Deep multimodal autoencoder for crack criticality assessment

Hugo Launay, David Ryckelynck, Laurent Lacourt, Jacques Besson, Arnaud Mondon, François Willot

Deep Multimodal autoencoder for crack criticality assessment

H. Launay[a], D. Ryckelynck[a,*], L. Lacourt[a], J. Besson[a], A. Mondon[a], F. Willot[a,b]

[a]*Mines Paris, PSL research university, Centre des Matériaux, CNRS UMR 7633, 91003 Évry, France*
[b]*Mines Paris, PSL research university, Centre de Morphologie Mathématique, 77300 Fontainebleau, France*

Published in International Journal of Numerical Engineering 123:6 pp. 1456–1480 (2022).

Abstract

In continuum mechanics, the prediction of defect harmfulness requires to solve approximately partial differential equations with given boundary conditions. In this contribution boundary conditions are learnt for tight local volumes (TLV) surrounding cracks in 3D volumes. A non-parametric data-driven approach is used to define the space of defects, by considering defects observed via X-Ray computed tomography [1, 2]. The dimension of the ambient space for the observed images of defects is huge. A nonlinear dimensionality reduction scheme is proposed in order to train a reduced latent space for both the morphology of defects and their local mechanical effects in the TLV. A multimodal autoencoder [3] enables to mix morphological and mechanical data. It contains a single latent space, termed mechanical latent space. But this latent space is fed by two encoders. One is related to the images of defects and the other to mechanical fields in the TLV. The latent variables are input variables for a geometrical decoder and for a mechanical decoder. In this work, mechanical variables are displacement fields. The autoencoder on mechanical variables enables projection-based model order reduction as proposed in [4]. The main novelty of this paper is a submodeling approach assisted by artificial intelligence. Here, for defect images in the test set, Dirichlet boundary conditions are applied to TLV. These boundary conditions are forecasted by the mechanical decoder with a latent vector predicted by the morphological encoder. For that purpose, a mapping is trained to convert morphological latent variables into mechanical latent variables, denoted "direct mapping". An "inverse mapping" is also trained for error estimation with respect to morphological predictions. Errors on mechanical predictions are close to 5% with simulation speed-up ranging for 3 to 120. We show that latent variables forecasted by the images of defects are prone to a better understanding of the predictions.

Keywords: Multimodal autoencoder; CNN; Transfer Learning; Finite Element Method; Digital Twin

1. Introduction

The assessment of cracked structures is still an active field of research in mechanical engineering [5]. Charts [6] can be used to estimate some quantities of interest on cracked structures but they are not accurate enough and are focused on specific cases. Finite element methods can be used to improve results and to deal with arbitrary crack configurations. Chart-based and other equivalent methods give a criterion for deciding if a component is safe or unsafe. Although this is sufficient in practice for deciding if a component must be

*Corresponding author
URL: david.ryckelynck@mines-paristech.fr (D. Ryckelynck)

rejected, these approaches are less robust than methods making use of full-field numerical computations, that provide more information as well as an understanding of the mechanical phenomena involved. Nevertheless such an approach leads to prohibitive computation times due to the fact that the crack needs to be meshed very finely. In this work focus is made on planar cracks in 3D volumes.

Moreover mechanical data are expected to fulfill partial differential equations (PDE) and neural networks have been proposed to solve partial differential equations in [7]. Machine learning algorithms have recently been proposed to analyse the criticality of defects for pipelines [8], in the aeronautical field [9] and for fabrication process [10]. Besides machine learning and numerical simulation have proven their complementarity for example for the study of defect in ball bearings [11]. The Simulation-driven machine learning approach is very attractive when we have mechanical models at our disposal. The use of machine learning techniques enables us to get rid of the parametrization of the object to model. This is crucial when considering defects observed by imaging techniques.

Besides the efficiency of multimodal autoencoder (MMAE) have recently been shown [12, 13]. The aim of this method is to use different inputs such as sound, text or images for example and reconstruct them after having compressed them in a common smaller latent space [3]. In the online phase a single input channel can be used to reconstruct all the output channels, this is denoted the cross-modal reconstruction.

In the present work, we show that morphological data and mechanical data related to defects are sharing a common latent space designed using MMAEs. In this paper a digital twin of a specimen containing non parametric cracks is considered. A multimodal autoencoder is trained with two different channels being the 2D crack image and its associated 3D displacement field. In the online phase the 2D image is used as input to reconstruct the 2 channels. The reconstructed displacement field is then transferred on the external surface of a tight local volume (TLV) and used as boundary condition of a small finite element problem. Since the boundary conditions are learnt, a full-field numerical computations on the entire system is not required. This results in better speed-up compared to a submodelling approach where additional simulations at the global scale have to predict these boundary conditions. Moreover, the bounding box in a submodel approach must be rather large so that the crack-induced fluctuations on the boundary can be neglected. An accurate prediction of the boundary conditions, including for small subdomains, is among the main goals of the presently-developed MMAE model. Also, contrary to surrogate models, we carry out classical finite element computations in the subdomain. This approach provides much more information on the mechanical response induced by the crack, that may in turn be interpreted.

The reconstructed 2D image will be used to define an error indicator. We then determine whether a linear relationship exists between the 2D image reconstruction and the error committed on 3D displacement field reconstruction.

2. Dataset and mechanical problem

In this work a tensile test is simulated on a specimen containing a crack inside the bulk material. The size of the crack compared to that of the specimen is not negligible and may have an impact larger than 20% on the limit load. The position of the crack varies slightly in the middle plane of the specimen but the shapes and sizes of the cracks vary greatly.

The defect dataset that is used in this study are random 2D planar cracks. They are generated by taking slices of 3D defects obtained from tomography [14, 2]. The resulting defects are therefore 2D images representing non-parametric cracks. Hence it is important

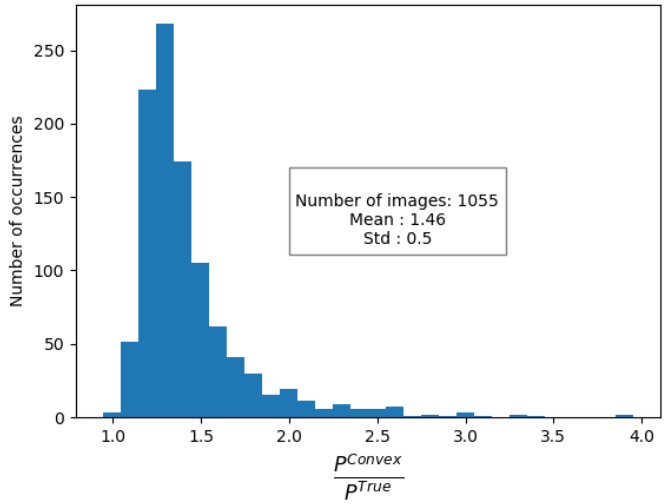


Figure 1: Histogram of the ratio between the perimeter of the convex envelope surface (P^{convex}) and that of the shape (P^{true}).

to keep in mind that this dataset is purely artificial and is not based on actual images of cracks. A large variety of cracks is obtained with various shapes and, for some of them, the 2D image containing the defect contains more than one crack. Since every crack is created with a 2D image, they are all planar cracks. A histogram measuring how close the defects are to a convex shape is represented in Fig. (1). The convexity criterion is the ratio between the perimeter of the crack, seen as a shape in a plane, and that of the convex envelope, included in the same plane.

As shown in Fig. (1), a large percentage of the cracks are strongly non-convex. About 25% have a convexity ratio larger than 1.5. These cracks are inserted in a 3D specimen shown in Fig. (2a). The information contained at the ends of the full specimen is poor and the mesh is coarse therefore partial differential equations and their finite element approximation are evaluated on the domain delimited by the red box shown on Fig. (2.a). This box is a parallelepiped of dimension $200 \times 100 \times 34\ mm^3$. The mesh contains about 170, 000 degrees of freedom (but varies with the crack shape). The elements used are reduced linear tetrahedrons. The ratio between the size of the elements on the ends and near the crack is around 40. The Hausdorff parameter [15] used to control the remeshing is equal to 0.3. Fig. (2b) and Fig. (2c) show the meshed zone of interest of the specimen and a cut view to see the size of the mesh around the crack.

In the remainder of this study, the cracks are in the middle plane but the crack is not necessarily centered in the box. Fig. (3) shows three images of cracks and below them the insertion of the crack in the 3D structure for defects 4, 21 and 22 (note that only the free surfaces are shown). For common crack shapes such as a penny shape, the limit load can be analytically evaluated in such simple structures. But for non-parametric random cracks or for structures containing multiple cracks that interact with each other, analytical results cannot be obtained in general. Finite element simulation is an option to compute those quantities though it usually leads to huge computation time. Despite the fact that the defect is large enough to have an important effect on the structure, it needs to be meshed finely since its size is much smaller than the whole structure. Hence elements around the cracks are much smaller than elements on the border of the domain Ω as shown in Fig. (3). Moreover problems such as volumetric locking and non convergence might appear due to the localization at the

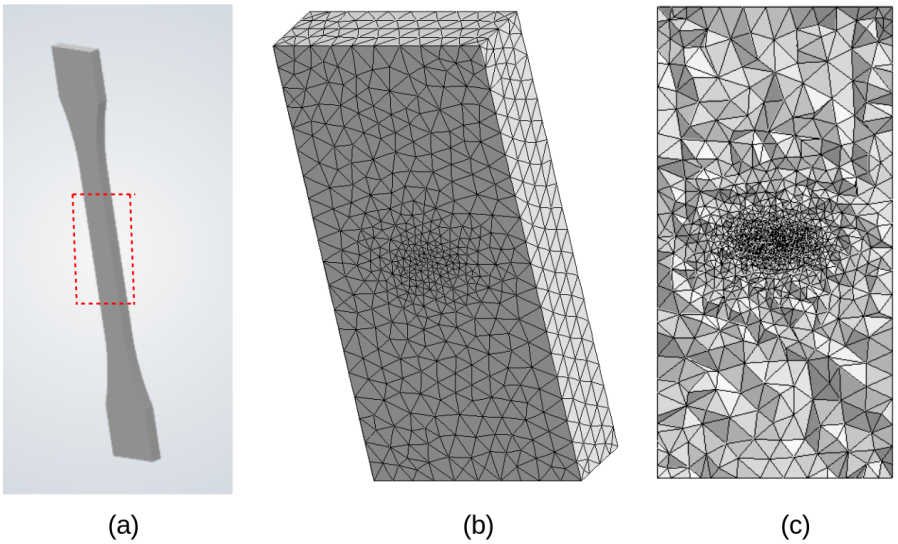


Figure 2: (a) Traction specimen example, (b) meshed zone of interest, (c) cut view of the meshed zone of interest.

crack front [16, 17].

In this study a nonlinear problem that involves finite strains and plasticity is considered. The Zset software [18, 19, 20, 21] is used. The primal variable of finite element models in mechanics is the displacement field. In the finite element (FE) model [22], the shape functions of the FE basis are denoted by $(\phi_j)_{j=1}^{n}$ with n the number of discretization nodes. It is convenient to introduce the functions $\boldsymbol{\phi}_i = \phi_j \vec{e}_k$ where $i = (j-1) \times d + k$, $i = 1, ..., \mathcal{N}$, $j = 1, ..., n$, $k = 1, ..., d$, d is the dimension of the problem and $\vec{e}_k$ refer to the canonical vectors of a Cartesian coordinates system. The approximation of the displacement with the shape functions $(\boldsymbol{\phi}_i)_{i=1}^{\mathcal{N}}$ reads:

$$\boldsymbol{v}(\boldsymbol{x}) = \boldsymbol{v}_0(\boldsymbol{x}) + \sum_{i=1}^{\mathcal{N}} \boldsymbol{\phi}_i(\boldsymbol{x}) u_i, \qquad \boldsymbol{x} \in \Omega, \tag{1}$$

where $\mathcal{N} = n \times d$ is the number of degrees of freedom (DOF) of the structure, $\boldsymbol{v}_0$ is a given displacement field that fulfills Dirichlet boundary conditions, $\boldsymbol{v}$ is the approximate finite element solution and $\vec{u} = (u_i)_{i=1}^{\mathcal{N}}$ the vector of the related degrees of freedom. For this work a large strain formulation which uses a logarithmic strain measurement is used [23, 24]. The mechanical equations for the prediction of the displacement field then read:

$$\text{Strain gradient decomposition:} \quad \underset{\sim}{F} = \underset{\sim}{I} + \underset{\sim}{\nabla}\vec{u} \tag{2a}$$

$$\text{Logarithmic strain:} \quad \underset{\sim}{E} = \frac{1}{2}\log(\underset{\sim}{F}^T.\underset{\sim}{F}) \tag{2b}$$

$$\text{Elastic/plastic partition is supposed:} \quad \underset{\sim}{E} = \underset{\sim}{E}^e + \underset{\sim}{E}^p \tag{2c}$$

$$\text{Constitutive law in logarithmic space:} \quad \underset{\sim}{T} = \underset{\approx}{C} : \underset{\sim}{E}^e \tag{2d}$$

$$\text{Second Piola-kirchhoff stress:} \quad \underset{\sim}{S} = \underset{\approx}{P} : \underset{\sim}{T} \tag{2e}$$

$$\text{Second invariant of the stress tensor:} \quad J_2(\underset{\sim}{S}) = \sqrt{\frac{3}{2}\ \mathrm{dev}(\underset{\sim}{S}) : \mathrm{dev}(\underset{\sim}{S})} \tag{2f}$$

$$\text{Yield stress:} \quad R_0 \tag{2g}$$

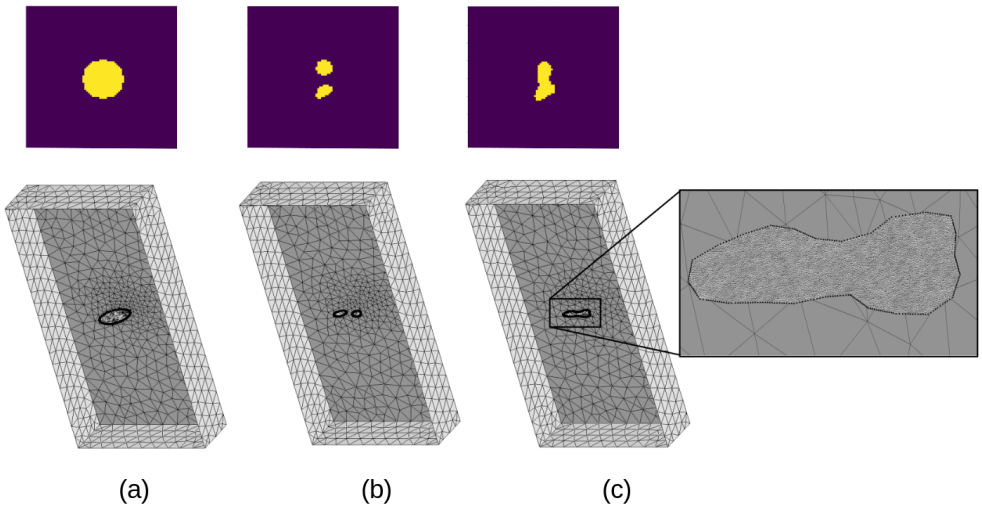


Figure 3: 2D cracks and insertion in 3D specimen for (a) defect 4,(b) defect 21 and (c) defect 22.

$$\text{Yield function:} \qquad f(\underset{\sim}{S}, R_0) = J_2(\underset{\sim}{S}) - R_0 \tag{2h}$$

$$\text{Relation with global Cauchy stress:} \qquad \underset{\sim}{\sigma} = \det(\underset{\sim}{F})^{-1} \underset{\sim}{F} \cdot \underset{\sim}{S} \cdot \underset{\sim}{F}^T \tag{2i}$$

More details on the projector $\underset{\approx}{P}$ and the dual stress $\underset{\sim}{T}$ of the logarithmic strain $\underset{\sim}{E}$ are given in [23, 24]. The material considered here is similar to aluminium. The Young modulus used for this simulation is $E = 70,000$ MPa, the Poisson coefficient is $\nu = 0.3$ and the elasticity limit is $R_0 = 400$ MPa. Hardening is not accounted for in this study. This is the reason why it is desirable to assist a submodel construction by artificial intelligence, via the selection of approriate boundary conditions.

In order to save useful data around the crack, a parallelepipedic box is defined and is refered to as the encoding mesh. With the goal of designing a common ambient space for mechanical data, the solution of the full specimen is transferred on it. Since the encoding mesh is much smaller and located at the same place as the crack, only the field around the crack is transferred on it. The encoding mesh, containing $64 \times 64 \times 64$ nodes, is defined using regular hexahedral elements and therefore is equivalent to a 3D image with each element representing a voxel. This allows using classical computer vision tools. Object detection techniques [25] can be used to obtain the cracks position. In the present work, all cracks are located in the same subdomain. Cracks are obtained from a database of 2D images and inserted in the subdomain randomly, so that they are not necessarily centered in the subdomain. The zone of interest where the crack is located is accordingly always the same. To treat a more general problem where the crack position in the entire domain and its orientation were random, additional parameters defining the position and orientation of the subdomain would be required. Fig. (4) illustrates the process needed to obtain the encoded field that contains the information around the crack. From left to right the figure 4 shows the 2D image representing the crack, the insertion of the 2D crack in the 3D mesh, the logarithmic strain along the vertical axis obtained with a finite element simulation (denoted $\underset{\sim}{E}_{22}$ as in eq.2), the position of the encoding mesh in the full structure and the encoding mesh containing the transferred information.

Simulation on the full specimen are only made to generate the input arrays for the training of the MMAE. During the online phase the MMAE generates a displacement field on the

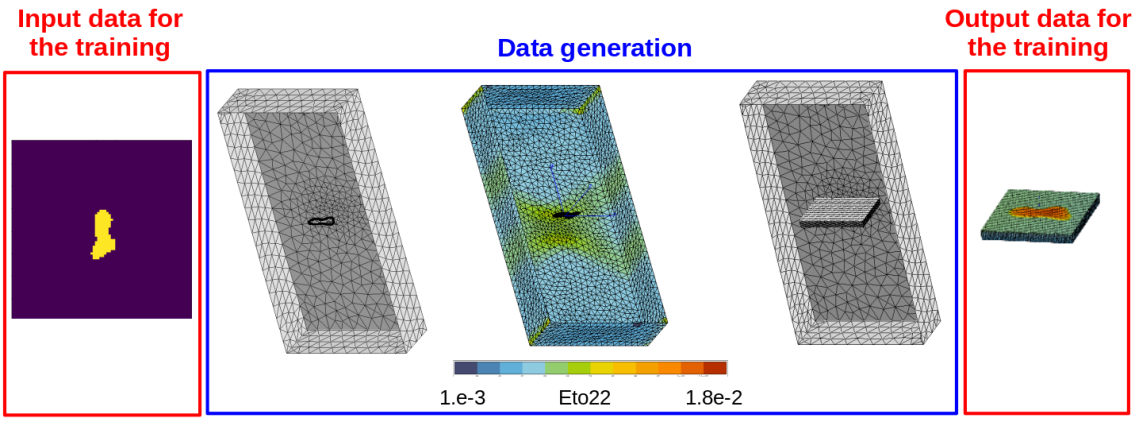


Figure 4: (a) 2D image representing the crack, (b) insertion of the 2D crack in the 3D mesh, (c) $\underset{\sim}{E}_{22}$ field, (d) position of the encoding mesh in the full structure and (e) the encoding mesh with the transferred information.

encoding mesh thanks to an input image of a crack. This displacement field on the encoding mesh is then transfered on a subdomain extracted from the full mesh.

The mechanical equations on the displacement $\vec{u}(t)$ read:

$$\begin{aligned} \text{div}[\sigma(\underset{\sim}{\vec{u}}(t))] &= \vec{0} \qquad \text{applied on } \Omega \\ \vec{u}(t) &= \vec{u}_0(t) \quad \text{applied on } \Gamma^{BC} \\ \underset{\sim}{\sigma}(\vec{u}(t)) \cdot \vec{u} &= \vec{0} \qquad \text{applied on } \Gamma \end{aligned} \tag{3}$$

where Ω represents the whole domain, Γ^{BC} represents the surface where the Dirichlet boundary conditions are applied and Γ represents the free surfaces where no boundary conditions are applied (including the free surface of the crack lips). The tensor $\sigma(\vec{u}(t))$ is obtained with the relation detailed in Eq. (2). In order to solve the partial differential equation described in Eq. 3, a quasi-static approach is used. The scheme shown on Fig. (4) is applied on the whole dataset of 2D images to generate 3D displacement fields expressed on the encoding mesh. The defect dataset contains 1,055 2D images. Finite element predictions of the displacement field have been performed for each defect. The mechanical dataset contains all the displacement fields associated to their respective 2D images. They are saved on the encoding mesh. The encoding mesh is represented as a array of dimension 5. The first three dimensions represent the spatial dimensions (x, y, z). The fourth dimension refers to the considered field either U_x, U_y or U_z. The fifth dimension, denoted t, is the time. In this study the time is assimilated to the incremental loading parameter and varies from 0 to 1. The displacement field is saved along the encoding mesh in a 5-dimensional array $T^{(j)}$ so that $T^{(j)}[x, y, z, i, k] = U_i^{(j)}(x, y, z, t_k)$ where $i \in \{x, y, z\}$ and $U_i^{(j)}(x, y, z, t_k)$ is the displacement in direction i at point (x, y, z) and time t_k for defect j. The tight local volume (TLV) is a part of the full mesh. It is extracted around the crack where the needed information is located. The TLV model gives access to all standard mechanical variables by using finite element modelling. This approach enables transfer learning in the sense that all the modelling capabilities of the finite element method do not need to be retrained and constitutes a huge asset of the developed method. As such, the model does not need to be trained again when new quantities of interest or new element types are used.

3. Multimodal autoencoder and transfer learning

As explained in [26], feed forward neural network (FNN) is a classical architecture in machine learning.A deep FNN contains many layers of artificial neural networks. The data is given to an input layer and then pass through some hidden layers. The last layer is called the output layer and gives the prediction. The weights w make a full the connection between the neurons from different layers. During the prediction phase, the data flows in one way from the input layer to the output layer. In the training phase, the global error defined by the mean-squared difference between the target value and the FNN output will be back-propagated through the hidden layers. This step is performed in order to update the weights, where the objective is to minimize the global error. An activation function is attached to each neuron. The output of each neuron is computed by multiplying the outputs from the previous layer with the corresponding weights. For the neuron j in the layer k, the data of the previous layer $k-1$ is summed up and then altered by an activation function. The output of the neuron j in layer k is computed as:

$$o_j^k = f\left(\sum_{i=1}^{N_{k-1}} w_{ij} o_i^{k-1} + b_i^{k-1}\right), \tag{4}$$

where N_{k-1} is the number of neurons in the previous layer $k-1$, w_{ij} is the weight connecting neurons i and j, o_i^{k-1} is the output of the neuron i in layer $k-1$. A common choice for the activation function is the rectified linear function ("ReLu"):

$$f(x) = \begin{cases} x, & \text{if } x \geq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

The specific architecture of the FNN, such as the number of layers and the number of neurons in each layer, has to be determined according to the complexity of the dataset. The global error, also called loss function or network performance, is defined according to the difference between the network prediction and the target data. The mean squared error can be used to measure the loss (but many other loss function can be used):

$$E(\vec{w}) = \frac{1}{N_o} \sum_{i=1}^{N_o} [o_i(\vec{w}) - t_i]^2, \tag{6}$$

where N_o is the number of outputs, o_i is the i^{th} output, $\vec{w}$ is the vector that contains the weights of neural network, and t_i is the i^{th} target value. Training a feed forward neural network is an optimization problem, where the global error is treated as the objective function.

One way to minimize the global error is to use the gradient descent algorithm to update the weighs [27] with the following:

$$\vec{w}^{n+1} = \vec{w}^n - \mu \frac{\partial \vec{E}(\vec{w}^n)}{\partial \vec{w}^n}. \tag{7}$$

The term $\frac{\partial \vec{E}(\vec{w}^n)}{\partial \vec{w}^n}$ is evaluated with the back propagation [28]. In the training process, many iterations are required to update the weights until the stopping criteria is fulfilled, where one iteration is also known as one epoch. Since 2D images and 3D displacement fields are considered in this work use of convolutional neuron networks is made. Good results for 2D and 3D images have been obtained with this method [29]. Details on convolutional layers are given in [4].

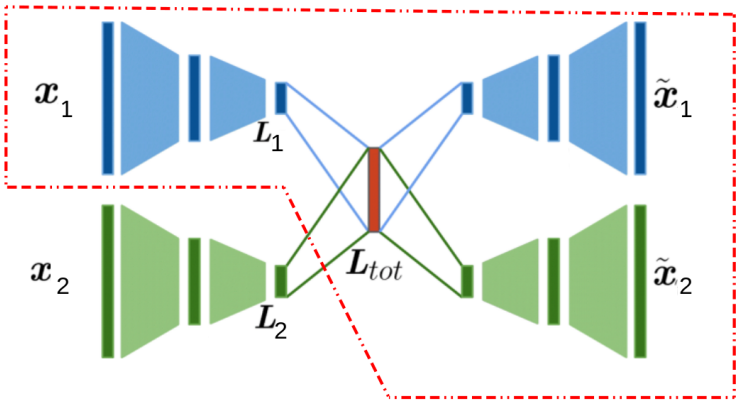


Figure 5: Multimodal autoencoder representation.

Autoencoders enable to compress the input information in a much smaller latent space and reconstruct it afterwards. This allows one to define a small latent space in which all the information is condensed [30]. Many applications of this bottleneck effect exist: clustering the data, denoising data or else exploring the smaller space. An autoencoder employs an input layer and an output layer connected by one or more hidden layers. The output layer has the same number of nodes (neurons) as the input layer. Its purpose is to reconstruct its inputs (minimizing the difference between the input and the output) instead of predicting a target value Y given inputs X . Therefore, autoencoders are selfsupervised learning models. They do not require labeled inputs to enable learning [31]. The feature space $\mathcal{F}$ should have lower dimensionality than the input space $\mathcal{X}$. An autoencoder consists of two parts, the encoder and the decoder, which can be defined as transitions Φ and Ψ such that:

$$\begin{aligned} \Phi &: \mathcal{X} \to \mathcal{F}, \\ \Psi &: \mathcal{F} \to \mathcal{X}, \\ \Phi, \Psi &= \underset{\Phi,\Psi}{\operatorname{argmin}} ||X - (\Psi \circ \Phi)X||^2, \\ \dim(\mathcal{F}) &\ll \dim(\mathcal{X}). \end{aligned} \tag{8}$$

Multimodal autoencoders (MMAE) category, which is a specific branch of autoencoders, are detailed in [3]. The main idea of MMAE is to take various inputs from different nature (such as 2D images and 3D images) and to compress them in the same latent space before rebuilding them. One of the main asset of MMAE it to remove some input channel during the online phase in order to reconstruct all of the channels [32] as the red border suggests on Fig. (5). x_i are the different input channels, $\widetilde{x}_i$ are the different output channels and L_i is the size of the latent space.

The aim of this article is to take advantage of this multi-modal aspect in order to train a mapping between morphological data and mechanical predictions as a surrogate model for fast mechanical prediction. A standard convolutional neuron network could be used to associate a 3D displacement field to a 2D image. But with a MMAE approach it is possible to interpret how features are represented in the latent space. Moreover an innovative point of the study is to use the reconstructed 2D image in order to create an error indicator depending on the training set to evaluate whether the 3D displacement field is accurate or not. The proposed MMAE has four modalities, in the sense that it has four encoders and four related decoders. This architecture is motivated by the dimension of the displacement field on the encoding

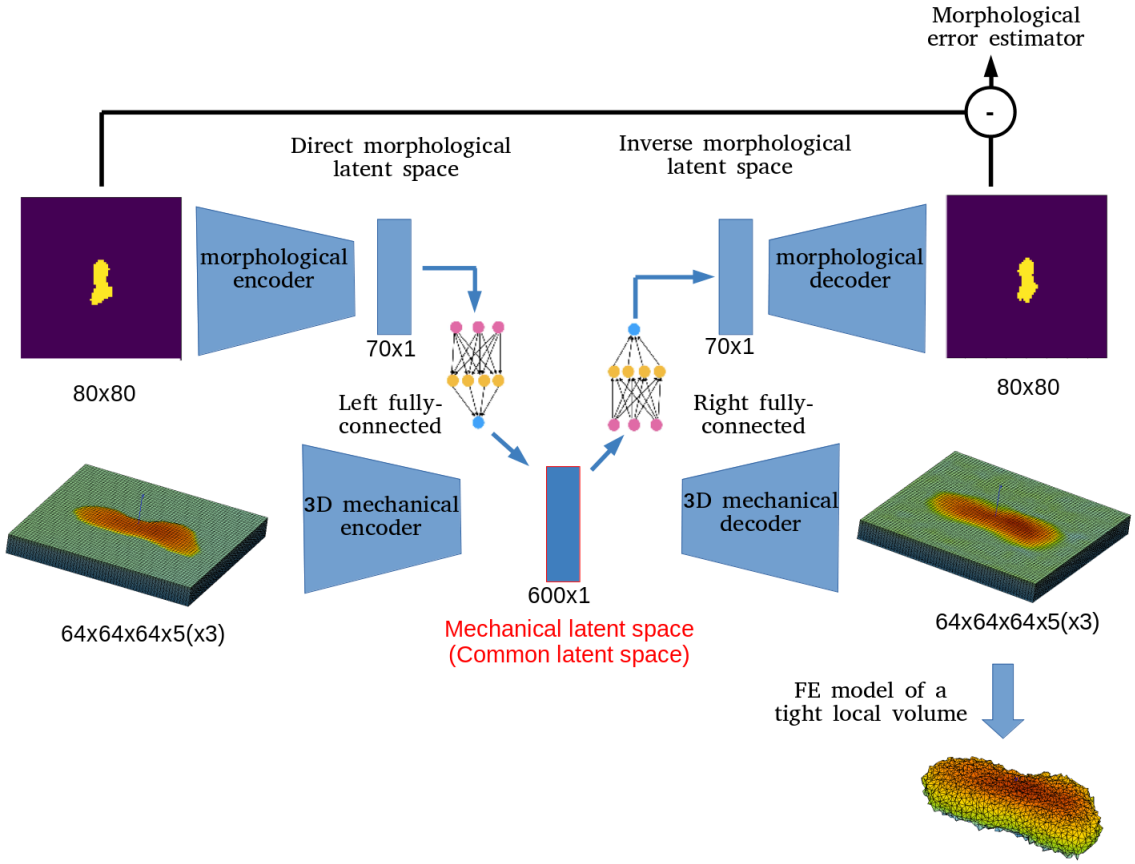


Figure 6: Modified MMAE approach and error indicator to generate displacement field on TLV.

mesh. Indeed this input contains the spatial information (3 dimensions), the temporal information and the field information (U_x,U_y or U_z) which results in an array of dimension 5. In order to reshape the dimension of the array into the usual dimensions of data in computer vision, the three displacement fields are treated with three different modalities, using three dedicated encoders and decoders.

The outputs of these three mechanical encoders are connected to the three inputs of the mechanical decoders via a mechanical latent space by using dense layers, termed mechanical dense layers, behind and in front of the latent layer. The mechanical latent space is a three dimensional space. It has latent variables for each dimension of the displacement field and there is no connection between these dimensions in the mechanical dense layers in front of the 3D latent space and behind it. The dimension of the latent space is 600. There are 200 latent variables per component of the displacement field. Latent variables related to each dimension of the displacement field are stacked together to create a common mechanical latent space.

The morphological encoder and decoder are connected to the mechanical latent space by two additional dense layers, termed multimodal dense layers, behind and in front of the three dimensional latent space. The inputs of the morphological encoder, being the 2D image, is 80×80 whereas for the displacement field is of size $64 \times 64 \times 64 \times 5(\times 3)$. The $(\times 3)$ is between parenthesis to express that the three components of the displacement fields have separated modalities. This architecture is represented on Fig. (6). The 4 encoders and decoders are coupled together by the multimodal dense layers, named left fully connected and right fully connected in Fig. (6). This architecture is non conventional and constitutes a important point of this paper.

During the training phase of the MMAE we have noticed that the 3D displacement chan-

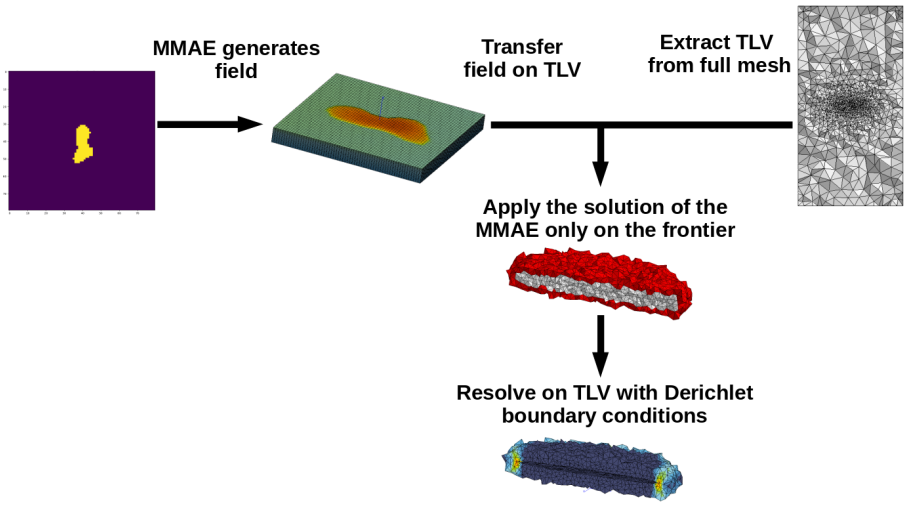


Figure 7: Online scheme

nels are the hardest to fit. Hence a transfer learning is accordingly carried out in two steps, the first step being used to train separate autoencoder for each modality and the second step being used to train the weights of the muldimodal dense layers. In the second step, the weights of convolutional layers and those of the mechanical dense layers are kept as constant. Then we know the latent variables of the proposed MMAE at the end of the first step of the transfer learning strategy. We also know morphological latent variables. Hence the multimodal dense layers, left fully connected and right fully connected in Fig. (6), are trained separately. The left fully-connected layer is trained to map the morphological latent coordinates, expressed as a 70-components vector, to the mechanical latent coordinates, expressed as a 600-components vector. The right fully-connected layer is trained as an inverse mapping that maps the mechanical latent coordinates to the morphological latent coordinates.

Using MMAE, the morphological as well as mechanical latent coordinates can be obtained from an image of a crack, and the associated displacement field can be predicted. The latent variable in the mechanical latent space can be used to recover the input image from the right fully-connected network. The discrepancy between the input and reconstructed images may then be used to define an error indicator. Such error indicator is meaningful since the reconstructed image and the displacement field come from a common latent space. Also, use of a 2D crack is made to forecast a 3D displacement field on the encoding mesh. In parallel the TLV is extracted from the full mesh of the target defect. The 3D displacement field from the encoding mesh gives access to the Dirichlet boundary conditions applied on the TLV. A standard FE prediction is then made with the displacement field on the external surface applied as Dirichlet boundary conditions. Fig. (7) illustrates this methodology.

For nodal fields, the transfer operator is straightforward. First, each node of the new mesh, having X_{new} as coordinates, is localized in the old mesh, then the shape functions ϕ_i^{old} of the old mesh are used to compute the value of the field f on the new mesh:

$$f(X_{new}) = \sum_i \phi_i^{old}(X_{new}) \cdot f(X_i), \tag{9}$$

where X_i is the coordinate of the vertex i of the corresponding element. The error due to the transfer has been measured by transferring the displacement field of the encoding mesh as Dirichlet boundary conditions of the TLV without using the MMAE. The error

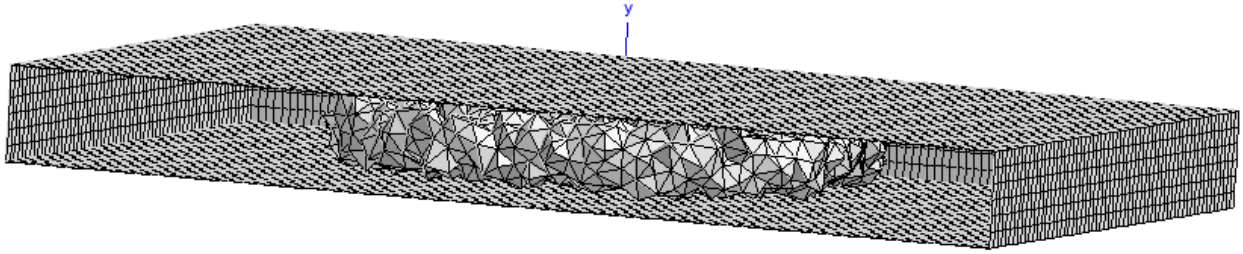


Figure 8: $\Omega_{TVL} \subseteq \Omega_{EM}$.

results of the two transfers, the first one from the full mesh to the encoding mesh and the second one from the encoding mesh to the TLV. The error is evaluated on the cumulated plasticity (see Eq. 10). The error that is used is an extension of the normalized mean squared error to continuous functions. Compairing the error based on the displacement and these of the cumulated plastic strain on cases of the training set it appeared that the first error measurement is more conservative than the second.

$$\eta = \frac{\frac{1}{\int_{\mathrm{d}\Omega}} \int (\Delta\varepsilon^{pcum})^2 \mathrm{d}\Omega}{\frac{1}{\int_{\mathrm{d}\Omega}} \int (\varepsilon^{pcum}_{FOM})^2 \mathrm{d}\Omega} = \frac{\int_\Omega (\Delta\varepsilon^{pcum})^2 \mathrm{d}\Omega}{\int_\Omega (\varepsilon^{pcum}_{FOM})^2 \mathrm{d}\Omega}, \qquad \text{where } \Delta\varepsilon^{pcum} = \varepsilon^{pcum}_{FOM} - \varepsilon^{pcum}_{Model}, \tag{10}$$

and where Ω describes the TLV domain, the subscript "FOM" refers to the Full Order Model simulation and the subscript "Model" refers to the solution obtained with the MMAE approach detailed in this paper. After computing this error for all the defects of the dataset the mean error has been computed and is equal to 0.07% and the standard deviation is 0.07%, this shows that the transfer operations does not generate much error. Importantly, the TLV must be included in the encoding mesh otherwise the field transfer from the encoding mesh to the TLV cannot be done properly. The latter reads:

$$\Omega_{TVL} \subseteq \Omega_{EM} \tag{11}$$

where Ω_{EM} refers to the encoding mesh domain and Ω_{TLV} to the tight little volume domain. Fig. (8) illustrates this necessary condition.

4. Results and interpretability

The displacement field shows varying orders of magnitude depending on time and displacement component. The MMAE is accordingly trained on the following rescaled field $\widetilde{T^{(j)}}$:

$$\begin{aligned} \widetilde{T^{(j)}}[x,y,z,i,k] &= \frac{T^{(j)}[x,y,z,i,k] - m_{ik}}{M_{ik} - m_{ik}}, \\ M_{ik} &= \max_{x,y,z,j} T^{(j)}[x,y,z,i,k], \\ m_{ik} &= \min_{x,y,z,j} T^{(j)}[x,y,z,i,k] \end{aligned} \tag{12}$$

for a given defect j. Note that since the images of defects are binary images, these do not need to be rescaled.

As explained in the previous section the four autoencoders have been trained separately (a single 2D AE and three 3D AE) and two dense neuron networks have been trained afterwards

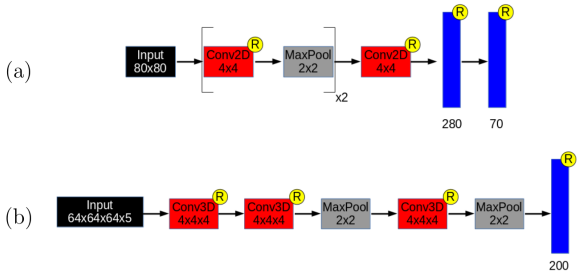


Figure 9: 2D (a) and 3D (b) encoder architectures.

in order to map the morphological and mechanical latent spaces. Since the 3D displacement field data are 4 dimensional (time and space) the filter channel (associated to the three color red, green and blue) of the 3D convolution layers is used as another dimension enabling to treat 4D data. In our case the filter channel will contain the displacement field (Ux,Uy or Uz). Classical CNN architectures are presented in [33] and the same notation is used to present the developed architecture. The 2D encoder is composed of two blocks of convolution and max pooling followed by an additional convolution. Then two dense layers lead to the latent space. For each layer the activation function is rectified linear (ReLu). Fig. (9) illustrates the architecture of the 2D and 3D encoder. The yellow R symbolises that the activation function is ReLu. The decoding part is exactly the same in the reverse way and the "max pooling" blocks are replaced by "up sampling" blocks.

In this work the ReLu activation function has been used. This activation function sometimes generates gradient losses during the training when not large enough dataset are used. We emphasize that the Exponential Linear Unit (ELU) activation function as an alternative. More details on the architectures such as the number of filters or the number of trainable parameters are given in Appendix A. The loss function used for each optimization is a L2 norm and the optimizer used is adam. Fig. (10) shows the loss in function of the epoch for the optimization process.

The position in Ω_{EM} where the error on the displacement field made by the MMAE in average can be evaluated with Eq. (13).The nodes where the error $\bar{\zeta}$ is superior to 1% are shown in red in Fig. (11). A circular crack with the mean diameter of the whole data set is also drawn, so that:

$$\zeta = \frac{|U_{FOM} - U_{MMAE}|}{U_{mean}}, \qquad \bar{\zeta} = \frac{\sum_{i=1}^{N_{defect}} \zeta_i}{N_{defect}} \tag{13}$$

where N_{defect} is the number of defects,U_{FOM} and U_{MMAE} refer to the solution for the displacement field of the full order model and of the MMAE approach respectively. U_{mean} is the mean displacement in the encoding mesh.

The error is located on the crack front and more precisely on the two closest sides from the front and the back of the specimen. For each 2D crack images the corresponding reconstructed 2D crack image and a reconstructed 3D displacement field is generated. Fig. (12) shows the reconstructions for defects 122, 131 and 16 which are data from the training set.

One of the reason why MMAE are used in this work is to see whether a relation between the error made during the online phase and the 2D image reconstruction can be made. The information of the input image is encoded in the common latent space and so gives a global indication, for both morphological and mechanical data, that would not be obtained if solely the morphological AE was used. During the online phase the displacement field generated on

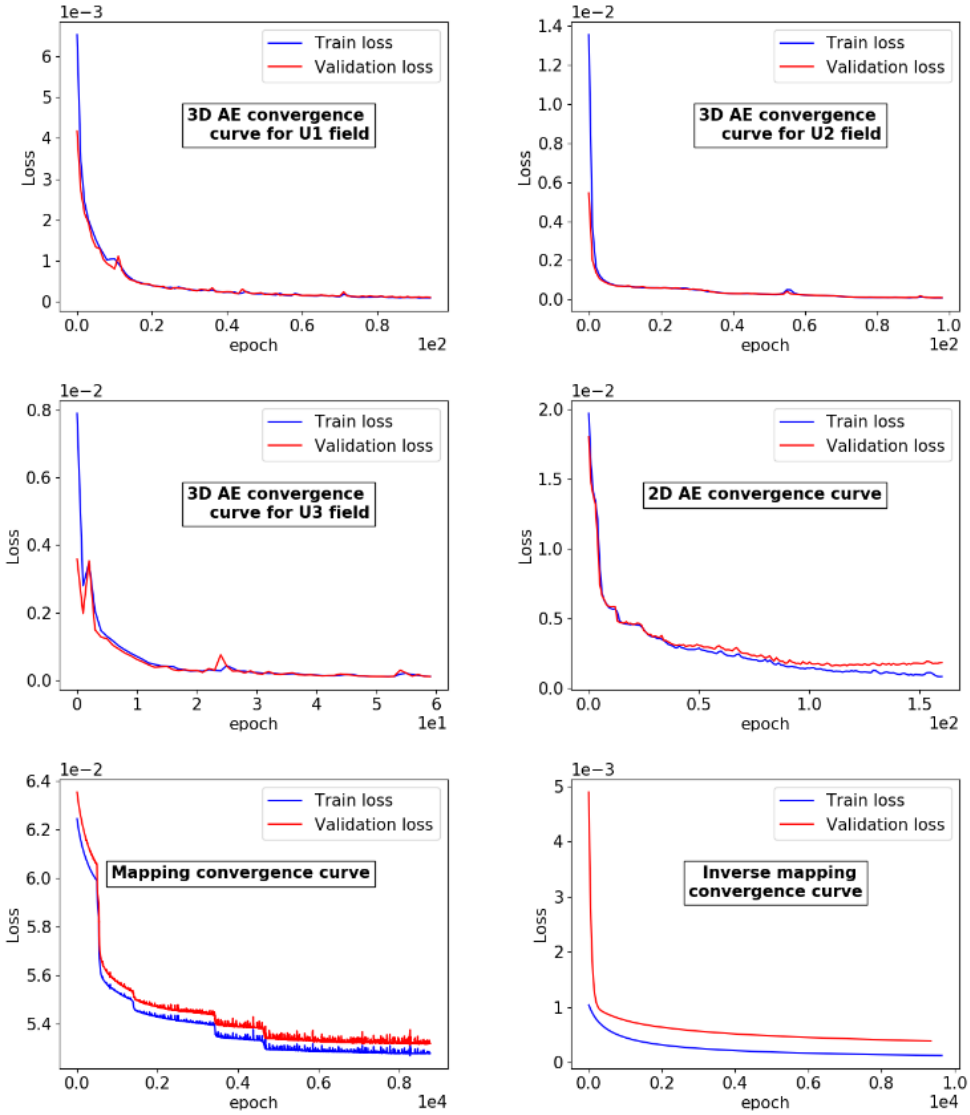


Figure 10: Loss curve for each optimization.

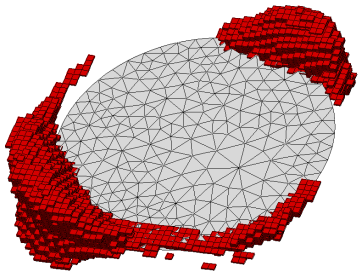


Figure 11: Element of the encoding mesh where $\bar{\zeta} > 1\%$ (red) with mean circular crack (grey)

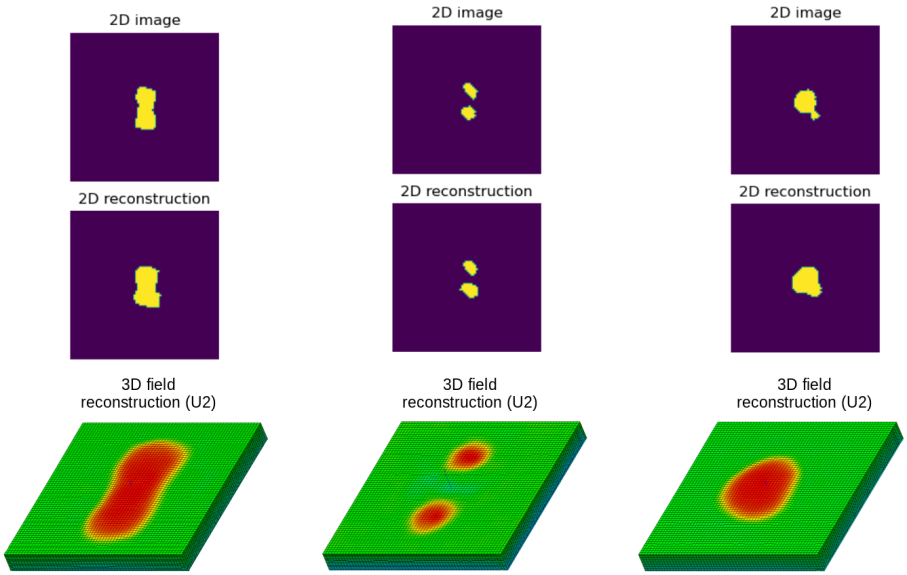


Figure 12: Initial crack images from training set, their reconstruction and the cross modal displacement field reconstruction (cracks 122, 131 and 16 from the training set).

the encoding mesh generated by the cross modal reconstruction is transferred on the external surface of the TLV. The error η of the online phase is based on the cumulated plasticity defined by Eq. (10). The following error indicator $\widetilde{\eta}$ is the average difference between the original image and its reconstruction:

$$\widetilde{\eta} = \frac{\sum_{i=1}^{N} |P(i) - \widetilde{P}(i)|}{\sum_{i=1}^{N} P(i)}, \qquad P(i), \widetilde{P}(i) \in \{0; 1\}, \tag{14}$$

where $P(i)$ and $\widetilde{P}(i)$ represent the value of the pixel i for the original and reconstructed image respectively. The number of pixel of the image is $N = 84 \times 84$. In order to see whether a linear relation between those two errors exists, for each simulation $\widetilde{\eta}$ and η are plotted on the same graph. Since there are few extreme values a probability density function is used to have a clear insight of the tendency. A linear relation is defined with the least mean square and a 40% error cone is drawn on Fig. (13).

A tendency is observed but the precision of the indicator is not high, it would give a good hint about the online computation. With a fixed criterion (here 5%) on the actual error η one can evaluate the True Positive (indicator low and true error low), True Negative (indicator high and true error high), False Positive (indicator low and true error high) and False Negative (indicator high and true error low) repartition. This repartition is shown in Fig. (14).

If the estimated error is too high then a full finite element simulation should be made. Therefore in order to be the more conservative the false positive should be minimized. Moreover the linear relation can be adapted in order to be more conservative if needed. In order to see more precisely the error that can be encountered for defects, a 2D histogram is made for defect 1042. This defect belongs to the test dataset hence the model has not been trained on this defect. It shows the density of Gauss points according to their value of cumulated plasticity and von Mises stress for the full order model (FOM) and for the developed model. Since at the end of the simulation most of the points are fully plastic and the von Mises value is equal to the limit stress $R_0 = 400MPa$, the graphs are drawn at t=0.8. Remind that the error is evaluated on the TLV only. This is represented in Fig (15).

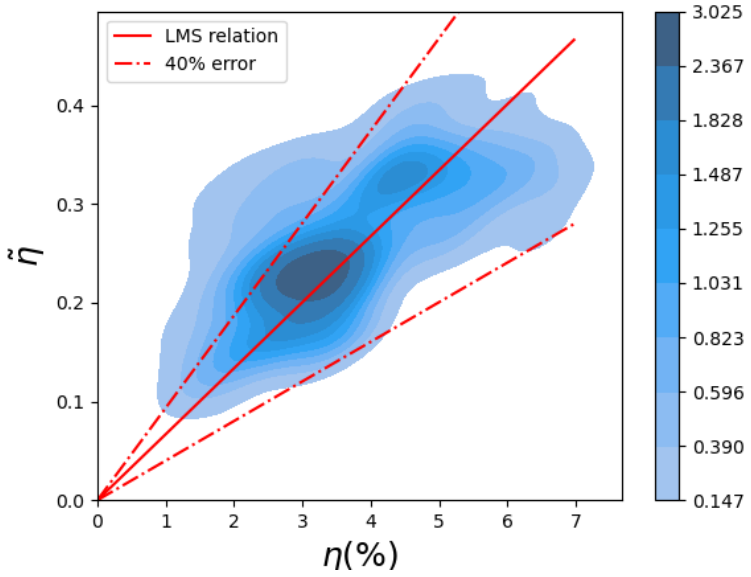


Figure 13: Error indicator evaluation map.

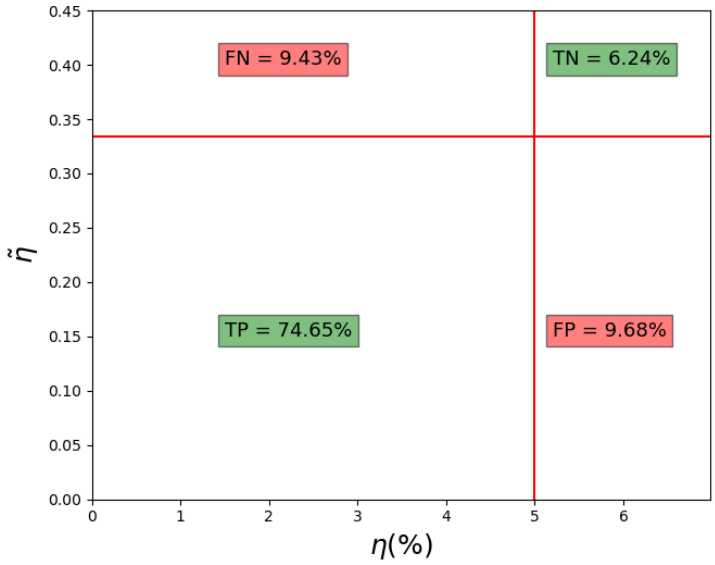


Figure 14: True Positive, True Negative, False Positive and False Negative repartition with a criterion of 5% of error.

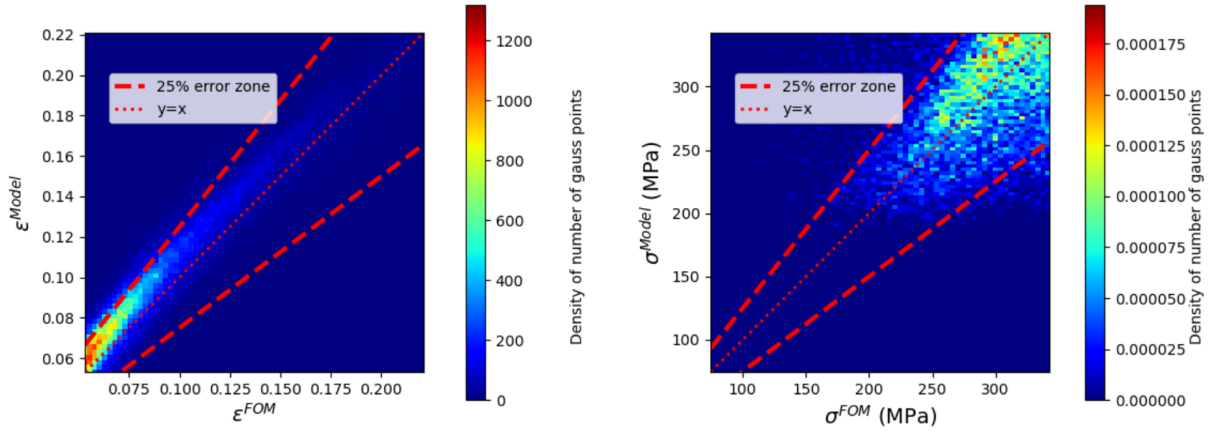


Figure 15: Error Graph on ε^{pcum} and σ^{Mises} t=0.8 for defect 1042 from test dataset.

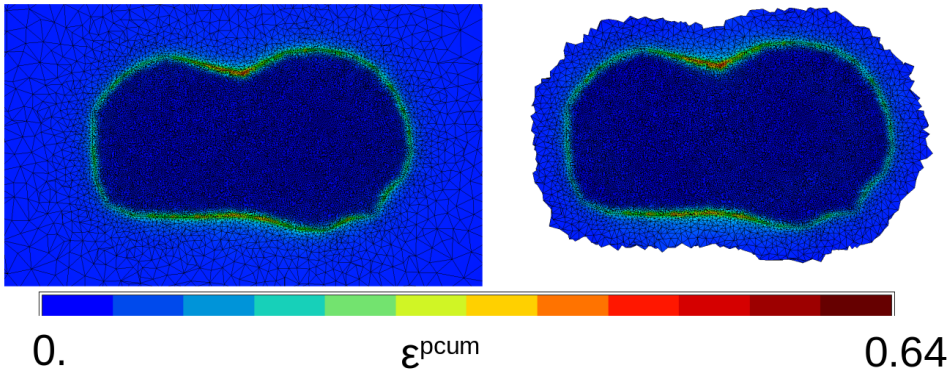


Figure 16: ε^{pcum} with FOM (left) and with the model (right) on the TLV for defect 1042 from test dataset.

Almost all the points are included in the 25% error zone, showing the good performance of the method for the defect 1042. The result maps of the cumulated plasticity for defect 1042 is shown Fig. (16).

It is clearly shown that the global aspect of the field between the two methods is highly similar. In order to have a more global view of the capacities of the so developed model an histogram of the error based on the cumulated plasticity (see Eq. 10) is made. Since the TLV is included in the full mesh, the error is computed strictly on the TLV. The classical split has been applied to train the model so the three sets "train", "validation" and "test" are represented on this histogram Fig. (17a).

The cumulated probability graph in Fig. (17b) shows that 95% of the defects have an error below 9.8% for the training set, 12.4% for the validation set and 16.3% for the testing set. Good results are achieved with the developed methodology. For every set of data the average error is lower than 7% underlining the accuracy of the model. In order to put into relief the need to developed such a model a naive model has been created. Instead of generating a specific 3D displacement field for each 2D image, a mean displacement field is computed over the training set. All the training data have been taken and a mean field have been generated (see Fig. 19a). The latter is then used to apply the boundary condition on the TLV that are free from defect dependency. As it has been done before, an histogram of the error made on the cumulated plasticity is made for the test set and compared with the developed model. Those results are shown on Fig. (18).

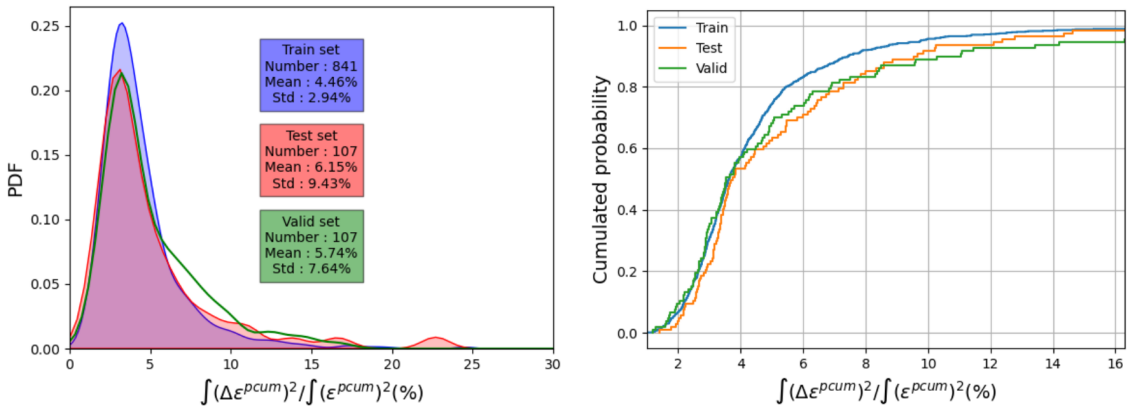


Figure 17: (a) Probability density function and (b) cumulated probability of the error for the training, validation and test sets

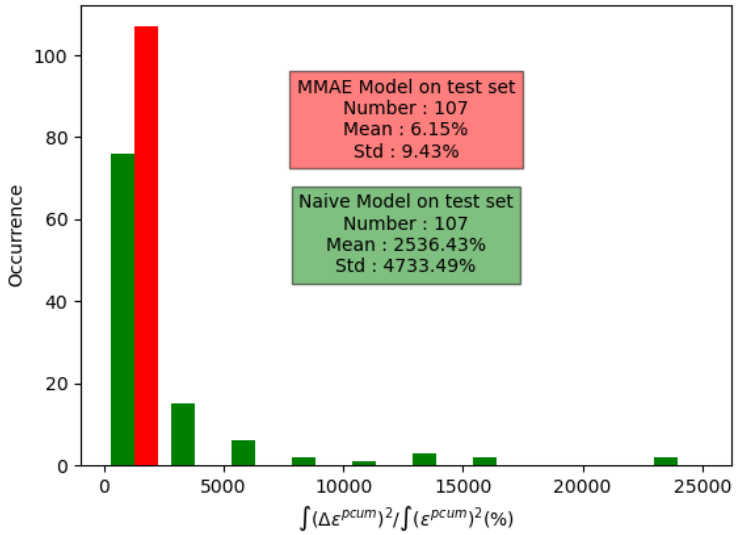


Figure 18: Histogram of the error for the test sets with the MMAE model and the naive model

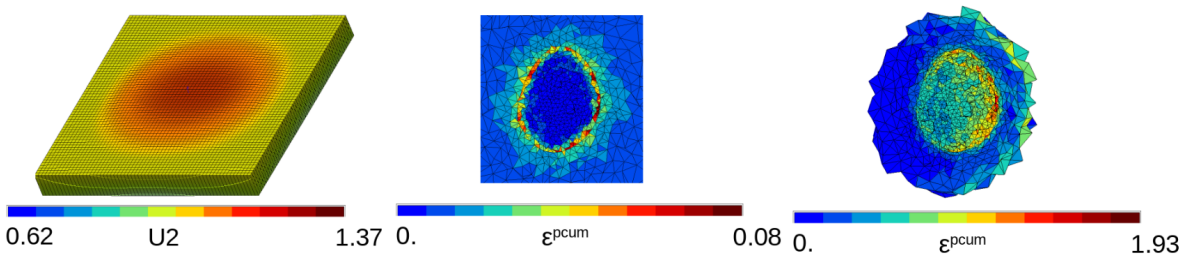


Figure 19: Encoding mesh with mean U2 field (a), ε^{pcum} field with FOM (b) and ε^{pcum} field with naive method (c) for defect 989

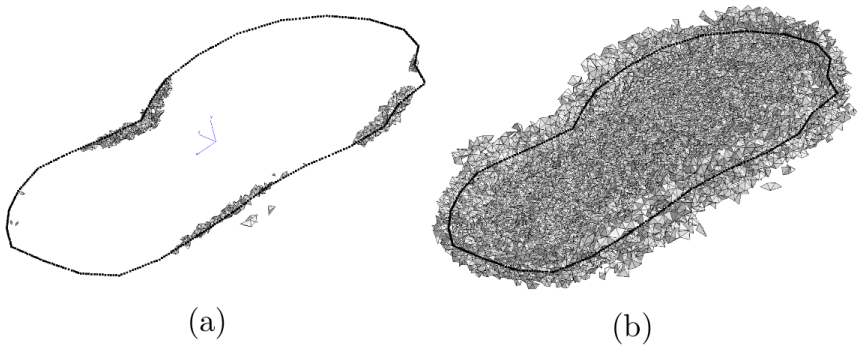


Figure 20: Mesh of defect 1042 where $\Delta\varepsilon^{pcum} > 5\%$ when (a) the BC are applied on the external surface of the TLV and when (b) the BC are applied on the whole TLV.

Very high error are obtained with this method except for defects having a shape close to the average shape in the training set. Indeed for small defects large displacements are imposed on the external surface generating important strains and stresses. The difference between the FOM and the naive model for defect 989 is shown on Figs. (19b,19c).

The fluctuation of the displacement field caught by the MMAE is therefore necessary to achieve good performance. To the contrary one can question why the use of the MMAE solution solely on the external surface is made and not on the whole domain. This would made great different for the computation time since only the integration of the constitutive law would be required. Therefore comparison is made on defect 1042 from the test set with boundary conditions on the whole domain and solely on the external surface is made. Solely the meshed domain where $\Delta\varepsilon^{pcum} > 5\%$ is conserved in both case in order to see the impact of the position of the boundary conditions. This is shown in Fig.(20a), (20b).

The number of element where $\Delta\varepsilon^{pcum} > 5\%$ is clearly inferior when the boundary conditions are applied on the external surface. The fact that the boundary conditions are solely on the external surface enables to make a finite element correction of the solution find by the MMAE.

The use of MMAE allows one to represent each defect as a point encoded in the latent space. Fig. (21) represents all the defect in the latent space, since the latent space is of dimension 600 a multi dimensional scaling (MDS) is made to have a 2D representation of this space [34, 35].

MDS is an information visualization method which consists in finding a low-dimensional dataset $\vec{Z}_0$whose matrix of Euclidean distances $d(\vec{Z}_0)$ is an approximation of the input dissimilarity matrix δ. To that end, a cost function called stress function is minimized with respect

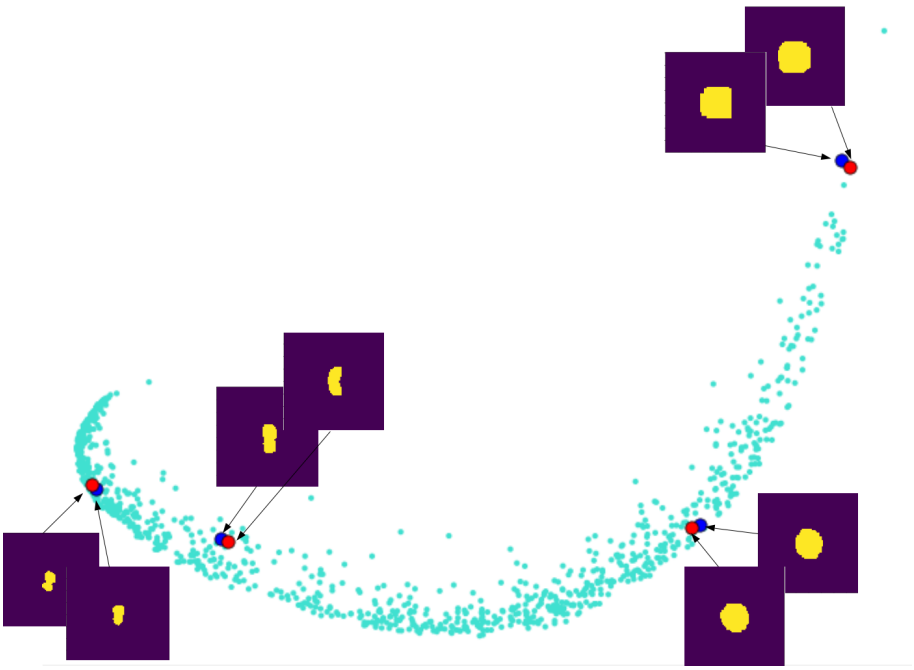


Figure 21: Low dimensional representation of the common latent space

to $\vec{Z}$:

$$\vec{Z}_0 = \underset{\vec{Z}}{\text{argmin}} \left(\sum_{i<j} (\delta_{ij} - d_{ij}(\vec{Z}))^2 \right) \tag{15}$$

From left to right the blue dots represent the defects 539, 938, 537 and 341 while the red dots represent their nearest neighbour respectively being the defects 4, 612, 105 and 368.

We remark one may use the reduced basis of the nearest neighbour in the common latent space where a full order model is computed to solve the problem. This would be an alternative since the latent space is common to the image and the displacement field. As shown in [36], no bijection necessary exists between the mechanical and the morphological latent space. This warrants the use of a common latent space generated with the MMAE. The technique consisting in using a reduced basis recommended by a neurone network is called "ROM-net" [37]. This approach works well for defects like gaz pores and has also been used in [36]. Nevertheless this method offers non-optimal results when dealing with cracks. The singularity at the crack tip is poorly represented on a reduced basis selected in a dataset of reduced basis. As illustrated in Fig. (11), the error is located at the crack tip, which is the zone of interest. Use of a submodelling approach and an adequate subdomain enables to circumvent this problem.

5. High level engineering analysis to assess the safety of structures containing cracks

The use of fracture assessement diagram (FAD) is common in the industry to certify whether a structure containing cracks are still operational or not. The determination of the stress intensity factor (SIF) and the limit load is necessary to use them [38]. Two domains (safe and unsafe) can then be drawn on this diagram. Such mechanical quantities of interest are usual outputs of finite element software in engineering. An explanatory FAD is given in Fig. (22) which is taken from [39].

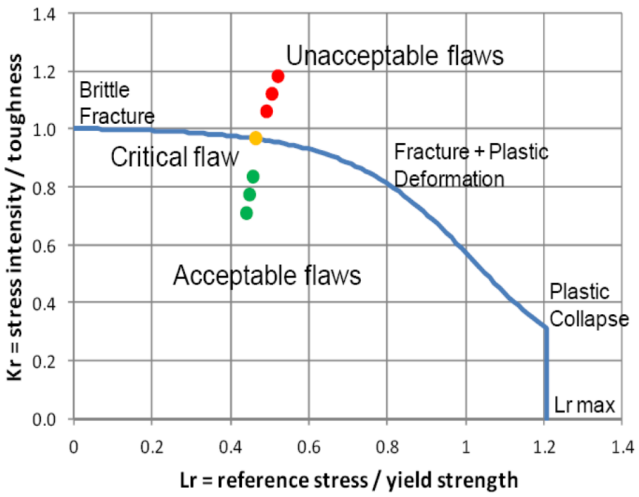


Figure 22: FAD example with the different regions

With surrogate models, only specific quantities of interest can be computed. With the presented work all quantities around the crack such as the stress intensity factor and the J integral can be computed with classical finite element methods. On the contrary the limit load is evaluated with the reaction forces at the extremities of the specimen. Not enough information are present on the TLV to compute the limit load. The gappy proper orthogonal decomposition enable to recover the full field when a reduced base is available. In this context the use of MMAE happens to be a very relevant choice and constitutes a major point of this paper. Indeed reduced basis can be computed for the training phase and then, during the online phase, the reduced base of the k-nearest neighbours (k-NN) in the latent space can be used to recover the full field and hence compute the limit load. This methodology consisting in using a reduced base from another defect has already been used in the literature [37, 2].

Considering a reduced basis $\vec{V}$ and following on the Gappy POD [40], any vector $\vec{u}$ which belongs to the column space $colspan(\vec{V})$ of $\vec{V}$ may be recovered by using few entries $\vec{u}[\mathcal{F}]$ of $\vec{u}$, if $\vec{V}[\mathcal{F},:]$ is a full column rank matrix. Here the set of degree of freedom $\mathcal{F}$ corresponds to the ones of the TLV. Such recovery procedure takes the form:

$$\vec{u} \in colspan(\vec{V}), \qquad \vec{u} = \vec{V} \cdot \left(\vec{V}[\mathcal{F},:]^T \cdot \vec{V}[\mathcal{F},:]\right)^{-1} \cdot \vec{V}[\mathcal{F},:]^T \cdot \vec{u}[\mathcal{F}] \tag{16}$$

Fig. (23) shows the global workflow that allows one to recover the full field by taking advantage of the information contained in the latent space. Note that if more than one reduced basis is selected (i.e. $k > 1$) to recover the full field, techniques have been developed in [38, 41] to create a global basis based on many others.

The same idea can be used for the stress field $\boldsymbol{\sigma}$ using the reduced basis of the stresses $\vec{V}^{\sigma}$. Once $\boldsymbol{\sigma}$ is recovered it is possible to integrate this field on the top surface to obtain the force at every time step and so the limit load. It is then possible to evaluate the limit load for every defect by taking the reduced basis of its the nearest neighbour in the latent space. To evaluate the J integral the information present in the TLV are sufficient. Two methods to evaluated the J integral are used. One for which the J integral is computed as the mean on the curvilinear abscissa and one more conservative method where J is equal to the maximum

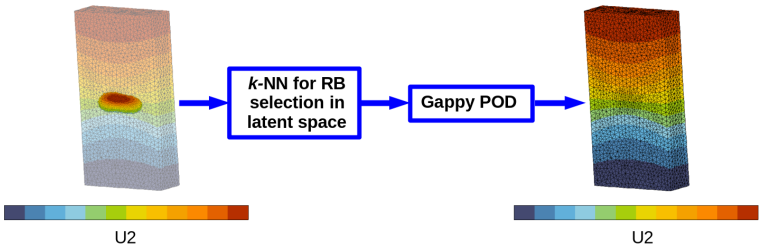


Figure 23: Full field recovery with use of the information contained in the latent space and gappy POD process

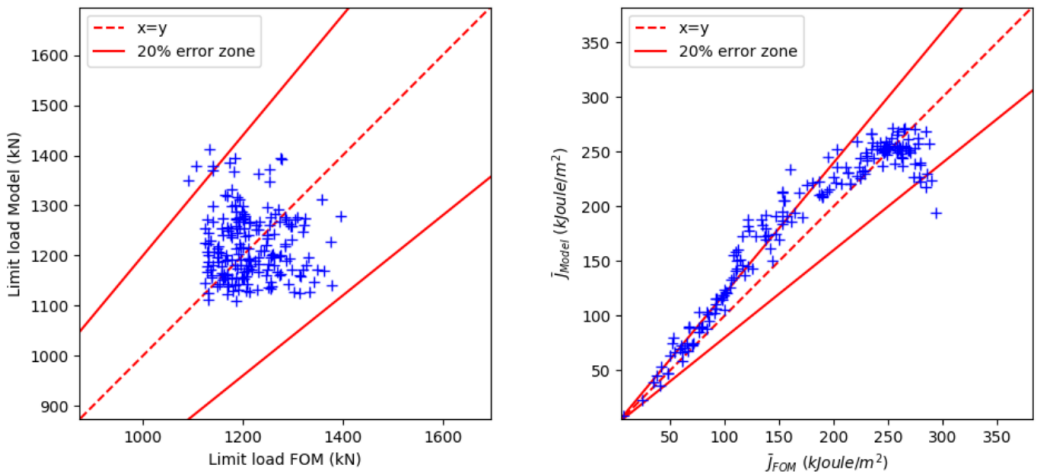


Figure 24: Limit load value with model and FOM (left) and $\bar{J}$ with model and FOM (right)

value along the crack front. Both of these quantities are detailed in Eq. (17).

$$J_{max} = \max_{s} J(s), \quad \bar{J} = \frac{\oint_{front} J(s) ds}{\oint_{front} ds}. \tag{17}$$

Hence graphs to evalutate the error between the full order model (FOM) and the developped model are computed over the validation and testing set. Fig. (24) shows that good results are obtained, most of the points are contained in the 20% error zone. If a closer look at defect 1042 (belonging to the test set) is taken, its nearest neighbour in the latent space is the defect 53 (belonging to the training set). Fig. (25) shows the force displacement curve for defect 1042 with the described method and with a full order model (FOM).

The traction curve is also obtained with the developed methodology is very similar to the curve of the FOM. Similarly it is possible to compute the J integral along the curvilinear abscissa for defect 1042 at specific time step on Fig. (26). The mean error for each figure reads:

$$\mu = \frac{1}{N_{nodes}} \sum_{i \leq N_{\text{nodes}}} \frac{|J_i^{FOM} - J_i^{Model}|}{|J_i^{FOM}|}, \tag{18}$$

where N_{nodes} refers to the number of nodes at the crack front and J_i^{FOM} and J_i^{Model} refer to the value of the J integral at the node i.

Good results are achieved for the computation of the J integral as well as for the limit load, underlining the performance of the so developed model. When the limit load is reached

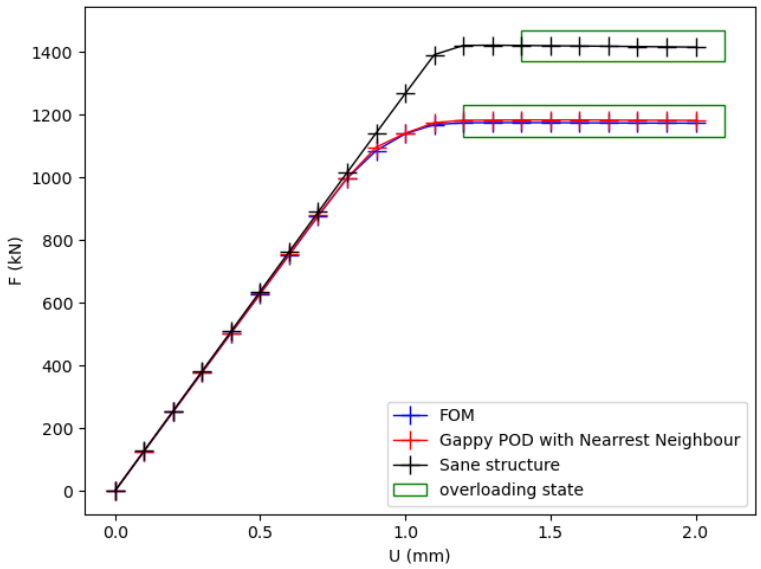


Figure 25: Traction curve for defect 1042

(green zone in 25) at t = 0.6 the error on the J-integral curves starts to increase (see Fig. 26). This is due to volumetric locking which is a common problem when cracked structures are considered. This type of problem is not related to the developed method. It is intrinsic to the finite element formulation and is an active field of research [16, 17]. Nevertheless to draw accurate FAD diagram the solution up to the limit load is sufficient. Both quantities J integral and limit load can be computed hence the loading path for a specific defect on FAD diagrams can be drawn. FAD diagrams are usually drawn with the SIF rather than with the J integral, the relation between these quantities is given in eq. 19.

$$K = \sqrt{J.E'} \qquad \text{with: } E' = \frac{E}{1-\nu^2} \tag{19}$$

On FAD the loading and the SIF are respectively normalized by the limit load and the material SIF ($Kmat$). The value of the material SIF is similar to the value of an aluminium material: $K_{mat} = 35$ kJoule/m^2. Fig. (27) shows the loading path for defect 1042 with the model and with the reference FOM. The loading path is restricted to the domain up to the overloading state as the structure necessarily belongs to the unacceptable flaws domain afterwards (see Fig. 22). Recall that the reduced base of defect 53 has been used to compute the force since it is the nearest neighbour of defect 1042 in the latent space. For each of the plotted path, the value of the stress intensity factor K is equal to the maximum value around the crack front. Both defects 1042 and 53 are represented on Fig. (27).

The loading path is similar to the reference one. Barely no error is made on the limit load compared to the evaluation of SIF. This problem is partially due to volumetric locking and bad pressure field computation around the front crack [17, 42]. speed-ups which are obtained for all the defects are around 3 in average and up to 10 for the best. The value of the speed-ups for each defect are represented on the histogram Fig. (28a).

The speed-ups can be improved by reducing the TLV but this will increase the error hence a compromise between speed and accuracy has to be made as it is shown on Fig. (28b).

The strength of this method is that once the MMAE has learned the link between the displacement field and the 2D images, any type of element can be used for the TLV. Moreover

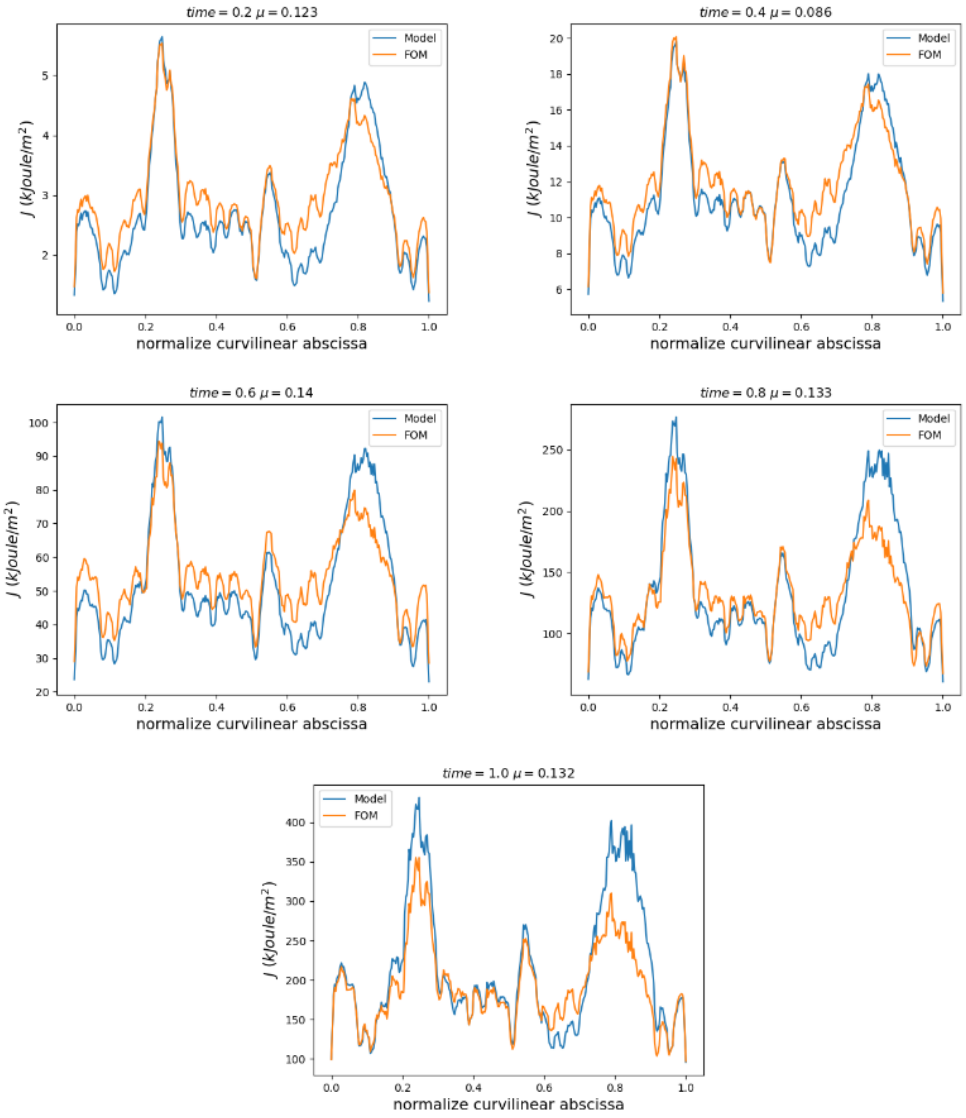


Figure 26: J-integral along the curvilinear abscissa for defect 1042 at each time step

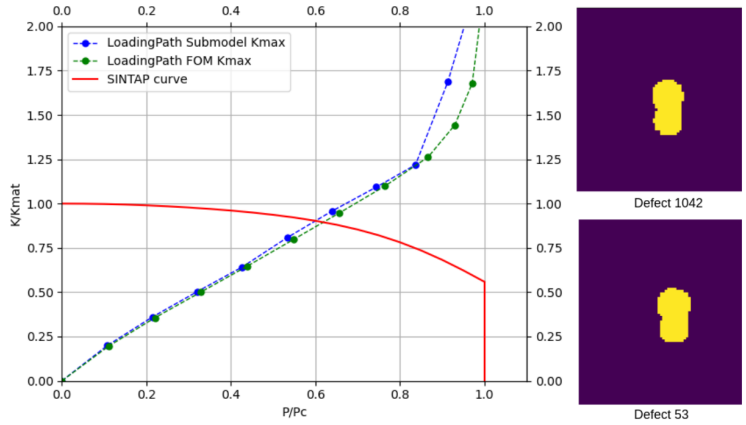
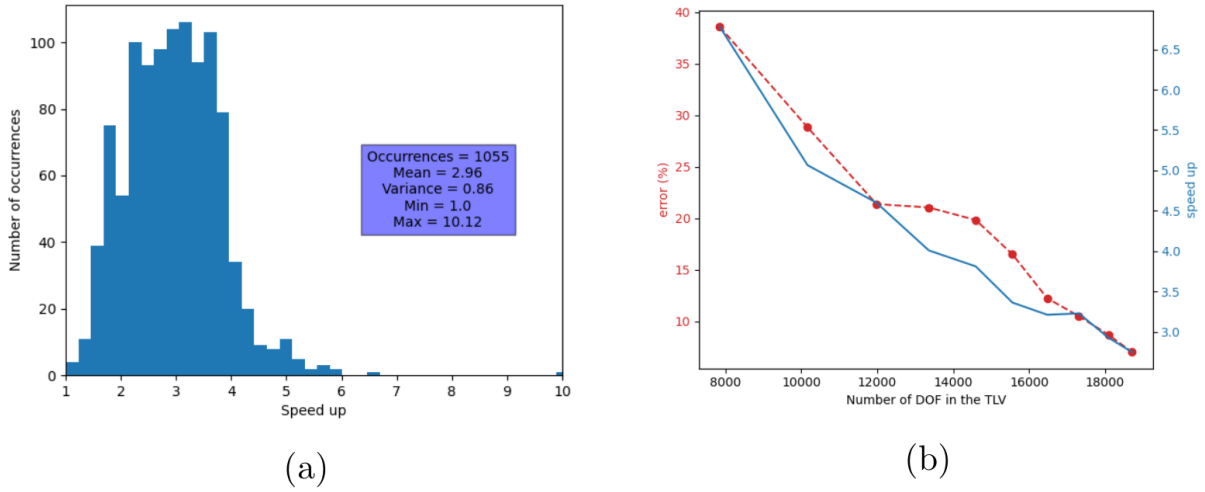


Figure 27: FAD with Model and FOM for defect 1042 (left) and the image of defect 1042 and 53, its nearest neighbour (right).



(a) (b)

Figure 28: speed-up histogram (a) and variation of the speed-up with the size of the problem for defect 1042 (b) speed-up and error evolution curve in function of the number of DOF.

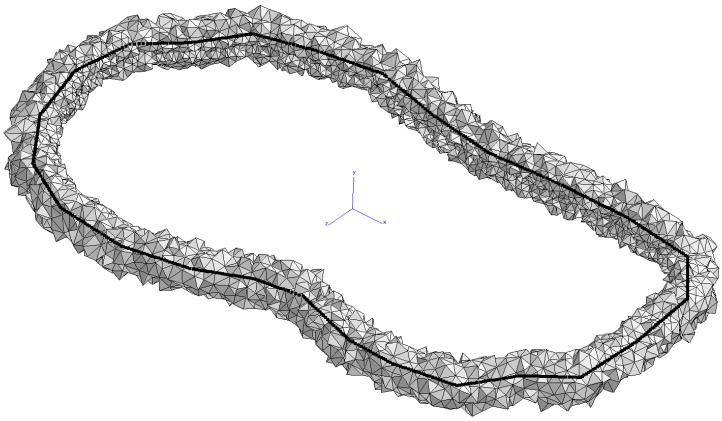
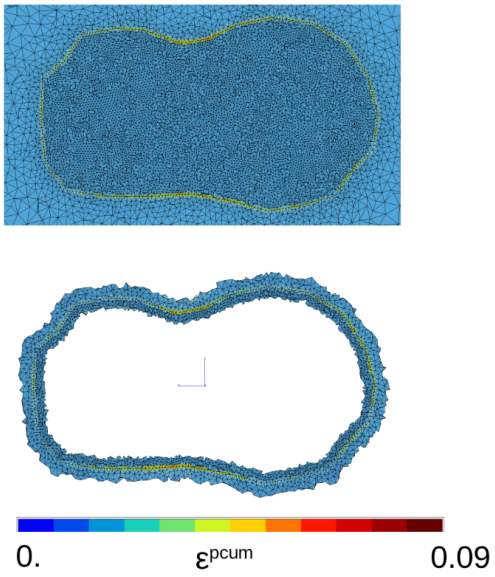


Figure 29: Smaller TLV for defect 1054

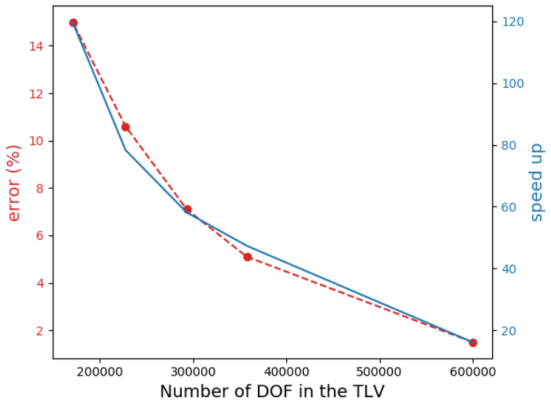
the mesh of the later can also be refined if needed. These modifications can enhance the speed-up. A smaller TLV for defect 1054 has been defined which consists in a torus around the front crack (see Fig. 29).

Other element type have been used with pressure control. They are more adapted for cracked structures simulations than the linear tetrahedron with reduced integration. This type of elements has additional degrees of freedom and need to be used with a quadratic formulation of the element [43, 44]. Therefore the computation times are much larger than the ones previously done. This is the reason why the training phase has not been done with these elements. The same MMAE model that has been trained in the previous section is used. The cumulated plasticity field obtained for defect 1042 with the MMAE approach and with the FOM are shown in Fig. (30a) while Fig. (30b) shows the evolution of the error η and the speed-up in function of the number of degrees of freedom also for defect 1042.

Much larger speed-ups are obtained with this configuration: up to 120 times faster (see Fig. 30). This type of approach enable to take entirely profit from the submodelling approach of the developed methodology. Nevertheless with such TLV the computation of the cumulated plasticity works well but for the J integral it leads to high level of error when not enough



(a)



(b)

Figure 30: (a) ε^{pcum} with FOM (up) and with the model (down) on the reduced TLV for defect 1042 and (b) speed-up and error evolution curve in function of the number of DOF.

elements (radius of the thorus inferior to $4.5mm$) around the front crack are present.

6. Conclusion

The mechanical effect of cracks in 3D volumes is known to be a function of crack morphology. We show how a MMAE can be trained to learn this implicit function from experimental data and simulation data. An automatic mechanistic simulation chain is required to get the simulation data prior training the MMAE. A high level engineering analysis of the prediction is enabled by using a mechanical submodel, termed tight local volume, feed by the MMAE predictions. The encoding mesh, that supports simulation data, is a continuous mesh. It does not allow to represent the discontinuity induced by cracks. Nevertheless, this encoding mesh is sufficient to develop an artificial intelligence for boundary conditions on a tight local volume. Good results have been achieved with the developed method. Low errors on the whole TLV are obtained (around 5%). The need of such a method has been demonstrated by comparing it with a naive model. All the standard outputs of a classical finite element computation are available and enables one to draw FAD if needed. speed-ups between 3 and 120 are obtained. Better speed-ups have been obtained by reducing the TLV size to the detriment of the accuracy. Indeed when large speed-ups are obtained the J integral is very inaccurate due to the lack of element around the crack front. Moreover with this method, speed-ups depends on the mesh hence with more complex geometries it would be possible to increase the speed-up. Nevertheless the model suffers from a major limitation at the moment. The position of the crack should, in full generality, be in the encoding mesh. Not all configurations can accordingly be handled. Adding two new parameters for the position and orientation of the crack could be an option. The encoding mesh would then be implanted according to these parameters. The main asset of this method is that all the possible outputs of a standard finite element computation are also available with the so developed model hence it has been possible to launch simulation with new elements though the MMAE model had been trained with different elements. On the contrary a surrogate model would have larger speed-up but

```
################# Encoder #################
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 80, 80, 1)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 80, 80, 16)        272
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 40, 40, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 40, 40, 32)        8224
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 20, 20, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 20, 20, 64)        32832
_________________________________________________________________
flatten (Flatten)            (None, 25600)             0
_________________________________________________________________
dense (Dense)                (None, 280)               7168280
_________________________________________________________________
dense_2 (Dense)              (None, 70)                19670
=================================================================
Total params: 7,229,278
Trainable params: 7,229,278
Non-trainable params: 0
_________________________________________________________________
################# Decoder #################
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 70)]              0
_________________________________________________________________
dense_3 (Dense)              (None, 70)                4970
_________________________________________________________________
dense_4 (Dense)              (None, 280)               19880
_________________________________________________________________
dense_5 (Dense)              (None, 25600)             7193600
_________________________________________________________________
reshape (Reshape)            (None, 20, 20, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 20, 20, 64)        65600
_________________________________________________________________
up_sampling2d (UpSampling2D) (None, 40, 40, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 40, 40, 32)        32800
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 80, 80, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 80, 80, 16)        8208
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 80, 80, 1)         257
=================================================================
Total params: 7,325,315
Trainable params: 7,325,315
Non-trainable params: 0
```

(a)

```
################# Encoder #################
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 64, 64, 64, 5)]   0
_________________________________________________________________
conv3d (Conv3D)              (None, 64, 64, 64, 8)     2568
_________________________________________________________________
conv3d_1 (Conv3D)            (None, 64, 64, 64, 16)    8208
_________________________________________________________________
max_pooling3d (MaxPooling3D) (None, 32, 32, 32, 16)    0
_________________________________________________________________
conv3d_2 (Conv3D)            (None, 32, 32, 32, 16)    16400
_________________________________________________________________
max_pooling3d_1 (MaxPooling3 (None, 16, 16, 16, 16)    0
_________________________________________________________________
flatten (Flatten)            (None, 65536)             0
_________________________________________________________________
dense (Dense)                (None, 200)               13107400
=================================================================
Total params: 13,134,576
Trainable params: 13,134,576
Non-trainable params: 0
_________________________________________________________________
################# Decoder #################
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 200)]             0
_________________________________________________________________
dense_1 (Dense)              (None, 200)               40200
_________________________________________________________________
dense_2 (Dense)              (None, 65536)             13172736
_________________________________________________________________
reshape (Reshape)            (None, 16, 16, 16, 16)    0
_________________________________________________________________
conv3d_transpose (Conv3DTran (None, 16, 16, 16, 16)    16400
_________________________________________________________________
up_sampling3d (UpSampling3D) (None, 32, 32, 32, 16)    0
_________________________________________________________________
conv3d_transpose_1 (Conv3DTr (None, 32, 32, 32, 16)    16400
_________________________________________________________________
up_sampling3d_1 (UpSampling3 (None, 64, 64, 64, 16)    0
_________________________________________________________________
conv3d_transpose_2 (Conv3DTr (None, 64, 64, 64, 8)     8200
_________________________________________________________________
conv3d_transpose_3 (Conv3DTr (None, 64, 64, 64, 5)     2565
=================================================================
Total params: 13,256,501
Trainable params: 13,256,501
Non-trainable params: 0
```

(b)

```
############# Fully connected left neurone network #############
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 70)]              0
_________________________________________________________________
dense_2 (Dense)              (None, 70)                4970
_________________________________________________________________
dense_3 (Dense)              (None, 600)               42600
=================================================================
Total params: 47,570
Trainable params: 47,570
Non-trainable params: 0
_________________________________________________________________


############# Fully connected right neurone network #############
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 600)]             0
_________________________________________________________________
dense (Dense)                (None, 600)               360600
_________________________________________________________________
dense_1 (Dense)              (None, 70)                42070
=================================================================
Total params: 402670
Trainable params: 402670
Non-trainable params: 0
```

(c)

Figure 31: Summary of the 2D autoencoder (a), the 3D autoencoder for any field U1, U2 or U3 (b) and the fully-connected left and right neurone network (c).

would be trained to predict only specific outputs and would need to be retrained each time other output would be needed.

Appendix A. Detail of the 2D and 3D autoencoders

Details of the 2D and 3D autoencoders, as provided by the TensorFlow summary, are given in Fig. (31).

References

[1] L. Lacourt, D. Ryckelynck, S. Forest, V. Rancourt, and S. Flouriot, "Hyper-reduced direct numerical simulation of voids in welded joints via image-based modeling," *International Journal for Numerical Methods in Engineering*, vol. 121, no. 11, pp. 2581–2599, 2020.

[2] D. Ryckelynck, T. Goessel, and F. Nguyen, "Mechanical dissimilarity of defects in welded joints via grassmann manifold and machine learning," *Comptes Rendus. Mécanique*, vol. 348, no. 10-11, pp. 911–935, 2020.

[3] G. Bhatt, P. Jha, and B. Raman, "Representation learning using step-based deep multi-modal autoencoders," *Pattern Recognition*, vol. 95, pp. 12–23, 2019.

[4] K. Lee and K. T. Carlberg, "Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders," *Journal of Computational Physics*, vol. 404, p. 108973, 2020.

[5] P. Le Delliou, A. Dahl, C. Sonnefraud, and W. Vincent, "Experimental results and numerical analyses of tests conducted on large alloy 600 centre cracked tensile specimens," in *Pressure Vessels and Piping Conference*, vol. 51630, p. V03BT03A024, American Society of Mechanical Engineers, 2018.

[6] U. Zerbst, S. Manfred, S. Webster, and R. A. Ainsworth, "Chapter 5 – the model parameters," in *Fitness-for-service Fracture Assessment of Structures Containing Cracks*, pp. 89 – 135, Oxford: Academic Press, 2007.

[7] J. Berg and K. Nyström, "A unified deep artificial neural network approach to partial differential equations in complex geometries," *Neurocomputing*, vol. 317, pp. 28–41, 2018.

[8] M. Layouni, M. S. Hamdi, and S. Tahar, "Detection and sizing of metal-loss defects in oil and gas pipelines using pattern-adapted wavelets and machine learning," *Applied Soft Computing*, vol. 52, pp. 247–261, 2017.

[9] M. San Biagio, C. Beltran-Gonzalez, S. Giunta, A. Del Bue, and V. Murino, "Automatic inspection of aeronautic components," *Machine Vision and Applications*, vol. 28, pp. 1–15, 2017.

[10] C. Escobar and R. Morales Menendez, "Machine learning techniques for quality control in high conformance manufacturing environment," *Advances in Mechanical Engineering*, vol. 10, no. 2, pp. 1–16, 2018.

[11] C. Sobie, C. Freitas, and M. Nicolai, "Simulation driven machine learning: Bearing fault classification," *Mechanical Systems and Signal Processing*, vol. 99, pp. 403–419, 2018.

[12] Z. Ji, Y. Zhao, Y. Pang, and X. Li, "Cross-modal guidance based auto-encoder for multi-video summarization," *Pattern Recognition Letters*, vol. 135, pp. 131–137, 2020.

[13] Y. Zhang, Y. Qiu, Y. Cui, S. Liu, and W. Zhang, "Predicting drug-drug interactions using multi-modal deep auto-encoders based network embedding and positive-unlabeled learning," *Methods*, vol. 179, pp. 37–46, 2020. Interpretable machine learning in bioinformatics.

[14] L. Lacourt, *Étude numérique de la nocivité des défauts dans les soudures*. PhD thesis, Mines ParisTech - Université PSL, 2019.

[15] M. Bartoň, I. Hanniel, G. Elber, and M.-S. Kim, "Precise hausdorff distance computation between polygonal meshes," *Computer Aided Geometric Design*, vol. 27, no. 8, pp. 580–591, 2010. Advances in Applied Geometry.

[16] J. Besson, D. Moinereau, and D. Steglich, *Local approach to fracture*. Presses des MINES, 2006.

[17] V. Davaze, N. Vallino, B. Langrand, J. Besson, and S. Feld-Payet, "A non-local damage approach compatible with dynamic explicit simulations and parallel computing," *International Journal of Solids and Structures*, vol. 228, p. 110999, 2021.

[18] J. Besson and R. Foerch, "Large scale object–oriented finite element code design," *Computer Methods in Applied Mechanics and Engineering*, vol. 142, pp. 165–187, 1997.

[19] R. Foerch, J. Besson, G. Cailletaud, and P. Pilvin, "Polymorphic constitutive equations in finite element codes," vol. 141, pp. 355–372, 1997.

[20] J. Besson and R. Foerch, "Application of object–oriented programming techniques to the finite element method. Part I— General concepts," vol. 7, no. 5, pp. 535–566, 1998.

[21] J. Besson, R. Le Riche, R. Foerch, and G. Cailletaud, "Application of object–oriented programming techniques to the finite element method. Part II— Application to material behaviors," vol. 7, no. 5, pp. 567–588, 1998.

[22] P. G. Ciarlet, *The finite element method for elliptic problems*. North Holland, 1978.

[23] S. Fayolle, "Modèles de grandes déformations GDEF-LOG et GDEF-HYPO-ELAS." Online at https://www.code-aster.org/V2/doc/v12/fr/man_r/r5/r5.03.24.pdf. Retrieved Jan. 4, 2022, 2015.

[24] C. Miehe, N. Apel, and M. Lambrecht, "Anisotropic additive plasticity in the logarithmic strain space: modular kinematic formulation and implementation based on incremental minimization principles for standard materials," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 47, pp. 5383–5425, 2002.

[25] J. Chai, H. Zeng, A. Li, and E. W. Ngai, "Deep learning in computer vision: A critical review of emerging techniques and application scenarios," *Machine Learning with Applications*, vol. 6, p. 100134, 2021.

[26] D. Huang, J. N. Fuhg, C. Weißenfels, and P. Wriggers, "A machine learning based plasticity model using proper orthogonal decomposition," *Computer Methods in Applied Mechanics and Engineering*, vol. 365, p. 113008, 2020.

[27] P. Cheridito, A. Jentzen, and F. Rossmannek, "Non-convergence of stochastic gradient descent in the training of deep neural networks," *Journal of Complexity*, vol. 64, p. 101540, 2021.

[28] Y. Guo, J. Chen, Q. Du, A. Van Den Hengel, Q. Shi, and M. Tan, "Multi-way backpropagation for training compact deep neural networks," *Neural Networks*, vol. 126, pp. 250–261, 2020.

[29] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," p. 396–404, 1989.

[30] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[31] E. Gomede, R. M. de Barros, and L. de Souza Mendes, "Deep auto encoders to adaptive e-learning recommender system," *Computers and Education: Artificial Intelligence*, vol. 2, p. 100009, 2021.

[32] Y. Wu, S. Wang, and Q. Huang, "Multi-modal semantic autoencoder for cross-modal retrieval," *Neurocomputing*, vol. 331, pp. 165–175, 2019.

[33] K. Raimi, "A compiled visualisation of the common convolutional neural networks." Online at https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d. Retrieved Jan. 4, 2022., 2019.

[34] J. T. Machado and Y. Luchko, "Multidimensional scaling and visualization of patterns in distribution of nontrivial zeros of the zeta-function," *Communications in Nonlinear Science and Numerical Simulation*, p. 105924, 2021.

[35] I. Borg and P. J. F. Groenen, *Modern Multidimensional Scaling Theory and Applications*. New York: Springer, 2005.

[36] H. Launay, F. Willot, D. Ryckelynck, and J. Besson, "Mechanical assessment of defects in welded joints: morphological classification and data augmentation," *Journal of Mathematics in Industry: Shape, Form and Patterns in Medicine, Biotechnology and Materials Science*, vol. 11, no. 18, 2021.

[37] T. Daniel, F. Casenave, N. Akkari, and D. Ryckelynck, "Model order reduction assisted by deep neural networks (rom-net)," *Advanced Modeling and Simulation in Engineering Sciences*, vol. 7, no. 1, pp. 1–27, 2020.

[38] H. Launay, J. Besson, D. Ryckelynck, and F. Willot, "Hyper-reduced arc-length algorithm for stability analysis in elastoplasticity," *International Journal of Solids and Structures*, vol. 208–209, pp. 167–180, 2021.

[39] R. K. Sharma, A. Ghosh, D. Bhachawat, S. Ingole, A. Balasubramanian, and U. Muktibodh, "Assessment of structural integrity of pressure tubes during cold pressurization," *Procedia Engineering*, vol. 86, pp. 359–366, 2014.

[40] R. Everson and L. Sirovich, "Karhunen-Loève procedure for gappy data," *Journal of the Optical Society of America A*, vol. 12, pp. 1657–1664, 1995.

[41] D. Ryckelynck, K. Lampoh, and S. Quilicy, "Hyper-reduced predictions for lifetime assessment of elasto-plastic structures," *Meccanica*, vol. 51, no. 2, pp. 309–317, 2016.

[42] G. Hütter, T. Linse, U. Mühlich, and M. Kuna, "Simulation of ductile crack initiation and propagation by means of a non-local Gurson-model," *International Journal of Solids and Structures*, vol. 50, no. 5, pp. 662–671, 2013.

[43] R. Taylor, "A mixed-enhanced formulation for tetrahedral finite elements," *International journal of Numerical Engineering*, vol. 47, no. 1–3, pp. 205–227, 2000.

[44] D. Al Akhrass, J. Bruchon, S. Drapier, and S. Fayolle, "Integrating a logarithmic-strain based hyperelastic formulation into a three-field mixed finite element formulation to deal with incompressibility in finite-strain elastoplasticity," *Finite Elements in Analysis and Design*, vol. 86, pp. 61–70, 2014.