve, till point "C", is obtained by the RNN surrogate model using the SMRU. Its predictions become less accurate when more than 200 steps are used between points "A" and "B". It indicates that this RNN's prediction is inconsistent for different input incremental sizes. While for the RNN surrogate models using respectively the SC-MRU-T and the SC-MRU-I in Figs. 17(c) and 17(d), the overlap of the displacement-reaction force curves shows that not only a good accuracy is obtained, but also RNNs' predictions are consistent for various input incremental sizes.



(a) BVPs definition (b) (c) (d)

Figure 17: Multi-scale simulation: (a) Definition of the macro-scale and meso-scale BVPs; and displacement vs. reaction force curves obtained by the FE^2 and the FE-RNN analyses using respectively the recurrent units (b) SMRU, (c) SC-MRU-T and (d) SC-MRU-I, where the FE-RNN analyses have considered successively 20, 200, 2000 and 20000 steps between the loading points "A" and "B".

According to the RNNs training data preparation, Section 3.3.3, the non-normalized input incremental size distributes in the range $[1.0 \times 10^{-4}, 5.0 \times 10^{-3}]$, which is less than or equal to the loading step size used for meso-scale BVP resolution in FE^2. This may explain why the displacement-reaction force curves obtained by the RNN surrogate models are slightly lower than by the FE^2 method, in Figs. 17(c) and 17(d).

Figures 18-23 illustrate the macro-scale Green-Lagrange strain and Cauchy stress fields distributions predicted by the nested computational homogenization, or FE^2 method, which serves as a reference solution, and by the finite element simulations using the different developed RNNs. The distributions correspond to the loading points "B" (Figs. 18 and 19), "C" (Figs. 20 and 21) and "D" (Figs. 22 and 23) reported on Figs. 17(b)-17(d). For all the simulations, 20 steps were considered between the points "A" and "B" reported on Figs. 17(b)-17(d). At points "B" and "C", the RNN with the SMRU as recurrent unit clearly exhibits a larger error in both the strain and stress predictions as compared to the RNNs using the self-consistent recurrent units. This is in agreement with the deviation observed in the loading curve in Fig. 17(b). Up to point "C", both RNNs using either the SC-MRU-T or the SC-MRU-I as recurrent unit exhibit fields distributions in good agreement with the FE^2 method: the maximum error is locally 10% in the strain distribution field at the notch, see Figs. 18(d), 21(c) and 21(d). We note that, at this particular location, the strain field of the reference solution reaches the bound of the training data: the values of the Green-Lagrange strain used for the training data are in the range of $[-0.11, 0.11]$. For the loading point "D", the local strain reaches 0.3 at the notch for the reference solution as illustrated in Fig. 22(a), well beyond the training range. The resulting errors of the RNN surrogate models are therefore much higher as it can be seen in Figs. 22 and 23, which explains the difference observed in the loading curves beyond point "C" in Figs. 17(b), 17(c) and 17(d).

The comparison of the computational cost breakdown of the multi-scale simulations using the nested

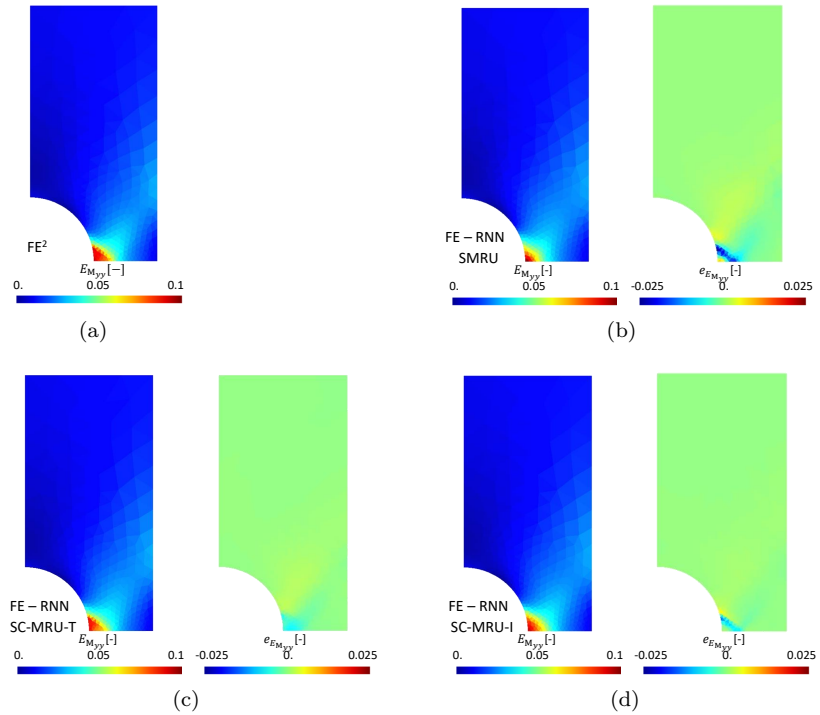


Figure 18: Comparisons of the Green-Lagrange strain, $E_{M_{YY}}$, distribution of an open-hole sample obtained, at loading point "B", (a) by the FE^2 method; and using the RNN surrogate model with the (b) SMRU; (c) SC-MRU-T and (d) SC-MRU-I as recurrent units; The loading point "B" refers to the marker on Figs. 17(b)-17(d). For the surrogate-based simulations, the right figure depicts the absolute error $e_{E_{M_{YY}}}$ with respect to the FE^2 method.

Table 4: Computational cost breakdown.

Pre-off-line	FE^2	SMRU	SC-MRU-T	SC-MRU-I
Data generation	-	23500 hr-cpu		
Training	-	within 10 hr-cpu		
On-line	**FE^2**	**SMRU**	**SC-MRU-T**	**SC-MRU-I**
Multi-scale simulation	18000 hr-cpu	0.27 hr-cpu	0.38 hr-cpu	0.28 hr-cpu

computational homogenization, or FE^2 method, and using the different developed RNNs is reported in Table 4. The computation time of the nested FE^2 multi-scale simulation is around 30 hours using 600 processors on a cluster. In comparison the analyses with the RNN surrogate models require around half an hour on a single Core AMD Ryzen Threadripper PRO 5995WX 2.7GHz processor. However, training the surrogate model is also computationally demanding. On the one hand, generating the 15000 loading data required around 470 hours using 50 AMD Ryzen Threadripper PRO 5995WX 2.7GHz processors. We note that this step is fully scalable. On the other hand, the training of each surrogate took within 10 hours on a personal computer with a single Core i7-1165G7 CPU 2.8GHz processor. Nevertheless, once trained, the RNNs can readily be used to conduct other multi-scale simulations.

6. Conclusions

In this work, we designed three recurrent units which can be used in a Recurrent Neural Network (RNN) to serve as a surrogate model of a meso-scale BVP in the context of computational multi-scale analyses. These three units are abbreviated as SMRU for Simplified Minimal Recurrent Unit, SC-MRU-T

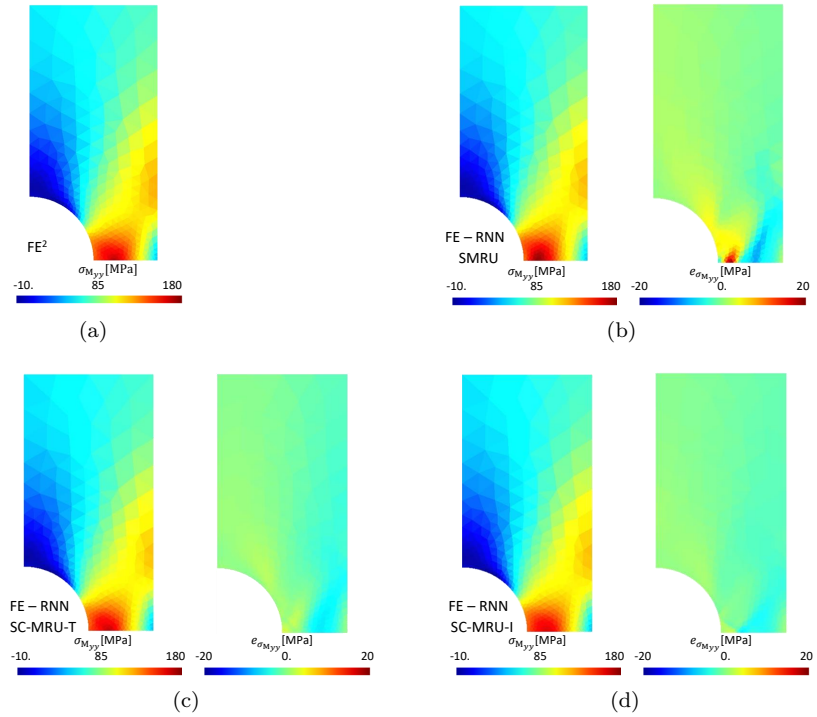


Figure 19: Comparisons of the Cauchy stress, $\sigma_{\mathrm{M}_{YY}}$, distribution of an open-hole sample obtained, at loading point "B", (a) by the FE^2 method; and using the RNN surrogate model with the (b) SMRU; (c) SC-MRU-T and (d) SC-MRU-I as recurrent units; The loading point "B" refers to the marker on Figs. 17(b)-17(d). For the surrogate-based simulations, the right figure depicts the absolute error $e_{\sigma_{\mathrm{M}_{YY}}}$ with respect to the FE^2 method.

for Self-Consistency reinforced Minimal Recurrent Unit using the Total strain as input, and SC-MRU-I for Self-Consistency reinforced Minimal Recurrent Unit using the strain Increment as input. Compared to the GRU, the presented recurrent units have simplified architectures, and the RNNs based on them have been used as surrogate models to substitute the homogenization of history-dependent elasto-plastic composite in multi-scale analyses.

On the one hand, among the presented three recurrent units, SMRU has the simplest architecture and uses the fewest learnable parameters. Thus the RNNs based on this unit have a fast training process. However, these RNNs can only provide predictions with good accuracy when the input increment size is comparable to the one of the training data. This inconsistency issue can be partly relieved by using training data with random augmentation: after trained with randomly augmented data, the RNN yields good results for input increment sizes belonging to a given range.

On the other hand, the RNNs based on the self-consistent recurrent units, either SC-MRU-T or SC-MRU-I, have demonstrated their consistency on the predictions for various input increment sizes. In particular, the RNN using the SC-MRU-T, which uses a moderate number of learnable parameters, shows the best training performance and yields accurate predictions for input increment sizes varying in a wide range. Reasonable training performance is obtained for the RNN using the SC-MRU-I, at the cost of an increased number of learnable parameters. However, in order to have accurate prediction for a wide range of input increment size, the RNN using the SC-MRU-I as recurrent unit needs to be trained with randomly augmented data using several different parameters $n_{\max}^{\text{training}}$.

In structural analyses, a similar accuracy as with a FE^2 multi-scale simulation can be reached with the RNN-based surrogate models as long as the loading case remains in the training range. When using the RNNs based on the SC-MRU-T and SC-MRU-I, consistent macroscopic responses have been obtained for various macroscopic loading steps, demonstrating that the RNNs based on the presented self-consistent recurrent

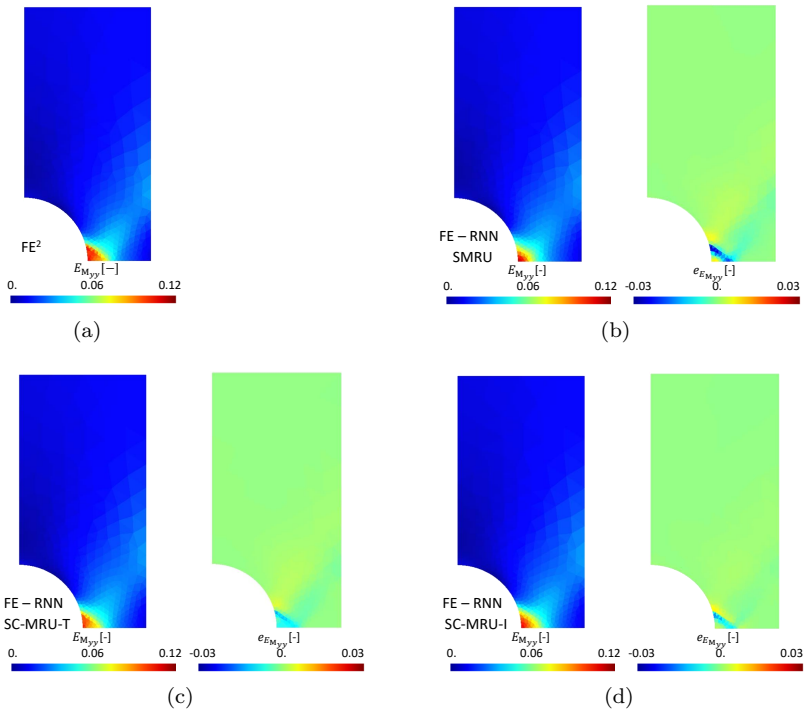


Figure 20: Comparisons of the Green-Lagrange strain, $E_{\mathrm{M}_{YY}}$, distribution of an open-hole sample obtained, at loading point "C", (a) by the FE^2 method; and using the RNN surrogate model with the (b) SMRU; (c) SC-MRU-T and (d) SC-MRU-I as recurrent units; The loading point "C" refers to the marker on Figs. 17(b)-17(d). For the surrogate-based simulations, the right figure depicts the absolute error $e_{E_{\mathrm{M}_{YY}}}$ with respect to the FE^2 method.

units have the potential to serve as reliable surrogate models in computational mechanics applications to accelerate the analysis process.

Acknowledgment

This project has received funding from the European Union's Horizon Europe Framework Programme under grant agreement No. 101056682 for the project "DIgital DEsign strategies to certify and mAnufacture Robust cOmposite sTructures (DIDEAROT)". The contents of this publication are the sole responsibility of ULiege and do not necessarily reflect the opinion of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Data availability

The raw/processed data required to reproduce these findings is available on `https://gitlab.uliege.be/didearot/didearotPublic/publicationsData/2024_scmru` under the Creative Commons Attribution 4.0 International (CC BY 4.0) licence [50].

Appendix A. Effect of the modified update hidden variable in the Minimal Gated Recurrent Unit

In this section we modify the computation of the update hidden variable in the Minimal Gated Recurrent Unit (MGRU), Eq. (17), which becomes

$$\hat{\mathbf{h}}_t = \tanh(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_h)\,. \tag{A.1}$$

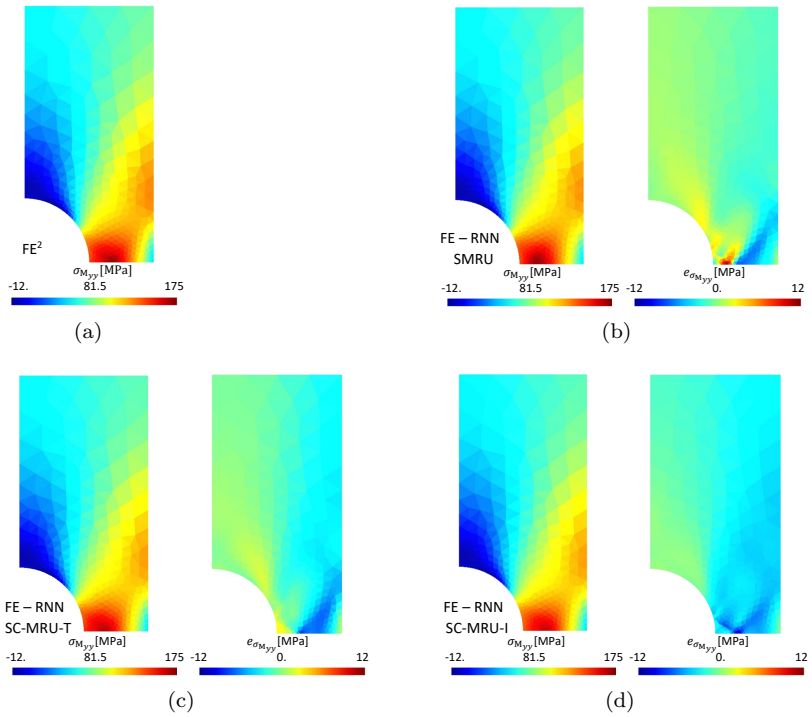


Figure 21: Comparisons of the Cauchy stress, $\sigma_{\mathrm{M}_{YY}}$, distribution of an open-hole sample obtained, at loading point "C", (a) by the FE^2 method; and using the RNN surrogate model with the (b) SMRU; (c) SC-MRU-T and (d) SC-MRU-I as recurrent units; The loading point "C" refers to the marker on Figs. 17(b)-17(d). For the surrogate-based simulations, the right figure depicts the absolute error $e_{\sigma_{\mathrm{M}_{YY}}}$ with respect to the FE^2 method.

We study the training performance of the two RNNs:

- One RNN adopts the original MGRU, Eqs. (16-18); and
- The other one uses the MGRU with the modified $\hat{\mathbf{h}}_t$, Eq (A.1).

We use the augmented data with $n_{\max}^{\text{testing}} = n_{\max}^{\text{testing}} = 2$, see Section 3.5. The original sequential data length is $L_0 = 1600$, the mini-batches contain 258 sequential data and are shifted every 30 epochs. The normalized value of $\mathbf{E}_{\mathrm{M}}$ is used as input. The number of hidden variables is set to 120. A $\boxed{\text{Fw}}$ layer $\{3, 120\}$ is applied to convert the input variables to the same dimensions as the hidden variables, and the "LeakyReLU" with negative slope 0.1 is used as activation function.

Using a learning rate of 0.001, the MSE vs. training epochs of the two RNNs are plotted in Fig. A.24. For the history-dependent non-linear RVE considered in this work, the training performance of the MGRU with the modified $\hat{\boldsymbol{h}}_t$ shows a slight improvement compared to the original version.

Appendix B. Effect of the non-linear transition blocks on the training performances of the RNN using the SC-MRU-I

According to the architecture of the SC-MRU-I, see Fig. 5(a) in Section 3.2.3, the key issue is to find the optimal design of the non-linear transition block. We first compare the effect of the non-linear transition block by considering either repeated $\boxed{\text{Q}}$ or repeated $\boxed{\text{Fw}}$ blocks. The blocks $\boxed{\text{Q}}$, $\boxed{\text{2Q}}$, $\boxed{\text{3Q}}$, and $\boxed{\text{2Fw}}$, $\boxed{\text{3Fw}}$, $\boxed{\text{4Fw}}$ are successively considered, with the activation function "LeakyReLU" with negative slope 0.1 being used in blocks $\boxed{n\text{Fw}}$. The numbers of learnable parameters of the RNNs when using these

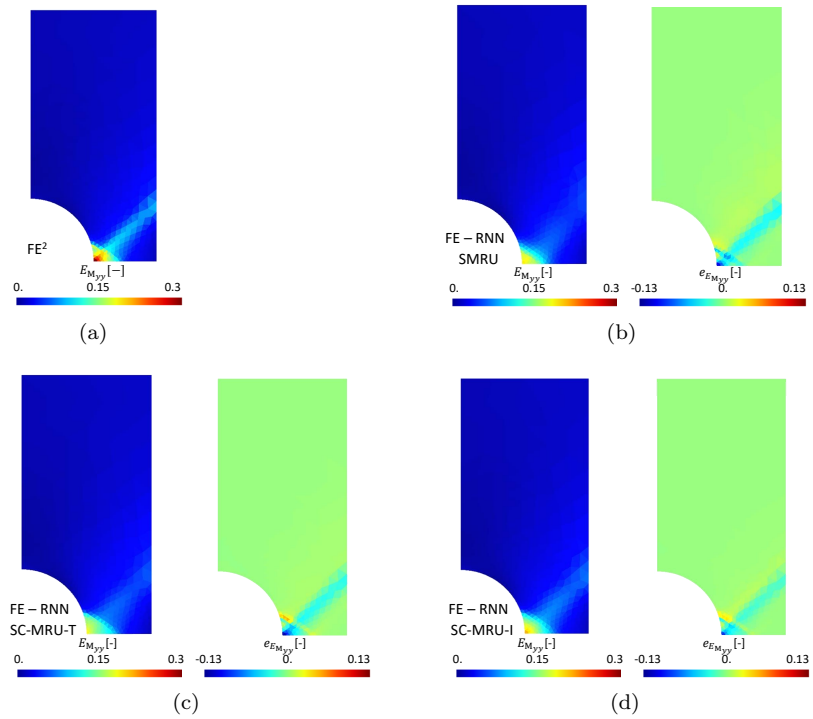


Figure 22: Comparisons of the Green-Lagrange strain, $E_{\mathrm{M}_{YY}}$, distribution of an open-hole sample obtained, at loading point "D", (a) by the FE^2 method; and using the RNN surrogate model with the (b) SMRU; (c) SC-MRU-T and (d) SC-MRU-I as recurrent units; The loading point "D" refers to the marker on Figs. 17(b)-17(d). For the surrogate-based simulations, the right figure depicts the absolute error $e_{E_{\mathrm{M}_{YY}}}$ with respect to the FE^2 method.

transition blocks are listed in Table B.5. The sequential data of original length $L_0 = 1600$ with random data augmentation $n_{\max}^{\text{training}} = n_{\max}^{\text{testing}} = 2$ are used for the training study of this section. The training mini-batches contain 258 sequential data and are shifted every 30 epochs. The training performance of these RNNs is assessed by comparing their MSE evolution with the number of epochs in Fig. B.25.

The Fig. B.25(a) shows that, after 15000 training epochs, the RNNs using either $\boxed{\text{2Q}}$ or $\boxed{\text{3Q}}$ as transition block reach lower MSEs than the RNN using $\boxed{\text{Q}}$. However, the RNN using $\boxed{\text{3Q}}$ does not exhibit better training performance than the one using $\boxed{\text{2Q}}$. The RNNs using the non-linear transition block $\boxed{n\text{Fw}}$ have almost the same training performance for $n = 2$ and 3, while a slightly lower MSE is reached when using $\boxed{\text{4Fw}}$. In Fig. B.25, it can be seen that the effect of using more non-linear transition layers leads to a saturation of the performance improvement: adding more non-linear transition layers not always makes the MSE decrease faster during training. In order to identify the optimal non-linear transition blocks sequence, one needs to consider both the number of epochs required for the MSE to reach a given thresehold, e.g. below 1.0×10^{-5}, as well as the training time required per epoch. From Table B.5 and Fig. B.25, it appears that the RNN with $\boxed{\text{2Fw}}$ shows better training performance than with $\boxed{\text{3Fw}}$, and the RNN with $\boxed{\text{2Q}}$ than with $\boxed{\text{3Q}}$, while the RNN with $\boxed{\text{2Q}}$ shows comparable performance as with $\boxed{\text{4Fw}}$. The training performance of the RNNs with $\boxed{\text{Q}}$ is similar as with $\boxed{\text{2Fw}}$, and although their MSE decrease with respect to the epochs number is not as fast as for the RNNs with $\boxed{\text{2Q}}$ and $\boxed{\text{4Fw}}$, they have nearly 50% less learnable parameters.

The training performance of the RNNs using the mixed non-linear transition blocks, either $\boxed{\text{Q}} \rightarrow \boxed{\text{Fw}}$ or $\boxed{\text{Fw}} \rightarrow \boxed{\text{Q}}$, is also studied and the evolution of their MSE during the training is plotted in Fig. B.26(a).

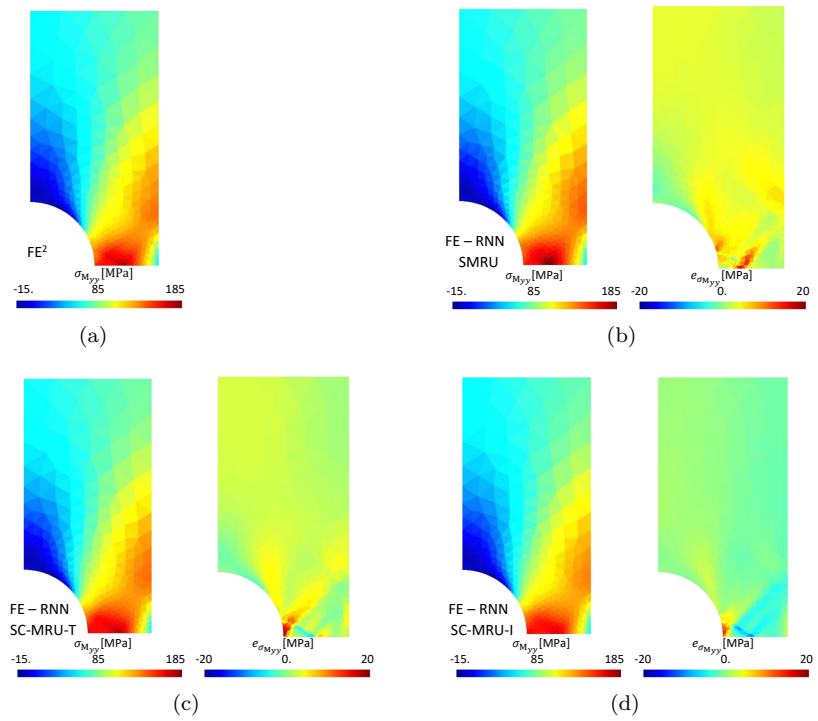


Figure 23: Comparisons of the Cauchy stress, $\sigma_{M_{YY}}$, distribution of an open-hole sample obtained, at loading point "D", (a) by the FE^2 method; and using the RNN surrogate model with the (b) SMRU; (c) SC-MRU-T and (d) SC-MRU-I as recurrent units; The loading point "D" refers to the marker on Figs. 17(b)-17(d). For the surrogate-based simulations, the right figure depicts the absolute error $e_{\sigma_{M_{YY}}}$ with respect to the FE^2 method.

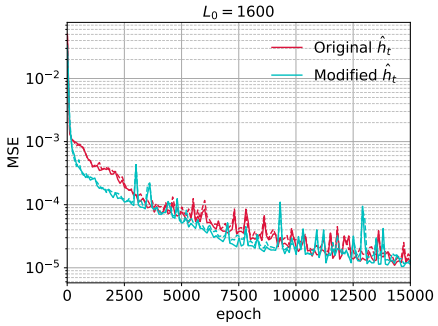
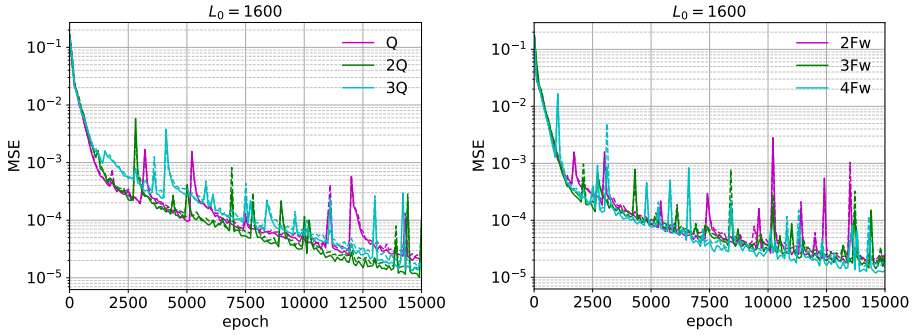


Figure A.24: MSE vs. training epochs of RNNs with MGRU, where the update hidden variables, $\hat{\boldsymbol{h}}_t$, are successively computed by Eq. (17) and (A.1).

Since the RNNs with these two blocks have comparable amount of learnable parameters, see Table B.5, Fig. B.26(a) indicates that the better training performance is obtained by the RNN using $\boxed{\text{Q}} \rightarrow \boxed{\text{Fw}}$ as a non-linear transition block. In Fig. 26(b), we compare the MSE evolution of the RNN using either $\boxed{\text{Q}} \rightarrow \boxed{\text{Fw}}$ or $\boxed{\text{2Q}}$ as non-linear transition blocks; the latter has previously been shown to exhibit a fast decrease of the MSE during its training, see Fig. B.25. A similar MSE decrease with the epochs number can be seen in Fig. B.26(b), although the RNN using $\boxed{\text{Q}} \rightarrow \boxed{\text{Fw}}$ has nearly 20% less learnable parameters than when using $\boxed{\text{2Q}}$ as a non-linear transition block, see Table B.5.

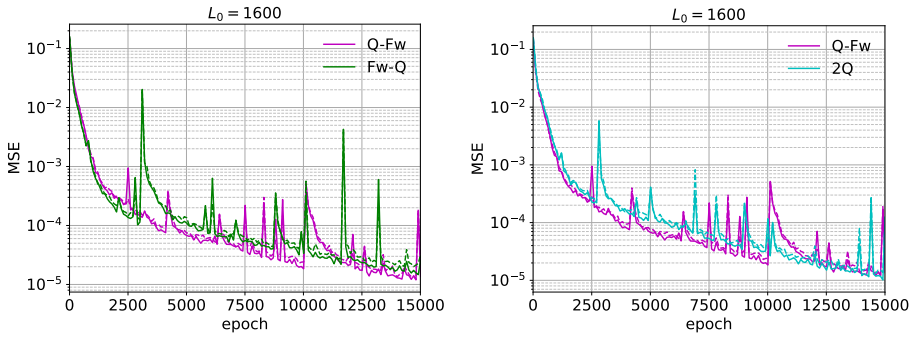
Table B.5: The numbers of learnable parameters of the RNNs using the SC-MRU-I recurrent unit with different non-linear transition blocks.

Non-linear transition block	number of hidden variable $\hbar = 120$		
Quadratic	Q	2Q	3Q
layer	59644	88684	117724
Feed-forward	2Fw	3Fw	4Fw
	59284	73804	88324
Mixed	Q-Fw	Fw-Q	
	74164	73804	



(a) (b)

Figure B.25: The evolution of the MSE vs. training epoch, in "solid line" for the training data and in "dashed line" for the testing data, for different non-linear transition blocks of the SC-MRU-I recurrent unit: (a) $\boxed{n\text{Q}}$; and (b) $\boxed{n\text{Fw}}$. The number of hidden variables is $\hbar = 120$.



(a) (b)

Figure B.26: The evolution of the MSE vs. training epoch, in "solid line" for the training data and in "dashed line" for the testing data, for the SC-MRU-I using: (a) The mixed $\boxed{\text{Q}} \rightarrow \boxed{\text{Fw}}$ and $\boxed{\text{Fw}} \rightarrow \boxed{\text{Q}}$ non-linear transition blocks; and (b) The $\boxed{\text{Q}} \rightarrow \boxed{\text{Fw}}$ and $\boxed{\text{2Q}}$ non-linear transition blocks. The number of hidden variables is $\hbar = 120$.

References

[1] T. Furukawa, G. Yagawa, Implicit constitutive modelling for viscoplasticity using neural networks, International Journal for Numerical Methods in Engineering 43 (1998) 195–219.

[2] M. Stoffel, F. Bamer, B. Markert, Neural network based constitutive modeling of nonlinear viscoplastic structural response, Mechanics Research Communications 95 (2019) 85 – 88. URL: http://www.sciencedirect.com/science/article/pii/S0093641318305822. doi:https://doi.org/10.1016/j.mechrescom.2019.01.004.

[3] T. Furukawa, M. Hoffman, Accurate cyclic plastic analysis using a neural network material model, Engineering Analysis with Boundary Elements 28 (2004) 195 – 204. URL: http://www.sciencedirect.com/science/article/pii/S095579970300050X. doi:https://doi.org/10.1016/S0955-7997(03)00050-X, inverse Problems.

[4] K. Wang, W. Sun, Meta-modeling game for deriving theory-consistent, microstructure-based traction–separation laws via deep reinforcement learning, Computer Methods in Applied Mechanics and Engineering 346 (2019) 216 – 241. URL: http://www.sciencedirect.com/science/article/pii/S0045782518305851. doi:https://doi.org/10.1016/j.cma.2018.11.026.

[5] M. Fernández, S. Rezaei, J. R. Mianroodi, F. Fritzen, S. Reese, Application of artificial neural networks for the prediction of interface mechanics: a study on grain boundary constitutive behavior, Advanced Modeling and Simulation in Engineering Sciences 7 (2020) 1–27.

[6] M. Lefik, B. Schrefler, Artificial neural network as an incremental non-linear constitutive model for a finite element code, Computer Methods in Applied Mechanics and Engineering 192 (2003) 3265 – 3283. URL: http://www.sciencedirect.com/science/article/pii/S0045782503003505. doi:https://doi.org/10.1016/S0045-7825(03)00350-5, multiscale Computational Mechanics for Materials and Structures.

[7] Y. M. A. Hashash, S. Jung, J. Ghaboussi, Numerical implementation of a neural network based material model in finite element analysis, International Journal for Numerical Methods in Engineering 59 (2004) 989–1005. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.905. doi:10.1002/nme.905. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.905.

[8] S. Jung, J. Ghaboussi, Characterizing rate-dependent material behaviors in self-learning simulation, Computer Methods in Applied Mechanics and Engineering 196 (2006) 608 – 619. URL: http://www.sciencedirect.com/science/article/pii/S0045782506001940. doi:https://doi.org/10.1016/j.cma.2006.06.006.

[9] M. Lefik, B. Schrefler, Artificial neural network for parameter identifications for an elasto-plastic model of superconducting cable under cyclic loading, Computers & Structures 80 (2002) 1699 – 1713. URL: http://www.sciencedirect.com/science/article/pii/S0045794902001621. doi:https://doi.org/10.1016/S0045-7949(02)00162-1.

[10] L. Wu, K. Zulueta, Z. Major, A. Arriaga, L. Noels, Bayesian inference of non-linear multiscale model parameters accelerated by a deep neural network, Computer Methods in Applied Mechanics and Engineering 360 (2020) 112693. URL: http://www.sciencedirect.com/science/article/pii/S004578251930578X. doi:https://doi.org/10.1016/j.cma.2019.112693.

[11] L. Wu, L. Noels, Recurrent neural networks (rnns) with dimensionality reduction and break down in computational mechanics; application to multi-scale localization step, Computer Methods in Applied Mechanics and Engineering 390 (2022) 114476. URL: https://www.sciencedirect.com/science/article/pii/S0045782521006940. doi:https://doi.org/10.1016/j.cma.2021.114476.

[12] F. Ghavamian, A. Simone, Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network, Computer Methods in Applied Mechanics and Engineering 357 (2019) 112594. URL: http://www.sciencedirect.com/science/article/pii/S0045782519304700. doi:https://doi.org/10.1016/j.cma.2019.112594.

[13] H. J. Logarzo, G. Capuano, J. J. Rimoli, Smart constitutive laws: Inelastic homogenization through machine learning, Computer Methods in Applied Mechanics and Engineering 373 (2021) 113482. URL: https://www.sciencedirect.com/science/article/pii/S0045782520306678. doi:https://doi.org/10.1016/j.cma.2020.113482.

[14] B. Bahtiri, B. Arash, S. Scheffler, M. Jux, R. Rolfes, A machine learning-based viscoelastic–viscoplastic model for epoxy nanocomposites with moisture content, Computer Methods in Applied Mechanics and Engineering 415 (2023) 116293. URL: https://www.sciencedirect.com/science/article/pii/S0045782523004176. doi:https://doi.org/10.1016/j.cma.2023.116293.

[15] Q. Guan, Z. Yang, N. Guo, Z. Hu, Finite element geotechnical analysis incorporating deep learning-based soil model, Computers and Geotechnics 154 (2023) 105120. URL: https://www.sciencedirect.com/science/article/pii/S0266352X22004578. doi:https://doi.org/10.1016/j.compgeo.2022.105120.

[16] M. Mozaffar, R. Bostanabad, W. Chen, K. Ehmann, J. Cao, M. A. Bessa, Deep learning predicts path-dependent plasticity, Proceedings of the National Academy of Sciences 116 (2019) 26414–26420. doi:10.1073/pnas.1911815116. arXiv:https://www.pnas.org/content/116/52/26414.full.pdf.

[17] M. B. Gorji, M. Mozaffar, J. N. Heidenreich, J. Cao, D. Mohr, On the potential of recurrent neural networks for modeling path dependent plasticity, Journal of the Mechanics and Physics of Solids 143 (2020) 103972. doi:https://doi.org/10.1016/j.jmps.2020.103972.

[18] L. Wu, V.-D. Nguyen, N. G. Kilingar, L. Noels, A recurrent neural network-accelerated multi-scale model for elasto-plastic heterogeneous materials subjected to random cyclic and non-proportional loading paths, Computer Methods in Applied Mechanics and Engineering 369 (2020) 113234. doi:10.1016/j.cma.2020.113234.

[19] J. Friemann, B. Dashtbozorg, M. Fagerström, S. M. Mirkhalaf, A micromechanics-based recurrent neural networks model for path-dependent cyclic deformation of short fiber composites, International Journal for Numerical Methods in Engineering 124 (2023) 2292–2314. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.7211. doi:https://doi.org/10.1002/nme.7211. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.7211.

[20] J. García-Ávila, D. d. J. Torres Serrato, C. A. Rodriguez, A. V. Martínez, E. R. Cedillo, J. I. Martínez-López, Predictive modeling of soft stretchable nanocomposites using recurrent neural networks, Polymers 14 (2022). URL: https://www.mdpi.com/2073-4360/14/23/5290. doi:10.3390/polym14235290.

[21] S. Deng, S. Hosseinmardi, D. Apelian, R. Bostanabad, Deep learning for multiscale damage analysis via physics-informed recurrent neural network, 2022. arXiv:2212.01880.

[22] S. Im, J. Lee, M. Cho, Surrogate modeling of elasto-plastic problems via long short-term memory neural networks and proper orthogonal decomposition, Computer Methods in Applied Mechanics and Engineering 385 (2021) 114030. URL: https://www.sciencedirect.com/science/article/pii/S0045782521003613. doi:https://doi.org/

10.1016/j.cma.2021.114030.
[23] S. Vijayaraghavan, L. Wu, L. Noels, S. Bordas, S. Natarajan, L. Beex, A data-driven reduced-order surrogate model for entire elastoplastic simulations applied to representative volume elements, Scientific Reports 13 (2023) 12781. doi:https://doi.org/10.1038/s41598-023-38104-x.
[24] S. Fetni, T. Q. D. Pham, T. V. Hoang, H. S. Tran, L. Duchêne, X.-V. Tran, A. M. Habraken, Capabilities of auto-encoders and principal component analysis of the reduction of microstructural images; application on the acceleration of phase-field simulations, Computational Materials Science 216 (2023) 111820. URL: https://www.sciencedirect.com/science/article/pii/S0927025622005316. doi:https://doi.org/10.1016/j.commatsci.2022.111820.
[25] S. B. Tandale, M. Stoffel, Recurrent and convolutional neural networks in structural dynamics: a modified attention steered encoder–decoder architecture versus lstm versus gru versus tcn topologies to predict the response of shock wave-loaded plates, Computational Mechanics 72 (2023) 765–786. doi:https://doi.org/10.1007/s00466-023-02317-8.
[26] I. Rocha, P. Kerfriden, F. P. van der Meer, Micromechanics-based surrogate models for the response of composites: A critical comparison between a classical mesoscale constitutive model, hyper-reduction and neural networks, European Journal of Mechanics - A/Solids 82 (2020) 103995. doi:10.1016/j.euromechsol.2020.103995.
[27] F. Kibrete, T. Trzepieciński, H. S. Gebremedhen, D. E. Woldemichael, Artificial intelligence in predicting mechanical properties of composite materials, Journal of Composites Science 7 (2023). URL: https://www.mdpi.com/2504-477X/7/9/364. doi:10.3390/jcs7090364.
[28] J. Dornheim, L. Morand, H. J. Nallani, D. Helm, Neural networks for constitutive modeling: From universal function approximators to advanced models and the integration of physics, Archives of Computational Methods in Engineering (2023). doi:https://doi.org/10.1007/s11831-023-10009-y.
[29] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707. URL: https://www.sciencedirect.com/science/article/pii/S0021999118307125. doi:https://doi.org/10.1016/j.jcp.2018.10.045.
[30] P. Pantidis, M. E. Mobasher, Integrated finite element neural network (i-fenn) for non-local continuum damage mechanics, Computer Methods in Applied Mechanics and Engineering 404 (2023) 115766. URL: https://www.sciencedirect.com/science/article/pii/S0045782522007228. doi:https://doi.org/10.1016/j.cma.2022.115766.
[31] S. B. Tandale, F. Bamer, B. Markert, M. Stoffel, Physics-based self-learning recurrent neural network enhanced time integration scheme for computing viscoplastic structural finite element response, Computer Methods in Applied Mechanics and Engineering 401 (2022) 115668. URL: https://www.sciencedirect.com/science/article/pii/S0045782522006235. doi:https://doi.org/10.1016/j.cma.2022.115668.
[32] Z. Yuan, R. Biswas, L. H. Poh, Accelerated offline setup of homogenized microscopic model for multi-scale analyses using neural network with knowledge transfer, International Journal for Numerical Methods in Engineering 124 (2023) 3063–3086. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.7239. doi:https://doi.org/10.1002/nme.7239. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.7239.
[33] Z. Liu, C. Wu, Exploring the 3d architectures of deep material network in data-driven multiscale mechanics, Journal of the Mechanics and Physics of Solids 127 (2019) 20 – 46. doi:10.1016/j.jmps.2019.03.004.
[34] V.-D. Nguyen, L. Noels, Interaction-based material network: A general framework for (porous) microstructured materials, Computer Methods in Applied Mechanics and Engineering (2021) 114300. URL: https://www.sciencedirect.com/science/article/pii/S0045782521005934. doi:https://doi.org/10.1016/j.cma.2021.114300.
[35] S. Gajek, M. Schneider, T. Böhlke, On the micromechanics of deep material networks, Journal of the Mechanics and Physics of Solids 142 (2020) 103984. doi:10.1016/j.jmps.2020.103984.
[36] L. Wu, L. Adam, L. Noels, Micro-mechanics and data-driven based reduced order models for multi-scale analyses of woven composites, Composite Structures 270 (2021) 114058. URL: https://www.sciencedirect.com/science/article/pii/S0263822321005183. doi:https://doi.org/10.1016/j.compstruct.2021.114058.
[37] F. Masi, I. Stefanou, P. Vannucci, V. Maffi-Berthier, Thermodynamics-based artificial neural networks for constitutive modeling, Journal of the Mechanics and Physics of Solids 147 (2021) 104277. URL: https://www.sciencedirect.com/science/article/pii/S0022509620304841. doi:https://doi.org/10.1016/j.jmps.2020.104277.
[38] F. Masi, I. Stefanou, Multiscale modeling of inelastic materials with thermodynamics-based artificial neural networks (tann), Computer Methods in Applied Mechanics and Engineering 398 (2022) 115190. URL: https://www.sciencedirect.com/science/article/pii/S0045782522003450. doi:https://doi.org/10.1016/j.cma.2022.115190.
[39] M. Maia, I. Rocha, P. Kerfriden, F. van der Meer, Physically recurrent neural networks for path-dependent heterogeneous materials: Embedding constitutive models in a data-driven surrogate, Computer Methods in Applied Mechanics and Engineering 407 (2023) 115934. URL: https://www.sciencedirect.com/science/article/pii/S0045782523000579. doi:https://doi.org/10.1016/j.cma.2023.115934.
[40] C. Bonatti, D. Mohr, On the importance of self-consistency in recurrent neural network models representing elasto-plastic solids, Journal of the Mechanics and Physics of Solids 158 (2022) 104697. URL: https://www.sciencedirect.com/science/article/pii/S0022509621003161. doi:https://doi.org/10.1016/j.jmps.2021.104697.
[41] C. Bonatti, B. Berisha, D. Mohr, From cp-fft to cp-rnn: Recurrent neural network surrogate model of crystal plasticity, International Journal of Plasticity 158 (2022) 103430. URL: https://www.sciencedirect.com/science/article/pii/S074964192200208X. doi:https://doi.org/10.1016/j.ijplas.2022.103430.
[42] F. Feyel, Multiscale fe2 elastoviscoplastic analysis of composite structures, Computational Materials Science 16 (1999) 344 – 354. URL: http://www.sciencedirect.com/science/article/pii/S0927025699000774. doi:http://dx.doi.org/10.1016/S0927-0256(99)00077-4.
[43] C. Miehe, J. Schotte, J. Schröder, Computational micro-macro transitions and overall moduli in the analysis of polycrystals

at large strains, Computational Materials Science 16 (1999) 372–382. Cited By 88.
[44] J. C. Heck, F. M. Salem, Simplified minimal gated unit variations for recurrent neural networks, in: 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), IEEE, 2017, pp. 1593–1596.
[45] D. Peric, E. A. de Souza Neto, R. A. Feijóo, M. Partovi, A. J. C. Molina, On micro-to-macro transitions for multi-scale analysis of non-linear heterogeneous materials: unified variational basis and finite element implementation, International Journal for Numerical Methods in Engineering 87 (2010) 149 – 170. URL: http://dx.doi.org/10.1002/nme.3014.
[46] J. Schröder, M. Labusch, M.-A. Keip, Algorithmic two-scale transition for magneto-electro-mechanically coupled problems: Fe2-scheme: Localization and homogenization, Computer Methods in Applied Mechanics and Engineering 32 (2016) 253–280. URL: http://www.sciencedirect.com/science/article/pii/S0045782515003242. doi:http://dx.doi.org/10.1016/j.cma.2015.10.005.
[47] V.-D. Nguyen, L. Wu, L. Noels, Unified treatment of microscopic boundary conditions and efficient algorithms for estimating tangent operators of the homogenized behavior in the computational homogenization method, Computational Mechanics 59 (2017) 483–505. URL: https://doi.org/10.1007/s00466-016-1358-z. doi:10.1007/s00466-016-1358-z.
[48] G.-B. Zhou, J. Wu, C.-L. Zhang, Z.-H. Zhou, Minimal gated unit for recurrent neural networks, International Journal of Automation and Computing 13 (2016) 226–234.
[49] 2020. URL: https://pytorch.org/, accessed: 2020-04-30.
[50] L. Wu, L. Noels, Data of "self-consistency reinforced minimal gated recurrent unit for surrogate modeling of history-dependent non-linear problems: application to history-dependent homogenized response of heterogeneous materials", 2024. URL: https://gitlab.uliege.be/didearot/didearotPublic/publicationsData/2024_scmru. doi:10.5281/zenodo.10551272.
[51] A. Cuitino, M. Ortiz, A material-independent method for extending stress update algorithms from small-strain plasticity to finite plasticity with multiplicative kinematics, Engineering computations 9 (1992) 437–437.