Daniel *et al. Adv. Model. and Simul. in Eng. Sci.(2020)7:16*
https://doi.org/10.1186/s40323-020-00153-6

Advanced Modeling and Simulation in Engineering Sciences

RESEARCH ARTICLE Open Access

Model order reduction assisted by deep neural networks (ROM-net)

Thomas Daniel[1,2*], Fabien Casenave[1], Nissrine Akkari[1] and David Ryckelynck[2]

*Correspondence: thomas.daniel@safrangroup.com
[1]SafranTech, Rue des Jeunes Bois, Chateaufort, 78114 Magny-les-Hameaux, France
Full list of author information is available at the end of the article

Abstract

In this paper, we propose a general framework for projection-based model order reduction assisted by deep neural networks. The proposed methodology, called *ROM-net*, consists in using deep learning techniques to adapt the reduced-order model to a stochastic input tensor whose nonparametrized variabilities strongly influence the quantities of interest for a given physics problem. In particular, we introduce the concept of *dictionary-based ROM-nets*, where deep neural networks recommend a suitable local reduced-order model from a dictionary. The dictionary of local reduced-order models is constructed from a clustering of simplified simulations enabling the identification of the subspaces in which the solutions evolve for different input tensors. The training examples are represented by points on a Grassmann manifold, on which distances are computed for clustering. This methodology is applied to an anisothermal elastoplastic problem in structural mechanics, where the damage field depends on a random temperature field. When using deep neural networks, the selection of the best reduced-order model for a given thermal loading is 60 times faster than when following the clustering procedure used in the training phase.

Keywords: Model order reduction, Machine learning, Deep neural networks, Nonlinear structural mechanics

Introduction

Numerical simulations in physics have become an essential tool in many engineering domains. The development of high-performance computing has enabled engineers and scientists to use complex models for real-world applications, with ultra-realistic simulations involving millions of degrees of freedom. However, such simulations are too time-consuming to be integrated in design iterations in the industry. They are usually limited to the final validation and certification steps, while the design process still relies on simplified models. Accelerating these complex simulations is a key challenge, as it would provide useful numerical tools to improve design processes. The development of numerical methods for fast simulations would also enable using new models that have not been applied to industrial problems yet, because of their complexities. Uncertainty quantification is another important example of analysis that would become practicable if the cost of simulations was sufficiently reduced. Indeed, quantities of interest monitored

in numerical simulations depend on the environment of the physical system, which is usually not exactly known. In some cases, these uncertainties strongly influence simulation results, and the probability distributions of the quantities of interest must be estimated in order to ensure the reliability of the industrial product.

The cost of numerical simulations can be reduced by *projection-based model order reduction*, which consists in restricting the search of the solution to a low-dimensional space spanned by a reduced-order basis. This reduced-order basis is inferred from a set of pre-computed high-fidelity solutions, using linear dimensionality reduction techniques such as the proper orthogonal decomposition (POD, [1–3]). In addition to this reduction in terms of degrees of freedom, a second reduction stage may be necessary for nonlinear problems, this time in terms of integration points. This is called *hyperreduction*, or *operator compression* according to the terminology introduced in [4]. Operator compression methods include the empirical interpolation method (EIM, [5]), the missing point estimation (MPE, [6]), the a priori hyperreduction (APHR, [7]), the best point interpolation method (BPIM, [8]), the discrete empirical interpolation method (DEIM, [9]), the Gauss-Newton with approximated tensors (GNAT, [10]), the enery-conserving sampling and weighting (ECSW, [11]), the empirical cubature method (ECM, [12]), and the linear program empirical quadrature procedure (LPEQP, [13]).

In this article, we consider uncertainties on a tensorial input variable which affects the solution of the partial differential equations (PDEs) governing the physical system. This input tensor can represent a three-dimensional field, physical constants used in the constitutive equations, images of defects, a X-ray computed tomography scan characterizing a microstructure, boundary conditions, or geometrical details. In the example presented at the end of this paper, the input tensor represents a three-dimensional temperature field influencing the physical properties of the system. The objective is to accelerate numerical simulations where a quantity of interest highly depends on this stochastic input tensor subjected to *nonparametrized* variabilities. This objective can be achieved with a single reduced-order model, as long as uncertainties on the input tensor are small enough or have a minor impact on the quantity of interest. In other situations, the solution of the governing PDEs lies in a manifold which cannot be covered by a single reduced-order model without increasing its dimension and thus degrading its efficiency. For example, traditional model order reduction techniques fail to solve advection-dominated problems. These problems require more sophisticated techniques, such as those proposed in [14] or [15]. In structural mechanics, the fatigue lifetime assessment of high-pressure turbine blades of an aircraft engine is very sensitive to variations of the temperature field, see [16]. Linear dimensionality reduction is not always suitable for this kind of applications. Nonetheless, linear methods have the critical advantage of being compatible with the Galerkin method, providing a reduced problem in the form of equations assembled on a reduced-order basis.

Many strategies have been proposed to address such problems. The concept of *local* reduced-order models was first introduced in [17], and applied to computational fluid dynamics in [18]. In these works, the set of pre-computed high-fidelity solutions was partitioned into several clusters, each of them being used to build a small cluster-specific reduced-order model. The resulting *dictionary* of local reduced-order models was then used to adapt the reduced-order basis to the current state of the solution by finding the closest cluster center. This technique works very well when the solution evolves on a

low-dimensional manifold. However, for some specific applications where there is no guarantee that the solution lies in a low-dimensional manifold, this technique might be subjected to the curse of dimensionality [19]. Indeed, in high dimension, the nearest neighbour is almost as far as the furthest point in the dataset because of the loss of contrast in distances, explaining the difficulties of high-dimensional clustering. Other approaches rely on the interpolation of reduced-order models. This trend was initiated by the subspace angle interpolation method [20–24]. A generalization of this method has been proposed in [25] with the ROM adaptation method based on the *Grassmann manifold* (also called *Grassmannian*). It has been successfully applied to the CFD-based aeroelastic analysis of a complete aircraft configuration for varying Mach number and angle of attack. In [26], this method has been improved to achieve real-time performance for linear problems. More recent works [27,28] propose other interpolation methods on Grassmannians. All these interpolation methods give excellent results for problems depending on a small number of parameters, but none of them have yet been applied to nonparametrized variabilities of a large input tensor.

The generic methodology developed in this article, called *ROM-net*, is another attempt to deal with the limits of traditional tools available in the model order reduction literature. It relies on projection-based model order reduction assisted by *deep learning* techniques. Our objective is to define a general framework for reduced-order model adaptation using deep neural networks, in order to see to what extent model order reduction can benefit from the recent advances in deep learning. Indeed, the growing interest for this discipline has led to the development of innovative methods in many fields. These advances have facilitated the development of surrogate models and data-driven approaches in physics, providing approximate solutions in real time. Other modeling strategies are based on a hybrid approach, mixing physics-based modeling and machine learning. For example, deep neural networks were used in [29] to model the Reynolds stress anisotropy tensor in Reynolds Averaged Navier Stokes (RANS) models, in computational fluid dynamics. In [30], a nonlinear dimensionality reduction is performed using deep convolutional autoencoders. The modeling strategy presented in [31] was the first hybrid approach involving both a dictionary of local hyperreduced-order models and computer vision techniques. In the context of image-based modeling, it showed that convolutional neural networks could be used to recognize the loading case of a mechanical experiment on a digital image, and select a suitable hyperreduced-order model to simulate the experiment.

The concept of ROM-net introduced in this article is an extension of the methodology presented in the aforementioned paper, designed to accelerate numerical simulations where a quantity of interest depends on a stochastic tensorial input. Dictionary-based ROM-nets can be made of one or several deep neural networks selecting the best reduced-order model from a dictionary. In our case, unlike the strategy presented in [31], this dictionary derives from the clustering of outputs of simplified simulations using distances on a Grassmannian. In the first section of this article, we introduce the formal definitions of ROM-nets and dictionary-based ROM-nets. Then, we describe the training procedure for dictionary-based ROM-nets. An application to a temperature-dependent problem in structural mechanics is presented in the last section of this paper, where we focus on the clustering procedure and the construction of a classifier for model selection.

ROM-nets and dictionary-based ROM-nets

Classical projection-based model order reduction

Let us consider a physics problem whose primal variable of the governing equations is denoted by $\mathbf{u}$ and defined on a domain $\Omega \subset \mathbb{R}^{\beta}$, $\beta \in [\![1;3]\!]$ and on a (normalized) time interval $[0;1]$. The governing PDEs are generally posed on an infinite-dimensional Hilbert space, but in practice, these equations are solved numerically on a finite-dimensional subspace, denoted by $\mathcal{H}$ in this article. In solid mechanics for example, $\mathcal{H}$ is the space spanned by finite-element shape functions $\left\{\boldsymbol{\phi}_i\right\}_{1\leq i\leq \dim(\mathcal{H})}$, and the primal variable $\mathbf{u}$ corresponds to the displacement field. The primal variable $\mathbf{u}$ computed at the n-th time step can be represented by a vector $\mathbf{U}_n \in \mathbb{R}^{\dim(\mathcal{H})}$ containing its coordinates in the finite-element basis $\left\{\boldsymbol{\phi}_i\right\}_{1\leq i\leq \dim(\mathcal{H})}$. In solid mechanics, this numerical solution can be obtained with the Newton–Raphson algorithm, an iterative procedure based on the linearization of the virtual work principle. The resulting linear system to be solved for the m-th iteration at the n-th time increment reads:

$$\mathbf{J}_n^{(m)}\delta\mathbf{U}_n^{(m)} = -\mathbf{R}_n^{(m)} \tag{1}$$

where $\mathbf{J}_n^{(m)} \in \mathbb{R}^{\dim(\mathcal{H})\times\dim(\mathcal{H})}$ is the Jacobian matrix, also called (global) tangent stiffness matrix, $\mathbf{R}_n^{(m)} \in \mathbb{R}^{\dim(\mathcal{H})}$ is the vector of residuals, and $\delta\mathbf{U}_n^{(m)} \in \mathbb{R}^{\dim(\mathcal{H})}$ is the correction applied to the vector of increments of the primal variable defined by:

$$\Delta\mathbf{U}_n^{(m)} = \Delta\mathbf{U}_n^{(m-1)} + \delta\mathbf{U}_n^{(m)} \tag{2}$$

with $\Delta\mathbf{U}_n^{(0)} = \mathbf{0}$. When the convergence criterion $||\mathbf{R}_n^{(m)}|| \leq \epsilon_{NR}||\mathbf{F}_n^{\text{ext}}||$ is satisfied for a given $m = m^*$ with ϵ_{NR} being the tolerance of the Newton–Raphson algorithm and $\mathbf{F}_n^{\text{ext}}$ being the vector of external forces, the solution at the n-th time increment is defined as:

$$\mathbf{U}_n = \mathbf{U}_{n-1} + \Delta\mathbf{U}_n^{(m^*)} \tag{3}$$

Equation (1) is the high-dimensional linear system deriving from the high-fidelity model composed of equilibrium, compatibility and constitutive equations.

Projection-based model order reduction consists in searching an approximation of the high-fidelity solution in a low-dimensional subspace $\mathcal{V}_{ROM} \subset \mathcal{H}$ adapted to the current physics problem. This subspace is spanned by an appropriate reduced-order basis $\left\{\boldsymbol{\psi}_i\right\}_{1\leq i\leq N}$, with N being very small compared to $\dim(\mathcal{H})$. The reduced-order basis approximation of the primal variable reads:

$$\forall t \in [0;1], \qquad \forall \mathbf{x} \in \Omega, \qquad \mathbf{u}(\mathbf{x},t) = \sum_{i=1}^{N} \gamma_i(t)\boldsymbol{\psi}_i(\mathbf{x}) \tag{4}$$

where $\{\gamma_i\}_{1\leq i\leq N}$ are the reduced coordinates which can be stored in a vector $\boldsymbol{\gamma} \in \mathbb{R}^N$. The coordinates of the modes $\left\{\boldsymbol{\psi}_i\right\}_{1\leq i\leq N}$ in the finite-element basis $\left\{\boldsymbol{\phi}_i\right\}_{1\leq i\leq \dim(\mathcal{H})}$ are stored in columns in a matrix $\mathbf{V} \in \mathbb{R}^{\dim(\mathcal{H})\times N}$ called reduction matrix. Hence:

$$\forall i \in [\![1;N]\!], \qquad \forall \mathbf{x} \in \Omega, \qquad \boldsymbol{\psi}_i(\mathbf{x}) = \sum_{j=1}^{\dim(\mathcal{H})} V_{ji}\boldsymbol{\phi}_j(\mathbf{x}) \tag{5}$$

These modes can be obtained by applying the POD [1] or the snapshot POD [2,3] to a set of pre-computed high-fidelity solutions evaluated at different time steps or for different configurations of the physical system. After the Galerkin projection of the governing equations on $\mathcal{V}_{ROM}$, the reduced linear system to solve at each iteration of the Newton–Raphson algorithm in the reduced-order model (ROM) is then:

$$\mathbf{V}^T\mathbf{J}_n^{(m)}\mathbf{V}\delta\boldsymbol{\gamma}_n^{(m)} = -\mathbf{V}^T\mathbf{R}_n^{(m)} \tag{6}$$

Any ordered set of $k \leq \dim(\mathcal{H})$ linearly independent vectors in $\mathcal{H}$ is called a *k-frame*. The *Stiefel manifold* $V(k, \mathcal{H})$ is the set of all orthonormal k-frames in $\mathcal{H}$. In projection-based model order reduction, reduced-order bases obtained by POD or snapshot POD belong to a Stiefel manifold. In this article, the notation $V(\mathcal{H})$ stands for the set of reduced-order bases:

$$V(\mathcal{H}) = \bigcup_{i=1}^{\dim(\mathcal{H})} V(i, \mathcal{H}) \tag{7}$$

The linear system (6) results from a first reduction stage in terms of degrees of freedom. For some nonlinear problems, a second reduction stage is required to efficiently decrease the computation time. This second reduction stage is referred to as *hyperreduction* or *operator compression*, as mentioned in the introduction. In this case, the definition of the set $V(\mathcal{H})$ can be extended to include sets of hyperreduction parameters in addition to the set of reduced-order bases.

ROM-nets

Our objective is to predict a quantity of interest Y via the computation of a primal variable $\mathbf{u}$ that belongs to a reduced approximation space and satisfies nonlinear physics equations depending on a stochastic input tensor X. In this article, $\mathcal{X}$ denotes the set of input variabilities and $\mathcal{Y}$ represents the set containing the quantity of interest. In structural mechanics, Y can represent a damage field, the von Mises stress in a zone of interest, or the displacement of a specific point in the structure, while X can stand for material constants, boundary conditions, geometrical parameters, a X-ray computed tomography scan characterizing the microstructure, images of defects, or even a three-dimensional field defined on the domain Ω such as a temperature field, residual stresses, or heterogeneous material parameters. The only restriction on the input is that it must have a tensorial representation, that is, a representation as a multidimensional array. For instance, images are second-order tensors or two-dimensional arrays, X-ray computed tomography scans are third-order tensors or three-dimensional arrays, and fields discretized on a finite-element mesh can be represented by first-order tensors or one-dimensional arrays. These tensorial inputs are stochastic because they contain the uncertainties on the physical system under study: when considering polycrystalline materials, X-ray computed tomography scans could be used to study macroscopic properties under microstructural variabilities such as grains' sizes, shapes and orientations. We refer the reader to [32,33] for more details on finite-element modeling based on X-ray computed tomography scans. In the application presented at the end of this paper, X is the finite-element discretization of a temperature field with variabilities evolving in $L^2(\Omega)$. These stochastic variabilities may be related to turbulence in a fluid-structure interaction with a high-Reynolds-number fluid flow. In aircraft engines, the temperature field in high-pressure turbine blades results from a complex turbulent flow coming from the combustion chamber. While the tensor X can be generated by a parametric stochastic model, it is assumed that we have no prior knowledge of the underlying model. Therefore, the proposed methodology is suitable for *nonparametrized* (or *generic*) input variabilities which can represent uncertainties on the environment of the physics problem. This feature is required when the method is trained on data simulated by a parametric model, but applied to real data with unknown distributions obtained from experimental measures or from a more complex model.

When the input $X \in \mathcal{X}$ is modified, the primal variable $\mathbf{u}$ evolves on a manifold $\mathcal{M}$. In some situations, it is complicated to build a relevant reduced-order model giving accurate predictions for the primal variable on the whole manifold. In such cases, predictions on the quantity of interest Y are inaccurate since they derive from the behavior of the primal variable. The reduced-order model must be adapted to the input to capture nonlinearities. In this paper, we propose a general framework for reduced-order model adaptation via deep learning algorithms.

Given two sets $\mathcal{A}$ and $\mathcal{B}$, the notation $\mathcal{B}^{\mathcal{A}}$ represents the set of functions $f : \mathcal{A} \to \mathcal{B}$. Let us now give the definitions of a *reduced-order solver* and a *ROM-net*:

Definition 1 *(Reduced-order solver)* Let us consider a physics problem, where a quantity of interest $Y \in \mathcal{Y}$ depends on a tensorial input variable $X \in \mathcal{X}$. A *reduced-order solver* is an operator $\mathcal{S} : V(\mathcal{H}) \to \mathcal{Y}^{\mathcal{X}}$ taking a reduced-order model $m \in V(\mathcal{H})$ as an input and returning a predictor $\mathcal{S}[m] : \mathcal{X} \to \mathcal{Y}$ for the quantity of interest. Given $X \in \mathcal{X}$, the quantity of interest Y can be approximated by:

$$\tilde{Y} := \mathcal{S}[m](X) \tag{8}$$

□

In this definition, the reduced-order model m consists in a reduced-order basis and, optionally, parameters related to a hyperreduction algorithm. The function $\mathcal{S}[m]$ can be seen as an operator solving the reduced linear system (6) and computing the quantity of interest associated to the reduced-order solution $\mathbf{u}$.

Definition 2 *(ROM-net)* Let us consider a physics problem, where a quantity of interest $Y \in \mathcal{Y}$ depends on a tensorial input variable $X \in \mathcal{X}$ and can be predicted by a reduced-order solver $\mathcal{S} : V(\mathcal{H}) \to \mathcal{Y}^{\mathcal{X}}$. A *ROM-net* $\mathcal{R} : \mathcal{X} \to V(\mathcal{H})$ is a deep learning algorithm returning a reduced-order model $\mathcal{R}(X) \in V(\mathcal{H})$ adapted to the input $X \in \mathcal{X}$. Given $X \in \mathcal{X}$, the quantity of interest Y can be approximated by:

$$\tilde{Y} := \mathcal{S}[\mathcal{R}(X)](X) \tag{9}$$

□

Contrary to surrogate modeling, using a reduced-order model $\mathcal{R}(X)$ enables satisfying homogeneous Dirichlet boundary conditions and solving the constitutive equations at least at some specific points if operator compression (hyperreduction) is used. Hence, a ROM-net provides a hybrid approach mixing physics-based modeling and deep learning. It is used as a reduced-order basis generator for complex problems where the reduced-order basis must be adapted to a tensorial input. In addition, it is noteworthy that the definition of the quantity of interest remains quite flexible after the training of a ROM-net. In solid mechanics for instance, the definition of the damage indicator of an uncoupled damage model can be changed without restarting the training phase. Figure 1 summarizes the concept of ROM-nets.

Remark In [30], another deep learning strategy for model order reduction is proposed for parametrized ordinary differential equations. The governing equations are mapped onto a nonlinear manifold thanks to a deep convolutional autoencoder. Contrary to projection-based model order reduction methods using linear dimensionality reduction techniques

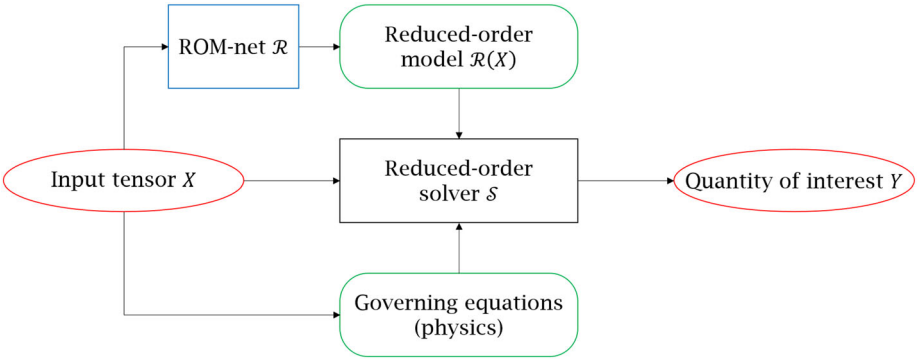

Fig. 1 Flowchart of model order reduction using a ROM-net

such as the POD or the snapshot POD, this methodology performs a nonlinear dimensionality reduction. The reduced (or *generalized*) coordinates in the latent space defined by the autoencoder's bottleneck layer are combined by the decoder in a nonlinear fashion to get the high-dimensional state approximation. When the decoder is linear, this methodology is equivalent to classical projection-based model order reduction. In the present paper, instead of looking for an approximate solution on a nonlinear trial manifold, ROM-nets adapt the linear subspace to the input variability. As explained in the next section, dictionary-based ROM-nets use several subspaces to get a piecewise linear approximation space, while the aforementioned methodology would have approximated the solution manifold by a single nonlinear manifold. The choice of keeping a linear method for dimensionality reduction is motivated by its compatibility with the Galerkin method, enabling an easy construction of a hyperreduced problem. □

Dictionary-based ROM-nets

When the solution manifold $\mathcal{M}$ is embedded in a low-dimensional vector space, one can construct a single global reduced-order model in order to compute approximate solutions of the physics problem for different input variabilities. When the solution manifold $\mathcal{M}$ is not embedded in a low-dimensional vector space, using one single global reduced-order model would result in either time-consuming or inaccurate reduced simulations, depending on the number of modes selected in the reduced-order basis. By partitioning the set $\mathcal{X}$ of input variabilities, one can define a *dictionary* of local reduced-order models which enables approximating $\mathcal{M}$ by several affine subspaces. Clustering algorithms can be used to split the set $\mathcal{X}$ into distinct subsets called *clusters*. Inputs belonging to the same cluster lead to solutions which can be predicted with the same local reduced-order model because of their proximity on the manifold $\mathcal{M}$. More precisely, for a given integer $K \in \mathbb{N}^*$, the clustering algorithm gives a partition of the set $\mathcal{X}$:

$$\mathcal{X} = \bigcup_{k=1}^{K} \mathcal{X}_k \tag{10}$$

$$\forall k \in [\![1;K]\!], \quad \mathcal{X}_k \neq \emptyset \tag{11}$$

$$\forall (i,j) \in [\![1;K]\!]^2, \quad i \neq j \implies \mathcal{X}_i \cap \mathcal{X}_j = \emptyset \tag{12}$$

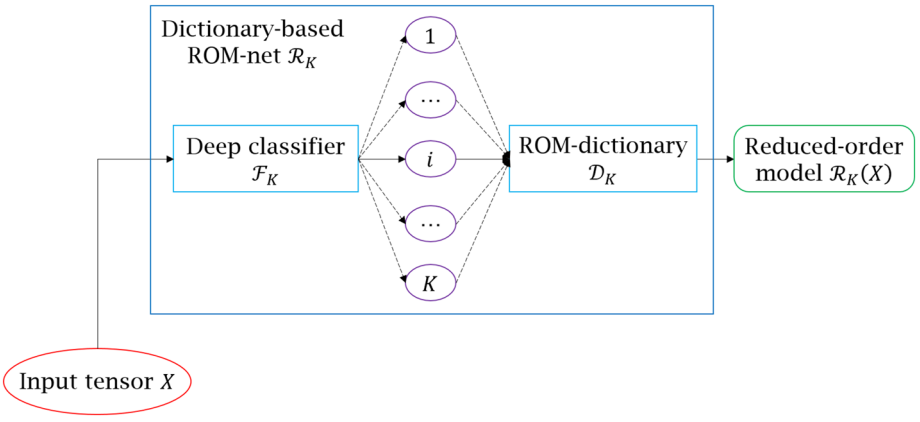

Fig. 2 Flowchart of a dictionary-based ROM-net

The dictionary of local reduced-order models contains K cluster-specific reduced-order models. Hence, for a given input $X \in \mathcal{X}$, one must identify the corresponding cluster $\mathcal{X}_k$ to select the most appropriate reduced-order model.

Definition 3 *(Dictionary of reduced-order models)* Given an integer $K \in \mathbb{N}^*$, an injective function $\mathcal{D}_K : [\![1;K]\!] \to V(\mathcal{H})$ is called a *dictionary of reduced-order models* of dimension K, or *K-ROM-dictionary*. □

Definition 4 *(Dictionary-based ROM-net)* Let us consider a physics problem, where a quantity of interest $Y \in \mathcal{Y}$ depends on a tensorial input variable $X \in \mathcal{X}$ and can be predicted by a reduced-order solver $\mathcal{S} : V(\mathcal{H}) \to \mathcal{Y}^{\mathcal{X}}$. Given an integer $K \in \mathbb{N}^*$, a ROM-net $\mathcal{R}_K$ is a *dictionary-based ROM-net* if there exist a deep classifier $\mathcal{F}_K : \mathcal{X} \to [\![1;K]\!]$ and a K-ROM-dictionary $\mathcal{D}_K : [\![1;K]\!] \to V(\mathcal{H})$ satisfying:

$$\forall X \in \mathcal{X}, \qquad \mathcal{R}_K(X) = \mathcal{D}_K \circ \mathcal{F}_K(X) \tag{13}$$

□

Figure 2 illustrates the concept of dictionary-based ROM-nets. The strategy presented in [31] for image-based modeling using convolutional neural networks and a dictionary of local reduced-order models fits the definition of a dictionary-based ROM-net. In this definition, the expression *deep classifier* denotes deep neural networks returning a single class label in $[\![1;K]\!]$ for a given tensorial input. In multiclass classification, deep classifiers usually have a softmax activation function in the output layer, giving an output vector $\mathbf{y}^{\text{pred}} \in \mathbb{R}^K$ such that y_k^{pred} is the probability for the input tensor to belong to the k-th class. The probabilities y_k^{pred} are also called *membership probabilities*. The deep classifier returns the integer corresponding to the class with the highest membership probability, that is:

$$\forall X \in \mathcal{X}, \qquad \mathcal{F}_K(X) = \underset{k \in [\![1;K]\!]}{\arg\max}(y_k^{\text{pred}}(X)) \tag{14}$$

Such classifiers are called *classical deep classifiers* in this article. The concept of deep classifier used in the definition of a dictionary-based ROM-net can be extended to include not only the classical ones, but also *deep clustering algorithms* [34–36]. These algorithms

use encoders to cluster the data in a low-dimensional latent space, avoiding the difficulties of high-dimensional clustering [19]

As mentioned earlier, the partition of the set $\mathcal{X}$ used to define cluster-specific reduced-order models is given by a clustering algorithm. Clustering algorithms generally rely on a *dissimilarity measure* quantifying the difference between two points in the dataset. In this paper, the expression *dissimilarity measure* refers to a *pseudo-semimetric*:

Definition 5 *(Dissimilarity measure)* A *dissimilarity measure* on $\mathcal{X}$ is a function $\delta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ such that $\delta(X, X') = \delta(X', X)$ for all $(X, X') \in \mathcal{X}^2$ and $\delta(X, X) = 0$ for all $X \in \mathcal{X}$. □

Dictionary-based ROM-nets involving classical deep classifiers

The rest of the article focuses on dictionary-based ROM-nets involving a classical deep classifier. In this case, the deep classifier $\mathcal{F}_K$ solves a classical multiclass classification problem to recommend a suitable reduced-order model from the dictionary. Nevertheless, as the classes are given by a clustering algorithm, one could wonder why a deep neural network is used for cluster assignment. When using a center-based clustering algorithm, each cluster $\mathcal{X}_k$ is represented by a center $\tilde{X}_k$. In theory, one could compute the dissimilarities between the new input tensor X and all the clusters' representatives $\tilde{X}_k$, and then select the cluster with the smallest dissimilarity $\delta(X, \tilde{X}_k)$. However, this procedure is not reasonable when repeated many times, because of the computation time required to evaluate the dissimilarities. Indeed, as further explained in the next section, dissimilarity measures that are suitable for model order reduction applications may involve numerical simulations. Hence, the time saving obtained by model order reduction would be counterbalanced by the time-consuming operations required for cluster assignment. The *true classifier* defined by:

$$\mathcal{K}_K(X) = \underset{k \in [\![1;K]\!]}{\arg\min}(\delta(X, \tilde{X}_k)) \tag{15}$$

is too expensive because it is based on numerical simulations. Note that $\mathcal{K}_K$ is not an artificial neural network. When using the ROM-net, the true classifier $\mathcal{K}_K$ is replaced by the approximate classifier $\mathcal{F}_K$ to bypass the computations required for cluster assignment. In the application presented at the end of this paper, replacing the true classifier by the approximate one enables fast ROM selection with a computation time reduced by a factor of 60. The next section gives some general guidelines for the training of a dictionary-based ROM-net.

Training procedure for dictionary-based ROM-nets

Let K be a positive integer. In this section, we describe the training phase of a dictionary-based ROM-net $\mathcal{R}_K$ made of a classical deep classifier $\mathcal{F}_K$ and a K-ROM-dictionary $\mathcal{D}_K$. First, an automatic data labelling procedure is presented. It aims at preparing the data for the supervised learning of the deep classifier $\mathcal{F}_K$, using simplified numerical simulations and a clustering algorithm. Then, we train the deep classifier $\mathcal{F}_K$ and build the K-ROM-dictionary $\mathcal{D}_K$ on the grounds of the clusters identified by the labelling procedure. Figure 3 summarizes the main steps for the training of a dictionary-based ROM-net.

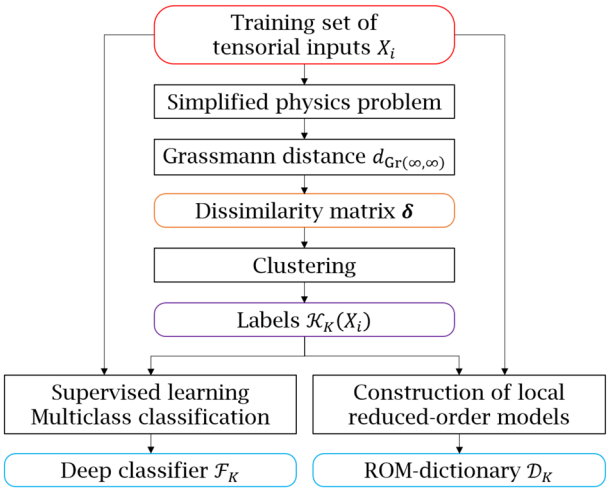

Fig. 3 Workflow for the training of a dictionary-based ROM-net

Automatic data labelling via clustering

ROM-oriented dissimilarity measure

Clustering the input space $\mathcal{X}$ enables labelling the training data for the deep classifier, and guides the construction of the ROM-dictionary by defining regions of the input space where high-fidelity simulations must be run to build local reduced-order models. The key point is the choice of the dissimilarity measure for clustering. Indeed, clustering aims at grouping points of a dataset that are similar. As shown in the last section of this article, for some specific applications, defining a dissimilarity measure based on a distance between input tensors leads to inaccurate reduced-order solutions. Furthermore, the difficulties of high-dimensional clustering appear when dealing with large input tensors.

In the application to structural mechanics presented at the end of this paper, this issue is due to the complex interaction between the thermal and the mechanical loadings. In this application, the input tensor X is a temperature field influencing the mechanical response of the material. To illustrate the aforementioned difficulty, let us imagine two temperature fields T_1 and T_2 taking different values only in a very small part ω of the structure Ω. Let us introduce a third temperature field T_3 being equal to T_1 in ω and taking arbitrary values in the rest of the solid domain. If ω is a critical zone from a mechanical point of view, T_1 and T_2 might lead to dissimilar displacement fields, while T_3 might give approximately the same displacement field as T_1 (if thermal expansion is negligible with respect to the strains induced by the mechanical boundary conditions). In this case, T_1 and T_3 should be assigned to the same cluster, while T_2 should be assigned to another one. However, taking the L^2 distance between temperature fields as the dissimilarity measure would assign T_1 and T_2 to the same cluster, as these fields are identical in most of the solid domain.

Consequently, for such cases, one must define a *ROM-oriented dissimilarity measure* accounting for the variability induced by the stochastic input tensor. In this paper, the dissimilarity is defined using the Grassmann distance [37] between subspaces spanned by outputs of a simplified physics problem. The simplified physics problem consists in computing a few time steps of the original problem with a less restrictive convergence criterion. The boundary conditions can even be simplified to facilitate convergence. The

idea is to discover the subspace in which the solution evolves at the beginning of the simulation for a given input X. Two input variabilities leading to solutions lying in nearby subspaces (in terms of principal angles) are then considered as similar.

Before giving a formal definition of the ROM-oriented dissimilarity measure, let us define some useful concepts. The Grassmannian $\mathrm{Gr}(k, n)$ is a Riemannian manifold whose points are all the k-dimensional linear subspaces of $\mathbb{R}^n$. The infinite Grassmannian $\mathrm{Gr}(k, \infty)$ parametrizes the k-dimensional subspaces in $\mathbb{R}^n$ for all $n \geq k$. As shown in [37], one can define a distance between two subspaces of different dimensions. This distance is independent of the dimension of the ambient space. The Grassmann metric $d_{\mathrm{Gr}(\infty,\infty)}$ is defined on the doubly-infinite Grassmannian $\mathrm{Gr}(\infty, \infty)$, which parametrizes subspaces of all dimensions regardless of the ambient space [37]. In practice, this metric can be obtained with the following formula:

$$d_{\mathrm{Gr}(\infty,\infty)}(\mathcal{A}, \mathcal{B}) = \left(\frac{\pi^2}{4} \mid a - b \mid + \sum_{i=1}^{\min(a,b)} \alpha_i^2 \right)^{1/2} \tag{16}$$

where $\mathcal{A}$ and $\mathcal{B}$ are two linear subspaces of dimension a and b respectively, and where the α_i's are the principal angles obtained by singular value decomposition:

$$\mathbf{A}^T\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{17}$$

with $\mathbf{A}$ and $\mathbf{B}$ being semi-orthogonal matrices whose columns form orthonormal bases of $\mathcal{A}$ and $\mathcal{B}$ respectively, and $\mathbf{\Sigma} \in \mathbb{R}^{a \times b}$ is a matrix whose only nonzero coefficients are $\Sigma_{ii} = \cos(\alpha_i)$ for $i \leq \min(a, b)$. These coefficients are non-negative, thus $\alpha_i \in [0; \pi/2]$ for all i.

Let us introduce the application $\mathcal{V} : \mathcal{X} \rightarrow \mathrm{Gr}(\infty, \infty)$ assigning the input tensor $X \in \mathcal{X}$ to the subspace $\mathcal{V}(X)$ spanned by the primal variable $\mathbf{U}$ of the governing equations during the numerical simulation of the simplified physics problem:

$$\mathcal{V}(X) = \mathrm{span}\left(\{\mathbf{U}_n(X),\ n \in [\![1; n_t]\!]\}\right) \tag{18}$$

with n_t being the number of time increments in the simplified simulation and $\mathbf{U}_n(X)$ denoting the vector representation of the primal variable computed for the input X and evaluated at the n-th time increment. The ROM-oriented dissimilarity measure used for the clustering of $\mathcal{X}$ is written δ and defined by:

$$\forall (X_i, X_j) \in \mathcal{X}^2, \quad \delta(X_i, X_j) = d_{\mathrm{Gr}(\infty,\infty)}\left(\mathcal{V}(X_i), \mathcal{V}(X_j)\right) \tag{19}$$

The dissimilarity measure is computed for all pairs of inputs belonging to the training set, resulting in a dissimilarity matrix $\boldsymbol{\delta} \in \mathbb{R}^{n_T \times n_T}$, where n_T is the cardinality of the training set.

The clustering algorithm

Once the dissimilarity matrix is calculated, a clustering algorithm must be applied to partition the dataset into K clusters. The choice of the algorithm depends on the context. In this methodology, clustering is used for the definition of local approximations of a non-linear solution manifold. Hence, the algorithm must focus on compactness rather than connectivity when looking for clusters in the dataset. This property is satisfied by k-means algorithm [38], the most well-known clustering approach. However, this algorithm needs to calculate clusters' centroids or means, which is impossible when the input data correspond to vector spaces. For this reason, we choose the k-medoids algorithm presented in

[39], relying on a Voronoi iteration approach like k-means. The algorithm proposed in [39] can be summarized as follows:

- *Initialization step:* select K initial medoids from the dataset.
- Repeat the two following steps until convergence:
  - *Data assignment step:* assign each point of the dataset to the cluster corresponding to its closest medoid.
  - *Medoid update step:* for each cluster, update the medoid by finding the point which minimizes the sum of distances to all the points in the cluster.

The choice of the hyperparameter K depends on the problem. More details concerning this aspect of the methodology can be found in the final section of this article.

Construction of the ROM-net

Training of a deep classifier for fast model selection

The ROM-net's classifier $\mathcal{F}_K$ is trained in a supervised fashion from pairs of examples $(X_i, \mathcal{K}_K(X_i))$ given by the clustering algorithm, where the label $\mathcal{K}_K(X_i) \in [\![1; K]\!]$ is the index of the cluster containing X_i. As explained earlier in the article, the true classifier defined by:

$$\mathcal{K}_K(X) = \underset{k\in[\![1;K]\!]}{\arg\min}(\delta(X, \tilde{X}_k)) = \underset{k\in[\![1;K]\!]}{\arg\min}\left(d_{\mathrm{Gr}(\infty,\infty)}\left(\mathcal{V}(X), \mathcal{V}(\tilde{X}_k)\right)\right) \tag{20}$$

is too expensive because it is based on numerical simulations, which motivates the use of an approximate classifier. In the previous equation, $\tilde{X}_k$ is the medoid of the k-th cluster.

The dataset is split into a training, a validation and a test set. For a given deep neural network architecture and for a given set of hyperparameters, the parameters of the classifier $\mathcal{F}_K$ are calibrated on the training set via backpropagation with Adam optimizer [40]. The accuracy of the calibrated classifier is evaluated on the validation set. The classifier is calibrated with different architectures and hyperparameters settings until the accuracy on the validation set reaches a satisfying value. Once the best architecture and set of hyperparameters have been selected, the calibrated classifier is evaluated on the test set to get the accuracy of the model for new unseen data. When the input X is an image, one could use well-known convolutional neural networks' architectures and fine-tune their pre-trained parameters to adapt the model to the current data, which is a common transfer learning technique [41].

Construction of a ROM-dictionary

Contrary to the classifier, the ROM-dictionary $\mathcal{D}_K$ is trained in an unsupervised fashion. Clustering results help for the selection of data-points in $\mathcal{X}$ for which high-fidelity simulations must be run. The solutions computed at every time step of these high-fidelity simulations are called snapshots. Selecting clusters' medoids as simulation points for snapshots is recommended, since the clusters are represented by their medoids. Additional snapshots can be computed if necessary. For each cluster, the snapshot POD is applied to the set of snapshots to obtain a local reduced-order basis.

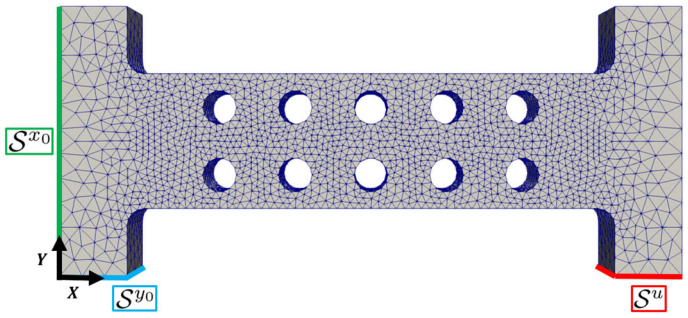

Fig. 4 Finite-element mesh of the structure. Contains 33047 quadratic tetrahedral elements and 54649 nodes. Details about the boundary conditions can be found in Eqs. (21)

Application to an anisothermal elastoplastic problem in structural mechanics

In this section, a temperature-dependent problem in structural mechanics is considered. Our objective is to study the influence of thermal loading uncertainties on a mechanical quantity of interest such as a damage field. The input tensor corresponds to the final temperature field in the structure, defined by a truncated Gaussian field. The quantity of interest is a damage indicator based on the accumulated plastic strain field and defined on the whole structure.

The high-fidelity model

Let us consider the solid body Ω shown on Fig. 4. The heat produced by mechanical phenomena is neglected, which enables solving the heat equation and then use the resulting temperature field history as a thermal loading for the mechanical problem. The structure is subjected to a displacement-controlled monotonic loading. Assuming a quasi-static evolution, equilibrium equations at the local level and boundary conditions read:

$$\begin{cases} \mathbf{div}(\boldsymbol{\sigma}(\mathbf{x},t)) = \mathbf{0} & \forall t \in [0;1] \quad \forall \mathbf{x} \in \Omega \\ \mathbf{u}(\mathbf{x},t).\mathbf{e}_y = -tu_0 & \forall t \in [0;1] \quad \forall \mathbf{x} \in \mathcal{S}^u \\ \mathbf{u}(\mathbf{x},t).\mathbf{e}_x = 0 & \forall t \in [0;1] \quad \forall \mathbf{x} \in \mathcal{S}^{x_0} \\ \mathbf{u}(\mathbf{x},t).\mathbf{e}_y = 0 & \forall t \in [0;1] \quad \forall \mathbf{x} \in \mathcal{S}^{y_0} \\ \mathbf{u}(\mathbf{0},t).\mathbf{e}_z = 0 & \forall t \in [0;1] \\ \boldsymbol{\sigma}(\mathbf{x},t).\mathbf{n}(\mathbf{x},t) = \mathbf{0} & \forall t \in [0;1] \quad \forall \mathbf{x} \in \partial\Omega \setminus (\mathcal{S}^u \cup \mathcal{S}^{x_0} \cup \mathcal{S}^{y_0}) \end{cases} \tag{21}$$

The structure is made of an elastoplastic generalized standard material described by the von Mises yield criterion and a nonlinear isotropic hardening law. In the framework of the infinitesimal strain theory, the constitutive equations are:

- Hooke's law:

$$\boldsymbol{\sigma} = \mathbb{C} : (\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^p - \alpha(T - T_0)\mathbf{1}) \tag{22}$$

- von Mises yield criterion with isotropic hardening:

$$f(\boldsymbol{\sigma}, R) = \sigma_{\text{eq}}(\boldsymbol{\sigma}) - R - \sigma_y \qquad \sigma_{\text{eq}}(\boldsymbol{\sigma}) = \sqrt{\frac{3}{2}\mathbf{s} : \mathbf{s}} \qquad \mathbf{s} = \boldsymbol{\sigma} - \frac{1}{3}tr(\boldsymbol{\sigma})\mathbf{1} \tag{23}$$

- Nonlinear isotropic hardening law (with p denoting the accumulated plastic strain):

$$R(p) = R_\infty(1 - \exp(-bp)) \tag{24}$$

- Flow rule for the plastic strain rate tensor:

$$\dot{\boldsymbol{\varepsilon}}^p = \frac{3}{2}\dot{p}\frac{\mathbf{s}}{\sigma_{\text{eq}}(\boldsymbol{\sigma})} \tag{25}$$

- Karush–Kuhn–Tucker conditions:

$$\dot{p} \geq 0, \quad f \leq 0, \quad \dot{p}f = 0 \tag{26}$$

- Consistency condition for the determination of the plastic multiplier:

$$\dot{p}\dot{f} = 0 \tag{27}$$

Under the assumption of isotropic elasticity, the fourth-order elastic stiffness tensor $\mathbb{C}$ can be decomposed as follows:

$$\mathbb{C} = \frac{E}{1+\nu}\mathbb{K} + \frac{E}{1-2\nu}\mathbb{J} \tag{28}$$

where E is the Young's modulus, ν is the Poisson's ratio, and $\mathbb{K}$ (resp. $\mathbb{J}$) is the projector onto the space of deviatoric (resp. spherical) second-order tensors. Material constants $E, \nu, \alpha, \sigma_y, R_\infty$ and b generally depend on the temperature. For this application, temperature-dependent coefficients E, α, σ_y are taken from experimental data on high strength structural steels for fire-resistant structures [42]. The other material parameters are taken as constants. The thermal loading applied to the structure is defined by:

$$\forall t \in [0;1], \qquad \forall \mathbf{x} \in \Omega, \qquad T(\mathbf{x}, t) = T_0 + t(T_{\max}(\mathbf{x}) - T_0) \tag{29}$$

with $T_0 = 22^oC$. The field $T_{\max}(\mathbf{x})$ will be replaced by a random temperature field to account for uncertainties on the thermal loading, while the mechanical loading is deterministic. Hence, for this study, the stochastic input X is a tensorial representation of the random temperature field obtained at $t = 1$. Let us consider a simple damage indicator $D : \Omega \times [0;1] \to [0;1]$ defined by:

$$\forall t \in [0;1], \qquad \forall \mathbf{x} \in \Omega, \qquad D(\mathbf{x}, t) = \frac{p(\mathbf{x}, t)}{p_f} \tag{30}$$

where p_f is the material's plastic strain to failure. A crack initiates at $\mathbf{x}_0 \in \Omega$ if $D(\mathbf{x}_0, t)$ reaches the value 1 for some $t \in [0;1]$. The quantity of interest for this application is the field $Y = D(\,.\,, 1) : \Omega \to [0;1]$.

The high-fidelity mechanical problem is solved using the finite-element method. Numerical simulations are performed with Z-set software [43].

Stochastic model for the thermal loading

A training set of random thermal loadings is generated using the stochastic model described in Appendix A. Briefly speaking, the stochastic model draws random combinations of fluctuation modes which are superposed with a reference temperature field $T_{\text{ref}} : \Omega \to \mathbb{R}$. In Eq. (29), the field $T_{\max}$ is replaced by the resulting random temperature fields, which gives the random thermal loadings. The reference temperature field defines the mean thermal loading. For this application, the reference temperature field is uniform at 650^oC. For real-world applications, the reference temperature field can be given by aerothermal simulations. Random temperature fields are denoted by $T_{\text{rand}} : \Omega \times \Theta \to \mathbb{R}$, where Θ is the sample space of the probability space associated to the stochastic model described in Appendix A.

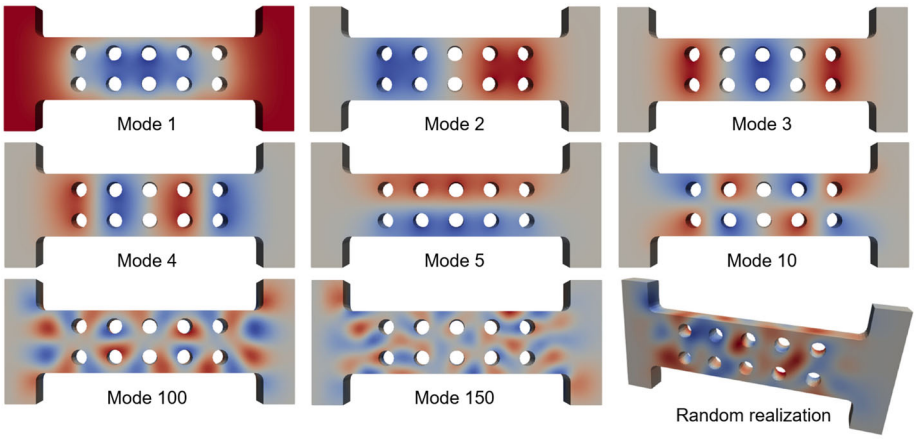

Fig. 5 Construction of random temperature fields. 150 zero-mean temperature fluctuation modes are generated, randomly combined and then superposed to a reference temperature field to get random temperature fields

Generating a database of $n_T = 10^4$ random temperature fields $T_{\text{rand}}(., \theta_i) : \Omega \to \mathbb{R}$ for $i \in [\![1; n_T]\!]$ from 150 fluctuation modes takes 17 min on one single computer thread. The standard deviation of random fluctuations is 50^oC. The database $\mathcal{T}$ of thermal loading variabilities can then be defined as (see Fig. 5):

$$\begin{aligned} \mathcal{T} = \{T : \Omega \times [0;1] \to \mathbb{R} \quad | \quad \exists\, i \in [\![1; n_T]\!], \\ \forall t \in [0;1],\ \forall x \in \Omega, \quad T(\mathbf{x}, t) = T_0 + t(T_{\text{rand}}(\mathbf{x}, \theta_i) - T_0)\} \end{aligned} \tag{31}$$

Remark When training the ROM-net, it is assumed that we have no prior knowledge of the data-generating stochastic model. Training data may come from complex aerothermal simulations with random boundary conditions defined by experts. In the present case, in the absence of such data, we use a stochastic model to generate the training data. This stochastic model is parametric, since every random temperature field comes from a random linear combination of 150 fluctuation modes. However, when training the deep classifier for local ROM recommendation, the training data corresponds to nodal values of the temperature fields rather than the 150 random coordinates. This way, in the test phase (or *online* phase in the model order reduction community), the ROM-net can be applied to thermal loadings coming from unknown stochastic models, complex aerothermal simulations or even experiments, as long as these thermal loadings correspond to perturbations of the reference thermal loading. Hence, the ROM-net can deal with nonparametrized variabilities of the input data. □

Figure 6 illustrates the influence of the thermal loading on the damage field computed with the high-fidelity model. The fields on the left represent two different temperature fields reached at $t = 1$ in the simulation. The critical zone is located around the first column of holes on the left-hand side of the structure. The second temperature field (case B) on the figure takes high values in this zone, leading to high values of the damage indicator. On the contrary, values taken by the first temperature field (case A) in this zone are relatively small. The resulting damage field takes smaller values in the first column of holes. One can observe that the shapes of damaged zones in A and B are not the same.

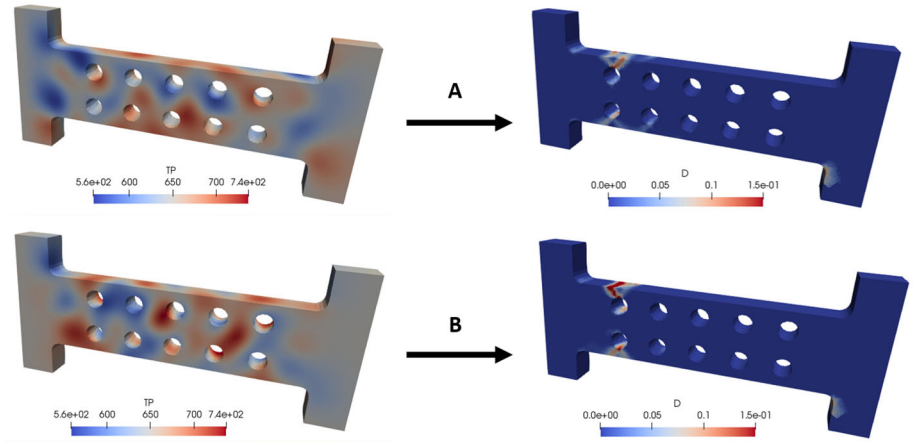

Fig. 6 Influence of the temperature field. Two damage fields (on the right) obtained from two different temperature fields (on the left)

In addition, the second column of holes is a bit damaged in A, while this region remains undamaged in B.

Construction of a dictionary-based ROM-net

To compute the damage field as a function of the thermal loading in a reasonable time, a dictionary-based ROM-net $\mathcal{R}_K$ based on a classical deep classifier $\mathcal{F}_K$ is used. In this paper, we focus on the training of the deep classifier $\mathcal{F}_K$.

Computation of the ROM-oriented dissimilarity matrix

For every thermal loading, we solve a simplified mechanical problem which is similar to the original one, with a less restrictive tolerance for Newton–Raphson's algorithm to reduce the number of required time steps. For the sake of simplicity, only two displacement fields of the simplified simulation are kept: one in the elastic regime, and one in the plastic regime. The space $\mathcal{V}(T)$ is the 2-dimensional space spanned by these fields. The solutions at the other time steps are discarded, but computing them is necessary to ensure the convergence of all the simplified simulations. The $10^4 \times 10^4$ matrix $\boldsymbol{\delta}$ is defined as the dissimilarity matrix whose coefficients are:

$$\delta_{ij} = \delta(T_i, T_j) = d_{\mathrm{Gr}(\infty,\infty)}\left(\mathcal{V}(T_i), \mathcal{V}(T_j)\right) \tag{32}$$

The 10^4 simplified mechanical simulations are distributed between 84 computer threads. The total computation time is 9h05min, which represents 5min per simulation. Once the simplified simulations are done, computing the dissimilarity matrix takes 11h16min, if its coefficients are distributed between 48 computer threads. Note that less than half of the coefficients are calculated, since the dissimilarity matrix is symmetric with zeros on its diagonal.

k-medoids clustering

The ROM-oriented dissimilarity is used to cluster the data with the k-medoids algorithm. The number K of clusters is a hyperparameter provided by the user. Numerous empirical methods have been proposed to estimate the best number of clusters for different criteria.

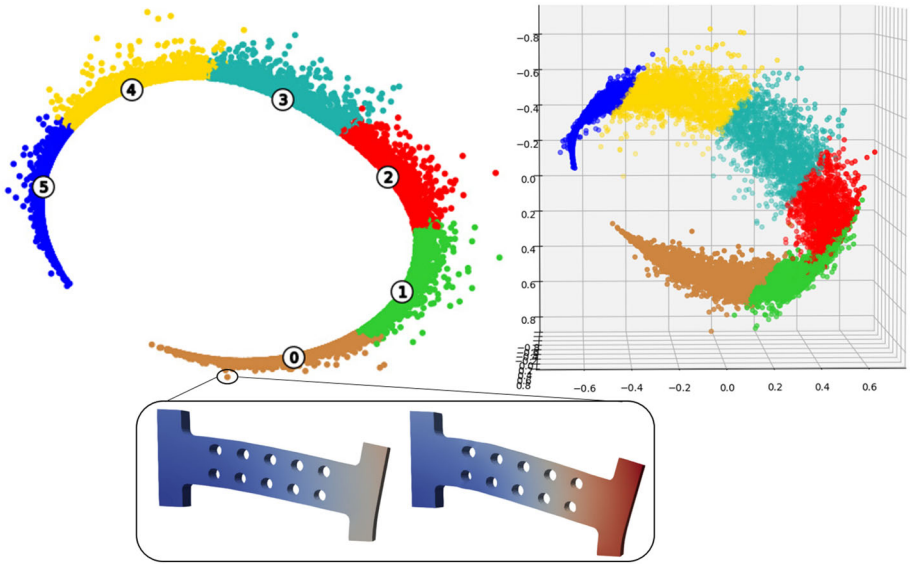

Fig. 7 MDS plots of clustering results. The positions of the clusters' labels correspond to the medoids. Each point corresponds to a subspace $\mathcal{V}(T_i)$ spanned by two displacements fields of the simplified mechanical problem with the thermal loading T_i

In our case, the dataset is not organized in distinct clusters. Therefore, the clustering algorithm is applied for different values of K, and the quality of clustering results is evaluated using silhouette analysis [44]. When selecting the most appropriate number of clusters based on silhouette analysis, one must also consider the trade-off between large numbers giving a better approximation of the nonlinear manifold, and small numbers facilitating the classification problem for the deep neural network. In addition, using a large number of clusters increases the cost of the construction of the ROM-dictionary. For the current problem, $K = 6$ has been identified as a good compromise. A single run of the clustering algorithm takes about 10 s. The true classifier associated to this clustering procedure is denoted by $\mathcal{K}_K$.

Visualization of clusters on the nonlinear manifold

The clustering results can be visualized thanks to Multidimensional Scaling (MDS) [45]. MDS is an information visualization method which consists in finding a low-dimensional dataset $\mathbf{Z}_0$ whose matrix of Euclidean distances $\mathbf{d}(\mathbf{Z}_0)$ is an approximation of the input dissimilarity matrix $\boldsymbol{\delta}$. To that end, a cost function called stress function is minimized with respect to $\mathbf{Z}$:

$$\mathbf{Z}_0 = \arg\min_{\mathbf{Z}} \left(\varsigma(\mathbf{Z};\boldsymbol{\delta})\right) = \arg\min_{\mathbf{Z}} \left(\sum_{i<j} (\delta_{ij} - d_{ij}(\mathbf{Z}))^2\right) \tag{33}$$

This minimization problem is solved with the algorithm Scaling by MAjorizing a COmplicated Function (SMACOF, [46]) implemented in Scikit-learn [47]. Figure 7 shows clustering results with low-dimensional representations obtained by metric MDS. The relative error $\varsigma(\mathbf{Z}_0;\boldsymbol{\delta})/\varsigma(\mathbf{0};\boldsymbol{\delta})$ is 11% for the 2D representation and 10% for the 3D representation. These visualizations illustrate the nonlinear manifold on which solutions of the mechan-

ical problem evolve when changing the thermal loading. Note that the positions of the clusters' labels coincide with the medoids.

Multiclass classification

Classifier based on an ensemble of DNNs. Clustering results obtained with the true classifier $\mathcal{K}_K$ enable training the approximate classifier $\mathcal{F}_K$ of the ROM-net $\mathcal{R}_K$. Among the 10^4 thermal loadings in the dataset, 6400 are used as training data, 1600 as validation data, and 2000 as test data for final evaluation. As the data labelling procedure involves numerical simulations, the dataset contains a limited amount of training examples in comparison with standard image classification problems. Working on a small dataset makes our deep classifier prone to overfitting. To address this issue, we use the *ensemble averaging method* [48], which consists in taking an ensemble of classifiers and averaging their predicted membership probabilities. Ensemble averaging is a common technique in *ensemble learning*. Generally speaking, ensemble methods aim at creating a meta-estimator from several base estimators (or models). Combining different estimators leads to more robust predictions and reduces overfitting. In addition, using an ensemble method replaces the task of finding a single very accurate model by the task of building an effective meta-estimator from several models with lower accuracies. The winners of numerous deep learning challenges used ensemble averaging to improve their predictions [49–51]. Our ensemble contains $N_{\text{models}} = 12$ different DNNs trained for the same classification problem with the same training data using Keras [52] and Tensorflow [53] libraries on Python, but with different architectures and loss functions. All of them use the softmax activation function to get membership probabilities. These predictions are combined in a soft voting scheme to give the final prediction:

$$\mathcal{F}_K(T) = \underset{l\in[\![1;K]\!]}{\arg\max}(y^{\text{pred}}(T,l)), \quad \text{with} \quad y^{\text{pred}}(T,l) = \frac{1}{N_{\text{models}}}\sum_{k=1}^{N_{\text{models}}} y_l^k(T) \tag{34}$$

with $y_l^k(T)$ denoting the membership probability of class l predicted by the k-th model. The 12 models in the ensemble include fully-connected networks (FC), convolutional neural networks (CNN) [54] and global average pooling convolutional neural networks (GAP-CNN) [55]. These DNNs are trained with different loss functions, namely cross-entropy, balanced cross-entropy to handle class imbalance, and the focal loss [56] which enables focusing more on misclassified data. Using an ensemble enables recycling the best DNNs obtained during training, and overcoming some weaknesses of every single model in the ensemble. Training one of the DNNs used in this ensemble on a Nvidia Quadro K5200 GPU takes about 2h on average in our case.

An important aspect of our classification problem is the preprocessing step, in which thermal loadings are prepared to be fed into neural networks. Thermal loadings in $\mathcal{T}$ are represented by their corresponding random temperature fields reached at $t = 1$. These temperature fields are projected onto a $38 \times 17 \times 4$ regular grid defined on a bounding box surrounding the solid body, see Fig. 8. For the grid's vertices being inside Ω, the value of the temperature is evaluated using the finite-element shape functions, while vertices being outside Ω are assigned a zero value. This procedure gives a *3D bitmap image* of the temperature field, represented by a third-order tensor. Thanks to this tensorial representation, 3D convolutional filters can now be applied to extract features of the input data, like 2D convolutional filters do for image analysis. For fully-connected networks, although

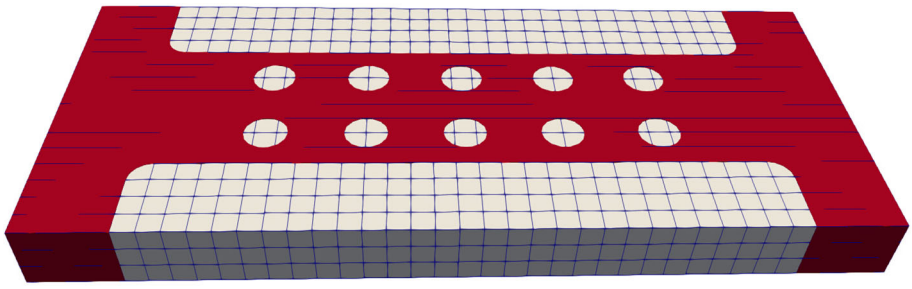

Fig. 8 Input tensor. Bounding box used to get a 3D image of a temperature field to be fed into DNNs

Table 1 Accuracies of the classifiers composing the ensemble

Model	Type	Loss function	Test accuracy (%)
1	FC	Cross-entropy	72.50
2	FC	Balanced cross-entropy	71.90
3	FC	Focal loss	67.95
4	FC	Focal loss	68.45
5	CNN	Cross-entropy	72.05
6	CNN	Cross-entropy	72.45
7	CNN	Cross-entropy	73.75
8	CNN	Balanced cross-entropy	72.00
9	CNN	Focal loss	69.45
10	GAP-CNN	Cross-entropy	63.05
11	GAP-CNN	Cross-entropy	68.00
12	GAP-CNN	Balanced cross-entropy	70.05
Ensemble	Unweighted averaging	–	80.00

Table 2 Classification results

Class	Precision	Recall	F1-score	Support
0	0.92	0.92	0.92	421
1	0.80	0.73	0.76	335
2	0.66	0.70	0.68	263
3	0.66	0.73	0.69	280
4	0.77	0.74	0.75	326
5	0.92	0.90	0.91	375
Micro avg	–	–	0.80	2000
Macro avg	0.788	0.787	0.785	2000
Weighted avg	0.805	0.800	0.800	2000

the third-order tensor is flattened, the projection on the grid acts as a subsampling procedure. This preprocessing operation takes about 3 min when the 10^4 fields are distributed between 280 computer threads, which means that the projection of a single field takes 5 s.

Analysis of classification results In the present study, when evaluated on the test set, the ensemble of DNNs reaches an accuracy of 80%, whereas accuracies of its base classifiers range from 63.05 to 73.75%, see Table 1. As expected, ensemble averaging reduces overfitting and thus improves the ability of the classifier $\mathcal{F}_K$ to generalize to new unseen data. Table 2 summarizes the values of precision, recall and F1-score. Figure 9 gives the confu-

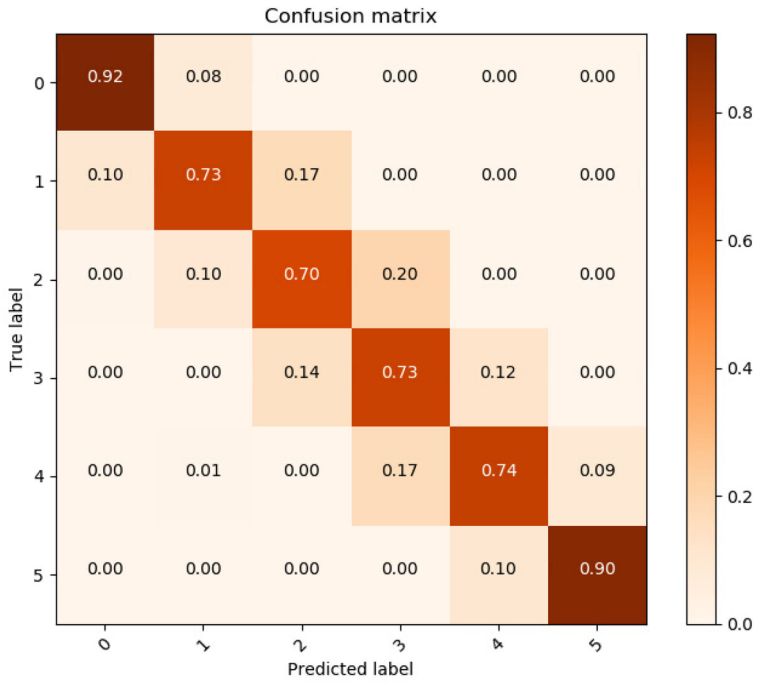

Fig. 9 Confusion matrix. Confusion matrix obtained on test data with the ensemble of DNNs

sion matrix, whose coefficient (i, j) is the percentage of examples of class i being assigned to class j. This matrix is diagonally dominant here, indicating that the predicted class usually corresponds to the true class. Because of the elongated aspect of the dataset and of the numerotation adopted here, the tridiagonal aspect of the confusion matrix indicates that misclassified examples are assigned to neighbouring clusters. This result can also be observed when visualizing misclassified examples on MDS plots, see Figs. 10, 11 and 12. On these figures, one can clearly see that misclassified examples of class i are mostly located close to the border between the i-th cluster and its neighbours. This is actually a nice property, because it means that when the ensemble fails to select the appropriate reduced-order model in the dictionary, it returns a reduced-order model that covers a part of the manifold that is close to the target one.

Evaluation of the methodology and discussion

Let us quickly summarize what has been done up to that point. A dataset of 10^4 thermal loadings has been generated with a stochastic model. Its ROM-oriented dissimilarity matrix has been computed. Based on this matrix, the dataset has been split into $K = 6$ clusters with k-medoids clustering algorithm. An ensemble of DNNs has been trained to assign new unseen thermal loadings to the best clusters using the ensemble averaging method.

When considering a new thermal loading, true cluster assignment requires one simplified simulation (5 min) and the computation of six Grassmann distances (less than 1 s). On the other hand, when using the deep classifier $\mathcal{F}_K$, preprocessing operations take 5 s and the evaluation of the deep classifier is quasi-instantaneous. Hence, the computation

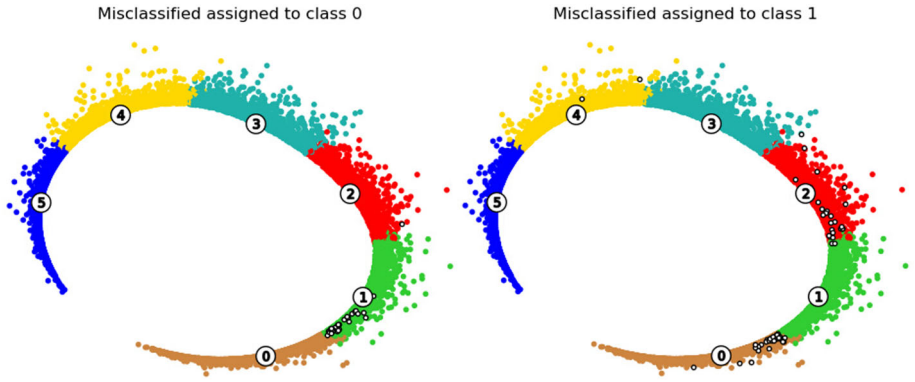

Fig. 10 Misclassified examples (classes 0 and 1)

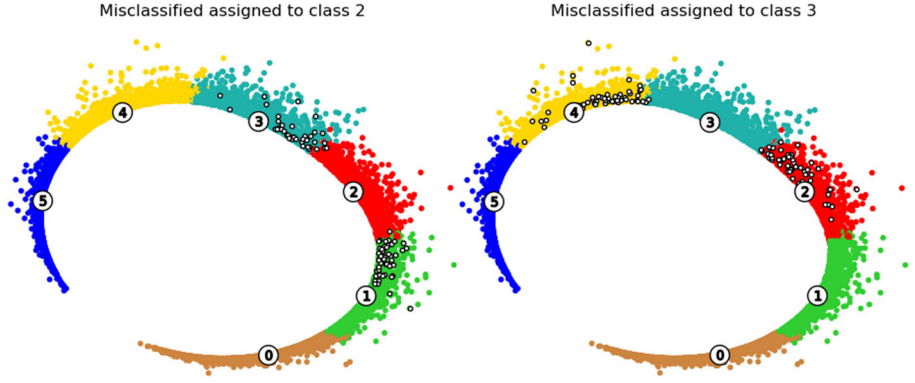

Fig. 11 Misclassified examples (classes 2 and 3)

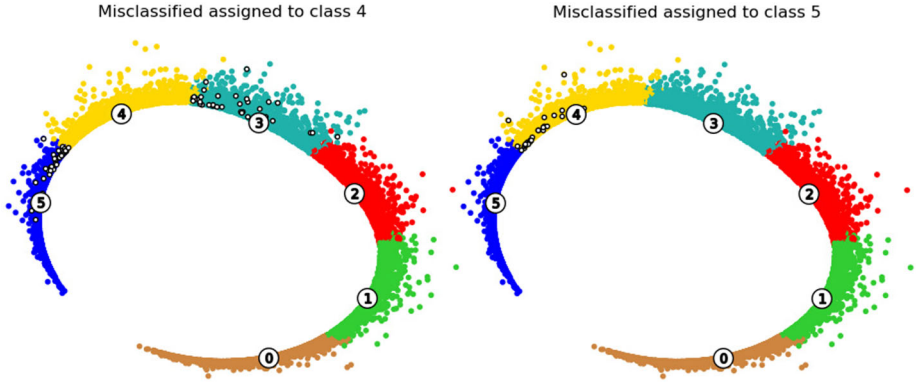

Fig. 12 Misclassified examples (classes 4 and 5)

time for the selection of the best reduced-order model is decreased by a factor of 60 when using the ROM-net's deep classifier $\mathcal{F}_K$ instead of the true classifier $\mathcal{K}_K$.

The clustering of the thermal loading database $\mathcal{T}$ with the ROM-oriented dissimilarity has defined 6 clusters which can be used to construct a dictionary of 6 local ROMs. Let us compare our methodology with another approach consisting in the construction of a

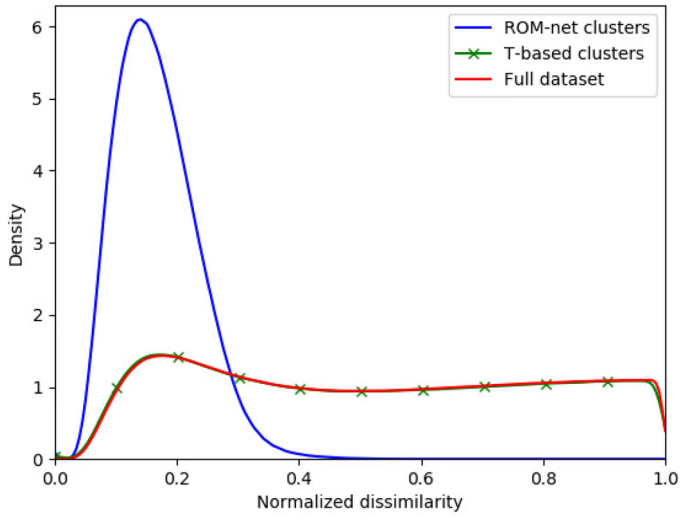

Fig. 13 Distributions of intra-cluster ROM-oriented dissimilarities

temperature-based ROM-dictionary $\mathcal{D}_T$. Such a dictionary comes from a direct clustering of the input space, that is, a k-medoids clustering of $\mathcal{T}$ using distances between the temperature fields evaluated at $t = 1$. This clustering only considers the input data and does not use simplified simulations to account for mechanical phenomena. Hence, cluster assignment is directly obtained by taking the minimum distance to clusters' medoids. In this case, there is no need for DNNs since the cost of the classification task is negligible. However, clustering high-dimensional data leads to meaningless results due to the loss of contrast in pairwise distances. This problem is known as *the curse of dimensionality* [19] and appears naturally when dealing with fields defined on a finite-element mesh. To overcome this difficulty, dimensionality reduction techniques must be applied prior to clustering. Distances are calculated in the low-dimensional latent space, whose dimension is a hyperparameter. In this paper, principal component analysis (PCA) is applied for linear dimensionality reduction with 30 principal components, the dimension 30 being a compromise between large dimensions and low dimensions discarding too much information.

The values $\delta(T_i, T_j)$ for two inputs T_i, T_j belonging to the same cluster are called *intra-cluster* ROM-oriented dissimilarities. The distributions of intra-cluster ROM-oriented dissimilarities for the ROM-net $\mathcal{R}_K$ and temperature-based dictionaries are shown on Fig. 13. The distribution obtained with a temperature-based dictionary does not depend on the number of clusters, and coincides with the distribution of dissimilarities $\delta(T_i, T_j)$ obtained without imposing the inputs T_i, T_j to belong to the same cluster.

This result shows that a distance on temperature fields does not lead to *local* ROMs for the Grassmann distance in the present application. However, in the context of ROM interpolation, the Grassmann distance was shown to be the adequate concept when manipulating ROMs [25,27,28]. Because of the complex interactions between the thermal and the mechanical loadings, direct clustering in the space of temperature fields is not appropriate for the mechanical problem presented in this paper.

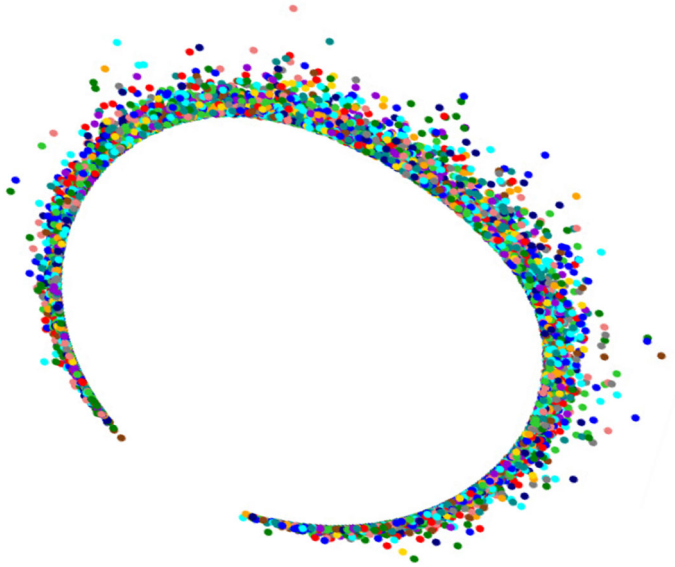

Fig. 14 Temperature-based clustering results visualized on the MDS plot of the dissimilarity matrix $\boldsymbol{\delta}$

When using some dissimilarity measure δ' for clustering in order to build a ROM-dictionary, the comparison of the distribution of intra-cluster ROM-oriented dissimilarities with the global distribution of ROM-oriented dissimilarities can be used as a validation criterion. If these distributions are similar, it means that the dissimilarity measure δ' does not provide *local* ROMs for the Grassmann distance. In this case, a dictionary-based ROM-net should be used with the ROM-oriented dissimilarity measure based on the Grassmann distance.

Figure 14 gives another visualization of the results shown on Fig. 13. It illustrates clustering results obtained for a temperature-based dictionary with 13 clusters. Points belonging to the same cluster have the same color. The high dispersion of the points assigned to a given cluster proves that direct clustering of the input space does not lead to *local* ROMs for the Grassmann distance.

These promising results highlight the potential of dictionary-based ROM-nets. Once clusters have been defined, one can construct one local ROM for each cluster, like in [17,18] where small local ROMs outperform a single global ROM in terms of accuracy and speed. This study in the context of dictionary-based ROM-nets is underway.

Conclusion

The concept of ROM-net gives a general framework for reduced-order model adaptation using deep neural networks. In this article, the potential of dictionary-based ROM-nets has been illustrated on a mechanical problem with nonparametrized variabilities of the thermal loading. It has been shown that direct clustering of the input space may give clusters which cannot be exploited to define *local* reduced-order models. This issue can be avoided by defining a ROM-oriented dissimilarity involving the Grassmann metric on results of simplified numerical simulations. Online cluster assignment can be performed

with a classifier based on deep neural networks to bypass numerical simulations, which reduces the computation time by a factor of 60.

Abbreviations
CFD: Computational fluid dynamics; CNN: Convolutional neural network; DNN: Deep neural network; FC: Fully-connected; GAP: Global average pooling; GPU: Graphics processing unit; HROM: Hyperreduced-order model; MDS: Multidimensional scaling; PCA: Principal component analysis; PDE: Partial differential equation; POD: Proper orthogonal decomposition; ROM: Reduced-order model.

Acknowledgements
The authors wish to thank Felipe Bordeu (SafranTech) and Julien Cortial (SafranTech), who implemented the Python library *BasicTools* (https://gitlab.com/drti/basic-tools) with FC.

Authors' contributions
The methodology was imagined by TD, FC, NA and DR. TD developed the algorithm with the valuable suggestions of DR, FC and NA. The algorithm was implemented by TD with the help of FC. The manuscript was written by TD and reviewed by DR, FC and NA. All authors read and approved the final manuscript.

Funding
Study funded by Safran and ANRT (Association Nationale de la Recherche et de la Technologie).

Availability of data and materials
Not applicable.

Ethics approval and consent to participate
The authors approve the Journal's ethics policy and consent to participate.

Consent for publication
The authors give their consent for publication.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]SafranTech, Rue des Jeunes Bois, Chateaufort, 78114 Magny-les-Hameaux, France, [2]MINES ParisTech, PSL University, Centre des materiaux (CMAT), CNRS UMR 7633, BP 87, 91003 Evry, France.

Appendix A: Stochastic model for the thermal loading

In the training procedure, uncertainties on the thermal loading are represented by a stochastic model generating zero-mean fluctuations around a reference temperature field $T_{\text{ref}} : \Omega \to \mathbb{R}$. Random temperature fields are generated following a three-step procedure based on the assumption of a linear thermal behavior:

- *Step 1:* compute temperature fluctuation modes on the solid's boundary $\partial\Omega$;
- *Step 2:* for each mode, solve the heat equation with Dirichlet boundary conditions being defined by the mode itself. Solutions of these heat equations give bulk fluctuation modes A_i^v satisfying the heat equation;
- *Step 3:* draw a random linear combination of the bulk modes, denoted by $\tau : \Omega \times \Theta \to \mathbb{R}$, and superpose it with the reference temperature field to obtain a realization of the random temperature field.

More precisely, the random field τ is defined using independent and identically distributed random variables y_i following the standard normal distribution $\mathcal{N}(0, 1)$:

$$\forall \theta \in \Theta, \qquad \forall \mathbf{x} \in \Omega, \qquad \tau(\mathbf{x}, \theta) = \sum_{i=1}^{N_{\text{modes}}} y_i(\theta) A_i^v(\mathbf{x}) \tag{35}$$

Consequently, τ is a Gaussian random field. To avoid getting unrealistic temperatures when superposing the reference temperature field with the random fluctuations, values

below zero Kelvin or beyond the melting point are truncated. The resulting random temperature field $T : \Omega \times \Theta \rightarrow \mathbb{R}$ satisfies the condition:

$$\forall \mathbf{x} \in \Omega, \qquad \mathbb{E}\left[T(\mathbf{x}, .)\right] \approx T_{\text{ref}}(\mathbf{x}) \tag{36}$$

where $\mathbb{E}$ denotes the mathematical expectation. Errors in this equation are negligible, since truncated values fall into the tail of the distribution.

The procedure for the construction of temperature fluctuation modes on $\partial\Omega$ in step 1 follows the ideas of [57]. An isotropic correlation function is defined:

$$\rho(\mathbf{x}, \mathbf{y}) = \rho(d(\mathbf{x}, \mathbf{y})) = \exp\left(-\frac{d(\mathbf{x}, \mathbf{y})}{d_0}\right) \tag{37}$$

where $d(\mathbf{x}, \mathbf{y})$ is the geodesic distance computed on the surface $\partial\Omega$. Geodesic distances are calculated thanks to the algorithm developed by Mitchell, Mount and Papadimitriou [58] and implemented in [59], see [60] for the code. In practice, geodesic distances are only evaluated between nodes of the finite-element mesh. After having computed the correlation matrix $C_{ij} = \rho(\mathbf{x}_i, \mathbf{x}_j)$ and defined a variance vector, one can get the covariance matrix $\mathbf{\Gamma}$ and find $\mathbf{A}$ such that $\mathbf{A}\mathbf{A}^T = \mathbf{\Gamma}$. The columns of $\mathbf{A}$ define the fluctuation modes on the solid's boundary.

Received: 31 October 2019 Accepted: 24 March 2020
Published online: 06 April 2020

References

1. Lumley J. The structure of inhomogeneous turbulent flows. Atm Turb Radio Wave Prop. 1967;1967:166–78.
2. Sirovich L. Turbulence and the dynamics of coherent structures, Parts I. II and III. Q Appl Math. 1987;45:561–90.
3. Chatterjee A. An introduction to the proper orthogonal decomposition. Curr Sci. 2000;78:808–17.
4. Casenave F, Akkari N, Bordeu F, Rey C, Ryckelynck D. A nonintrusive distributed reduced-order modeling framework for nonlinear structural mechanics–application to elastoviscoplastic computations. Int J Numer Methods Eng. 2020;121(1):32–53.
5. Barrault M, Maday Y, Nguyen NC, Patera AT. An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. Compt Rendus Mathemat. 2004;339(9):666–72.
6. Astrid P, Weiland S, Willcox K, Backx T. Missing point estimation in models described by proper orthogonal decomposition. Proc IEEE Conf Decis Control. 2005;53(10):1767–72.
7. Ryckelynck D. A priori hyperedution method: an adaptive approach. J Comput Phys. 2005;202(1):346–66.
8. Nguyen NC, Patera AT, Peraire J. A best points interpolation method for efficient approximation of parametrized functions. Internat J Numer Methods Engrg. 2008;73:521–43.
9. Chaturantabut S, Sorensen D. Discrete empirical interpolation for nonlinear model reduction. Decision and Control. In: proceedings of the 48th IEEE Conference 2009 held jointly with the 2009 28th Chinese control conference, CDC/CCC 2009. 2010; pp 4316–21.
10. Carlberg K, Farhat C, Cortial J, Amsallem D. The GNAT method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows. J Comput Phys. 2013;242:623–47.
11. Farhat C, Avery P, Chapman T, Cortial J. Dimensional reduction of nonlinear finite element dynamic models with finite rotations and energy-based mesh sampling and weighting for computational efficiency. Int J Numer Methods Eng. 2014;98(9):625–62.
12. Hernandez JA, Caicedo MA, Ferrer A, Cortial J. Dimensional hyper-reduction of nonlinear finite element models via empirical cubature. Computer methods in applied mechanics and engineering. 2017;313:687–722.
13. Yano M, Patera AT. An LP empirical quadrature procedure for reduced basis treatment of parametrized nonlinear PDEs. Comput Methods Appl Mech Eng. 2018;344:1104–23.
14. Iollo A, Lombardi D. Advection modes by optimal mass transfer. Phys Rev E. 2014;89:022923. https://doi.org/10.1103/PhysRevE.89.022923.
15. Cagniart N, Maday Y, Stamm B. Model order reduction for problems with large convection effects. In: Chetverushkin B, Fitzgibbon W, Kuznetsov Y, Neittaanmäki P, Periaux J, Pironneau O, editors. Contributions to partial differential equations and applications. Computational methods in applied sciences, vol. 47. Berlin: Springer; 2019.
16. Casenave F, Akkari N. An error indicator-based adaptive reduced order model for nonlinear structural mechanics—application to high-pressure turbine blades. Math Comput Appl. 2019;24:2.
17. Amsallem D, Zahr M, Farhat C. Nonlinear model order reduction based on local reduced-order bases. Int J Numer Methods Eng. 2012;92:1–31.
18. Washabaugh K, Amsallem D, Zahr M, Farhat C. Nonlinear model reduction for CFD problems using local reduced order bases. In: 42nd AIAA fluid dynamics conference. 2012. https://doi.org/10.2514/6.2012-2686
19. Bellman RE. Adaptive control processes. Princeton: Princeton University Press; 1961.
20. Lieu T, Lesoinne M. Parameter adaptation of reduced order models for three-dimensional flutter analysis. AIAA Paper. 2004;2004:888.

21. Lieu T, Farhat C, Lesoinne M. POD-based aeroelastic analysis of a complete F-16 configuration: ROM adaptation and demonstration. AIAA Paper. 2005;2005:2295.
22. Lieu T, Farhat C. Adaptation of POD-based aeroelastic ROMs for varying Mach number and angle of attack: application to a complete F-16 configuration. AIAA Paper. 2005;2005:7666.
23. Lieu T, Farhat C, Lesoinne M. Reduced-order fluid/structure modeling of a complete aircraft configuration. Comput Methods Appl Mech Eng. 2006;195:5730–42.
24. Lieu T, Farhat C. Adaptation of aeroelastic reduced-order models and application to an F-16 configuration. AIAA J. 2007;45:1244–57.
25. Amsallem D, Farhat C. Interpolation method for adapting reduced-order models and application to aeroelasticity. AIAA J. 2008;46(7):1803–13.
26. Amsallem D, Farhat C. An online method for interpolating linear parametric reduced-order models. SIAM J Sci Comput. 2011;33(5):2169–98. https://doi.org/10.1137/100813051.
27. Mosquera R, Hamdouni A, El Hamidi A, Allery C. POD basis interpolation via Inverse distance weighting on Grassmann manifolds. Discr Contin Dyn Syst Series S. 2018;12(6):1743–59.
28. Mosquera R, El Hamidi A, Hamdouni A, Falaize A. Generalization of the Neville-Aitken interpolation algorithm on Grassmann manifolds: applications to reduced order model. 2019. https://arxiv.org/pdf/1907.02831.pdf.
29. Ling J, Templeton J, Kurzawski A. Reynolds averaged turbulence modeling using deep neural networks with embedded invariance. J Fluid Mech. 2016;807:155–66.
30. Lee K, Carlberg K. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. 2019. arxiv:1812.08373.
31. Nguyen F, Barhli SM, Munoz DP, Ryckelynck D. Computer vision with error estimation for reduced order modeling of macroscopic mechanical tests. Complexity. 2018;. https://doi.org/10.1155/2018/3791543.
32. Proudhon H, Moffat A, Sinclair I, et al. Three-dimensional characterisation and modelling of small fatigue corner cracks in high strength Al-alloys. Comput Rendus Phys. 2012;13:316–27. https://doi.org/10.1016/j.crhy.2011.12.005.
33. Buljac A, Shakoor M, Neggers J, Bernacki M, Bouchard PO, Helfen L, Morgeneyer TF, Hild F. Numerical validation framework for micromechanical simulations based on synchrotron 3D imaging. Comput Mech. 2017;59:419–41.
34. Xie J, Girshick R, Farhadi A. Unsupervised deep embedding for clustering analysis. In: Proceedings of ICML'16, 478-487 (2016)
35. Guo X, Gao L, Liu X, Yin J. Improved deep embedded clustering with local structure preservation. In: Proceedings of IJCAI'17. 2017. 1753–59.
36. Moradi-Fard M, Thonet T. Deep k-means: jointly clustering with k-means and learning representations. 2018. arxiv:1806.10069.
37. Ye K, Lim LH. Schubert varieties and distances between subspaces of different dimensions. SIAM J Matrix Anal Appl. 2016;37(3):1176–97.
38. MacQueen JB. Some methods for classification and analysis of multivariate observations. In: Proceedings of 5-th Berkeley symposium on mathematical statistics and probability. 1967;1:281–97.
39. Park HS, Jun CH. A simple and fast algorithm for k-medoids clustering. Expert Syst Appl. 2009;36:3336–41.
40. Kingma DP, Ba J. Adam: a method for stochastic optimization. 2014. arxiv:1412.6980.
41. Pan SJ, Yang Q. A survey on transfer learning. IEEE Trans Knowl Data Eng. 2010;22:1345–59. https://doi.org/10.1109/TKDE.2009.191.
42. Chen J, Young B, Uy B. Behavior of high strength structural steel at elevated temperatures. J Struct Eng. 2006;132(12):1948–54.
43. Mines ParisTech and ONERA the French aerospace lab. Zset: nonlinear material & structure analysis suite. http://www.zset-software.com (1981-present.)
44. Rousseeuw PJ. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math. 1987;20:53–65.
45. Borg I, Groenen P. Modern multidimensional scaling—theory and applications. 2nd ed. Berlin: Springer; 2005.
46. de Leeuw J. Applications of convex analysis to multidimensional scaling. In: Barra JR, Brodeau F, Romier G, van Cutsem B, editors. Recent developments in statistics. Berlin: Springer; 1977. p. 133–45.
47. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: machine learning in python. J Mach Learn Res. 2011;12:2825–30.
48. Haykin S. Neural networks—a comprehensive foundation. Second edition. 1999;351–91.
49. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ, eds. Advances in neural information processing systems 25. Curran Associates Inc; 2012. p. 1097–105. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.
50. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014. arxiv:1409.1556
51. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: The IEEE conference on computer vision and pattern recognition (CVPR). 2016.
52. Chollet F, et al. Keras. 2015. https://keras.io
53. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org. 2015. https://www.tensorflow.org/
54. Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, Liu T, Wang X, Wang G, Cai J, Chen T. Recent advances in convolutional neural networks. Patter Recogn. 2018;77:354–77.
55. Lin M, Chen Q, Yan S. Network in network. CoRR abs/1312.4400. 2013.
56. Lin T, Goyal P, Girshick R, He K, Dollar P. Focal loss for dense object detection. In: IEEE transactions on pattern analysis and machine intelligence. 2018.

57. Scarth C, et al. Random field simulation over curved surfaces: Applications to computational structural mechanics. Comput Methods Appl Mech Engrg. 2018;. https://doi.org/10.1016/j.cma.2018.10.026.
58. Mitchell JSB, Mount DM, Papadimitriou CH. The discrete geodesic problem. SIAM J Comput. 1987;16(4):647–68.
59. Surazhsky V, Surazhsky T, Kirsanov D, Gortler SJ, Hoppe H. Fast exact and approximate geodesics on meshes. ACM Trans Graph. 2005;24(3):553–60.
60. Kirsanov D, Malhotra G, Knock S. gdist 1.0.3. https://pypi.org/project/gdist/. 2013.

Publisher's Note