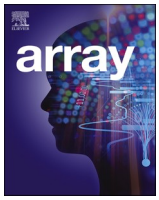
Array 14 (2022) 100137

Contents lists available at ScienceDirect

Array

journal homepage: www.sciencedirect.com/journal/array



Surrogate modeling of parametrized finite element simulations with varying mesh topology using recurrent neural networks

Lars Greve [a,*], Bram Pieter van de Weg [a,b]

[a] Volkswagen AG, D-38436, Wolfsburg, Germany
[b] Applied Mechanics and Data Analysis, Faculty of Engineering Technology, University of Twente, P.O. Box 217, 7500, AE Enschede, the Netherlands

A B S T R A C T

A machine learning based strategy is proposed for creating parametric surrogate models from parametrized finite element model simulation results. In the first major step, a unified nodal data structure is created from the topologically inhomogeneous set of finite element simulations. This is achieved by utilizing re-sampling and the coherent point drift method for node registration of the different designs. In the second major step, a parametric surrogate model is trained for predicting the initial coordinates using a fully-connected feed-forward neural network. Two different recurrent neural network modeling approaches are presented and compared for the prediction of various field quantities with different degrees of complexity and non-linearity.

For the first proposed modeling approach, a *node-by-node* prediction is applied, where the time series of each structural node is predicted independently via a compact long-short term memory (LSTM) model. For each node, the initial coordinates of the node are used as additional input features. For the second modeling approach, an *all-at-once* prediction is applied, where the time series of all structural nodes are predicted at once by training an LSTM model on a reduced output space obtained by principal component analysis (PCA).

Output fields exhibiting moderate non-linearity could be well predicted by both approaches, but only the *node-by-node* approach allowed an accurate generalized representation of a strongly non-linear and narrow field quantity representing the observed crack patterns within the finite element structure. The existence of a new yet unobserved crack pattern could be identified by the *node-by-node* approach and confirmed by subsequently running the corresponding FE simulation.

1. Introduction

Today, the crashworthiness of a vehicle is mostly assessed virtually using the finite element method (FEM), implemented in specialized crash simulation solvers, e.g. Ref. [1]. The shape and topology of all relevant structural components is represented by so-called finite elements in the crash simulation model, and the design of a crashworthy vehicle requires an iterative process. During the development process, design changes are introduced to improve the structural safety of the vehicle cell structure, e.g. in order to avoid severe structural buckling or the occurrence of fracture during a crash. These design changes often include local geometrical modifications, e.g. the appropriate placement of holes, beads, cut-outs, etc. In other words, the applied design changes modify the shape and/or mesh topology of the involved structural components. Specialized pre-processing software enables parametrizing the finite element models via corresponding geometrical features, where a feature for example defines the position of a hole position or the length of a bead, etc. Hence, a design of experiments (DOE) using different feature parameter combinations can be created and simulated with minimum manual effort. However, loading and manually assessing a large set of simulation results in a graphical post-processor is tedious and time-consuming for the engineer. Furthermore, it has to be noted that a technical solution usually has to meet several criteria, e.g. manufacturability, accessibility, order of assembly, etc. Hence, is often difficult to express the underlying engineering optimization problem in terms of a cost function in advance. A parametrized solution of the investigated simulation series is sought to allow for fast decision making, which enables a quick (real-time) manual assessment of the structural response depending on the feature parameter values.

A corresponding parametrized surrogate model maps the input features to the observed responses of the simulated DOE in a generalized way, such that valid predictions within the trained domain can be obtained. Several methods for creating surrogate models can be found in the literature. For many of these methods, dimensionality reduction plays a role in order to increase the computational efficiency. A comprehensive summary on dimensionality reduction methods is provided in Ref. [2]. Due to its simplicity, linear dimensionality reduction methods such as the proper orthogonal decomposition (POD), also referred to as principal component analysis (PCA), have been employed in conjunction with interpolation methods for fluid dynamics

* Corresponding author.
E-mail address: Lars.Greve@Volkswagen.de (L. Greve).

https://doi.org/10.1016/j.array.2022.100137
Received 4 November 2021; Received in revised form 10 January 2022; Accepted 22 February 2022
Available online 14 April 2022

applications [3] and for structural mechanics applications [4]. Non-linear dimensionality reduction methods, such as the proper generalized decomposition (PGD), have been used in conjunction with regression methods for obtaining the parametric solution response of a vehicle crash model [5].

Recently, machine learning concepts have been adopted for the creation of surrogate models in the field of structural mechanics for the prediction of time-dependent responses. Contributions can be divided into methods for time evolution prediction and methods for time-series prediction of parametrized FE structures.

In [6], a method for learning the time evolution of physical structures is developed using structure-preserving neural networks (SPNN), which comply with the first and second principles of thermodynamics without previous knowledge on the nature of the system. In Ref. [7], the time evolution of a mechanical structure is predicted using sparse auto-encoders for model reduction and a SPNN.

Recurrent neural networks (RNNs) are used for the prediction of sequences in many scientific and industrial fields. Among RNNs, the long short-term memory (LSTM) [8] and the slightly reduced gated recurrent unit (GRU) [9] models gained significant popularity. Whereas LSTMs have been successfully used for speech recognition and other language processing tasks, until recently only a few application were present for time-series prediction in the field of computer aided engineering (CAE).

In [10], Gated Recurrent Unit (GRU) neural networks are applied for modeling elasto-plastic deformation for arbitrary loading paths, where the predictions of a complex anisotropic hardening model could be well represented by the proposed network.

In [11], an LSTM model is applied for predicting the time-dependent structural deformation response of a parametrized FE structure including bifurcation phenomena. In Ref. [12], the features of an FE structure with hidden geometrical and numerical parametrization are revealed in an unsupervised manner using a convolutional neural network autoencoder (CNN-AE), where the corresponding latent space vector is then passed to an LSTM model for prediction of the time-series response of the whole structure.

[13] propose a GRU-based neural network for modeling the large deformation of lithium-ion battery cells. A unit cell model of the anode/separator/cathode assembly subjected to a large set of random loading paths provides the virtual training data base.

Recently, LSTM has been combined with POD to reduce the predictive dimension in a surrogate model for a large-scale elasto-plastic FE model [14], where the introduced time-dependent mechanical loads are introduced as input features over time.

In [15], a deep neural network featuring a “bottleneck”, referred to as “minimal state cells”, is embedded in a gated recurrent neural network for the time-series prediction of the deformation behavior of various types of materials. A Bayesian-based method is applied to a traditional LSTM cell in Ref. [16], using automatic relevance determination to impose sparsity and to assess the uncertainty of the predictions.

Changing mesh topologies are often present in real engineering situations, which adds some challenges to processing the inhomogeneous data. Since the internal structure of each design is different, it usually requires re-sampling and re-ordering of the data to a unified dimension space prior to training. In Ref. [17], mesh morphing methods based on discrete surface flow are used for obtaining a unified reference data configuration. Graph neural networks [18] naturally represent the native structure of a finite element mesh as a set of nodes connected by elements. However, maintaining the connectivity is not necessarily required for the present application. PointNet [19] allows processing of point clouds in arbitrary order, and has been applied to object classification and part segmentation tasks. Point set registration methods are used in other scientific fields such as computer vision or pattern recognition for aligning different point clouds to each other [20], where rigid, affine and elastic registration methods are distinguished. Rigid registrations only involve simple translations and rotations, whereas affine registrations include scaling and shearing. Hence, both rigid and affine registrations are very limited in representing a proper mapping between point clouds featuring severe geometrical differences. The coherent point drift (CPD) method [21] represents an elastic registration method, allowing a smooth coherent transition of one point cloud to another while maintaining topological features, e.g. holes. This makes the CPD method suitable for the given task in the present work.

In the present contribution, two approaches for obtaining parametrized surrogate models are presented and compared. A set of FE crash simulation results obtained by a DOE provides the training data for the surrogate model. The challenge of handling different FE mesh topologies is tackled by a two-step approach. In the first major step, the unified initial FE node coordinates of a geometrically parametrized FE structure are predicted by a fully-connected feed-forward neural network. For this purpose, the underlying mesh topologies of the designs are first re-sampled to a point cloud with a uniform data size. Then, the point clouds of the designs are registered to each other using the coherent point drift method and the data points are put into unified order via Euclidean distance based selection of nodal pairs. At this stage, the FE mesh representation is transformed from a “*nodes connected via elements*” representation to a unified pure point-based representation. In the second major step, various time series responses of the unified FE structure are predicted and represented on top of the previously predicted initial coordinates point cloud. Two different modeling approaches for the time-series prediction are investigated and compared, referred to by the *node-by-node* and the *all-at-once* approaches. The approaches are compared to corresponding finite element simulation data to access accuracy and predictability, and are also assessed with respect to model training time and model evaluation time.

In section 2, the investigated load case is discussed. In section 3, the parametric solution approaches for the initial coordinates and the time-series predictions are discussed. In section 4, the parametric solution approaches are compared to the finite element simulation results used for training. The *node-by-node* approach is also applied to unseen validation sets and used to reveal yet undetected response patterns. Section 5 concludes the paper.

2. Investigated load case

The proposed strategy is explained utilizing an academic load case representing a typical practical crash simulation situation including fracture assessment.

2.1. Parametrization of the finite element model

The finite element (FE) model represents a flat metal sheet specimen, which is subjected to tensile loading up to the onset of fracture, Fig. 1a.

The specimen includes several holes, where the positions of the centers of the two parametrized holes represent the investigated feature parameters pos_x and pos_y, Fig. 1a. The structure is subsequently referred to as the “two-moving-holes” (TMH) load case. The base mesh can be automatically re-meshed using the pre-processor Ansa [22], eventually leading to different mesh topologies, depending on the actual feature parameter values, Fig. 1b. During loading, different deformation and fracture patterns occur, depending on the position of the holes, as will be discussed in detail in the subsequent sections.

2.2. Material model and investigated outputs

The simulations have been carried out using an explicit crash FE solver [1] in conjunction with a user material subroutine. A predictive material model for the simulation of the deformation and fracture behavior of the sheet metal material has been developed by the authors in earlier research [23,24], based on a comprehensive experimental material characterization program at different loading conditions. In the

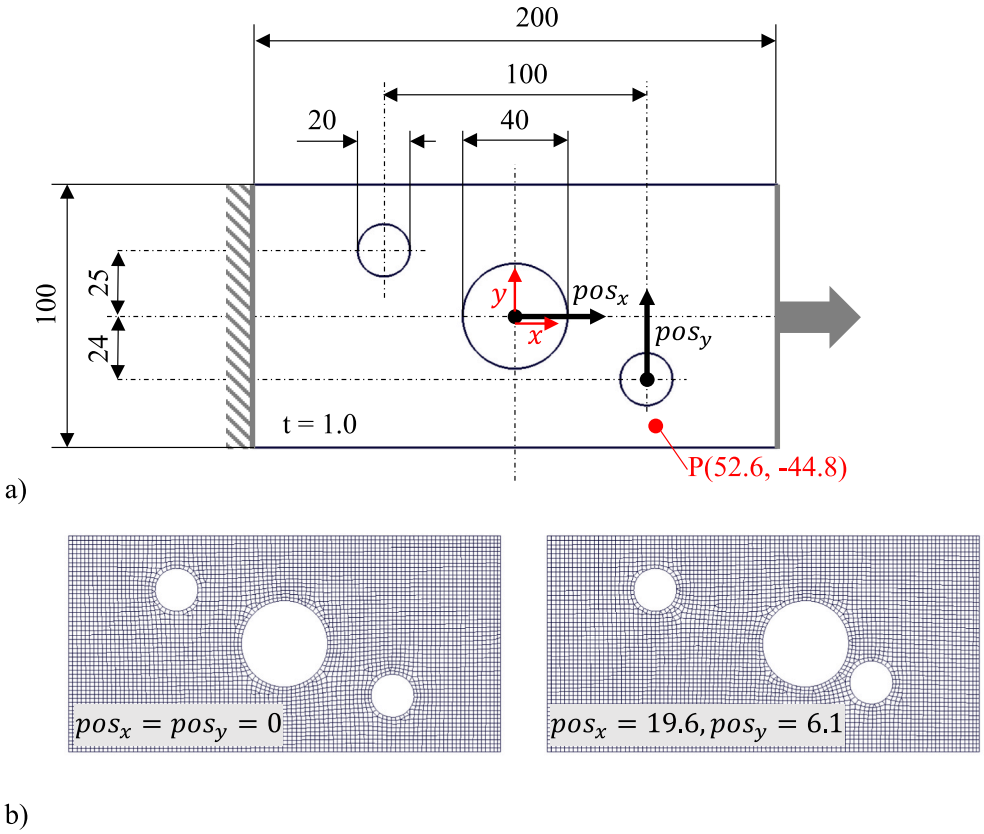


Fig. 1. The two-moving-hole (TMH) load case: a) base configuration for $pos_x = pos_y = 0$ (dimensions in millimeter); b) automatically created finite element models using an average element size of 2 mm. Evaluation point '*P*' is discussed in section 4.1.

present study, the model is used to provide the virtual training data set.

The underlying mechanical theory is only briefly revisited subsequently, since the present contribution focuses on the creation of parametric surrogate models. In the present work, the investigated outputs are the displacement field, the accumulated equivalent plastic strain field and the post-necking indicator field. Specifically, the nodal displacements (*DISP*) represent the structural deformation of the TMH structure over time for each finite element node and the equivalent plastic strain (*EPMX*) represents the accumulation of the equivalent plastic strain increments created for each finite element over time. Hence, state variable *EPMX* stores information about the irreversible deformation of the finite elements.

Post-necking indicator (*PNCK*) is used for the identification of structural cracks. Parameter *PNCK* is initially zero for all elements in the un-deformed state and becomes positive non-zero once the corresponding finite element enters the state of localized necking during deformation. The modified Hill/Storen-Rice model, based on the classical necking model by Ref. [25], and [26] is applied as a necking predictor. Once a finite element exceeds the corresponding load state dependent critical equivalent plastic necking strain, the post-necking criterion is activated, [23]. The calculation of *PNCK* is

$$PNCK = \frac{\sqrt{A_{neck}} \sum e^{\Delta \bar{\varepsilon}^p_{post}} - 1}{l_c}, \tag{1}$$

where A_{neck} is the area of the shell element at the onset of necking, $\Delta \bar{\varepsilon}^p_{post}$ are the increments of the equivalent plastic strain in the post-necking phase and l_c is a material-specific characteristic length. *PNCK* indicates a crack (fracture), when the parameter value exceeds unity, after which the corresponding finite element is removed from the simulation in order to represent the mechanical feedback of the opened crack.

The time-series of *DISP*, *EPMX* and *PNCK* are predicted by the proposed parametric surrogate models presented in the subsequent sections.

3. Parametric surrogate models

As mentioned in the introductory section, the overall parametric surrogate models are created using a two-step approach. In the first step, the initial coordinates of the unified FE structure are predicted as a function of the input feature parameter values. In the second step, selected time series responses of the unified FE structure are predicted and represented on top of the predicted initial coordinates point cloud. This includes predicting the displacements and various element field quantities, as already noted in section 2 and further explained in the subsequent sections.

3.1. Prediction of the initial coordinates

Design of experiments: According to Fig. 1, the investigated features and feature ranges are the x-position of the large hole, pos_x, varied in the range [0, 20] and the y-position of the small hole, pos_y, varied in the range [−5, 10].

The initial coordinates data can be readily obtained from the parametrized FE model and does not require extensive computation. In total, 25 designs are created using Latin hypercube sampling (LHS) [27] of the feature space. Due to the dense sampling, the geometrical difference between two designs with similar feature values is relatively small, which should allow for a reliable point cloud registration in between designs, as discussed in subsequent sections. The DOE table is provided in Table 1.

Pre-processing: Due to the different mesh topologies, the number of nodes of the underlying DOE FE simulation results vary for each design and range in between 4532 and 4955 nodes. This inhomogeneous data

Table 1
DOE feature values of the initial coordinates data.

Design Id	pos_x [mm]	pos_y[mm]	Design Id	pos_x [mm]	pos_y [mm]
0	19.6	6.1	13	18.8	5.5
1	2.0	0.7	14	5.2	6.7
2	4.4	3.1	15	15.6	−3.5
3	10	−1.7	16	6	1.3
4	16.4	7.3	17	11.6	−0.5
5	0.4	−2.9	18	12.4	1.9
6	17.2	4.9	19	10.8	8.5
7	14.8	−4.1	20	2.8	0.1
8	8.4	7.9	21	7.6	−1.1
9	18.0	−2.3	22	1.2	−4.7
10	3.6	9.7	23	9.2	2.5
11	13.2	9.1	24	14	3.7
12	6.8	4.3			

structure has to be converted to unified size and order. First, all designs are re-sampled to a uniform number of nodes using clustering and up-sampling. It has to be noted that the original FE mesh connectivity, represented by so-called shell or solid finite elements connecting the nodes (= the point cloud) and building the mesh topology, is not required anymore. Nodes lying on a free edge of the investigated two-moving-hole specimen and the remaining nodes are treated separately. This separation helps maintaining the original free edges contours in the unified model so that the original shape of the geometry can be easily recognized by the human eye. The maximum number of free and other edge nodes is determined over all designs, given by $maxNodes_i$, where the obtained number is multiplied by a chosen positive factor λ_i and converted to N_i representing the targeted number of nodes of the unified data structure:

$$N_i = \mathrm{int}(\lambda_i \; maxNodes_i), \tag{2}$$

where index $i = [\mathrm{free},\ \mathrm{other}]$ refers to the separated free or other edges nodes of the design.

Consequently, for $\lambda_i < 1$, the number of data points is reduced, whereas for $\lambda_i > 1$, additional redundant data points are created. This up-sampling allows to better maintain the data quality during the sequential point cloud registration process. For each design, the data is re-sampled as follows in order to represent N_i nodes. If N_i is larger than the number of the corresponding node set of the design, additional redundant nodes are created using randomly selected redundant nodes. If N_i is smaller than the number of the corresponding node set of the design, the nodes of the design are clustered to N_i clusters using the *K-Means* algorithm in scikit-learn [28] and for each cluster only the node closest to the cluster centroids is selected. For the TMH example, $\lambda_{\mathrm{free}} = 2.0$ is chosen in order to maintain a sharp representation of the free edge contours over the design sequence, whereas no up-sampling is performed for the other edges, $\lambda_{\mathrm{other}} = 1.0$.

The initial coordinate sets of the designs are sorted to a sequence list such that the geometrical difference between the designs becomes as small as possible. For the present TMH case, the design sequence is obtained by first sorting the designs with respect to the input feature value for the large hole, column pos_x in Table 1, and subsequently sorting with respect to the input feature values for the small hole, column pos_y in Table 1. The central design (design #3) is located in the center of the obtained design sequence, as sketched in Fig. 2.

The Coherent Point Drift (CPD) [21] method is then initially applied for transforming the neighbor source point clouds to the fixed target point cloud of central design, Fig. 3. A smooth coherent movement is imposed using the CPD method, preserving the topological structure of the point clouds. This ensures a consistent movement of the point clouds, which can be learned by the surrogate model. For the TMH example, the regions around holes are always mapped to each other between designs, so that the trained surrogate model will not predict unrealistic movements of nodes passing through the holes. The CPD process is then continuously repeated, where the previous source point clouds become the targets, and the next neighbor point clouds along the design sequence are selected as the new moving sources. Once the point clouds are mapped to each other, the selection of the unified and re-ordered nodes is performed by taking the minimum Euclidean distance to the reference nodes.

The basic CPD algorithm is briefly reviewed in Appendix A.1 and the complete discussion of the underlying mathematical foundations is provided in Ref. [21]. The CPD algorithm requires the setting of various free parameters, for which the following values have been used: the ratio of uniform/GMM mixing, $w = 0$, the variance of the Gaussian kernel matrix, $\beta = 10$, and the smoothness regularization parameter $\alpha = 0.05$.

For application, the implementation by Ref. [29] has been extended to a graphics processing unit (GPU) compliant version.

Network architecture and training: A simple fully-connected feed-forward network with one hidden layer with $N_{\mathrm{n}} = 16$ neurons and sigmoid activation functions is proposed for the prediction of the initial coordinates, Fig. 4, where $\widehat{\boldsymbol{y}}_{\mathrm{coords}} \in \mathbb{R}^{N_{\mathrm{c}} \times 1}$ contains the predicted 3-dimensional *xyz* coordinates of 3411 unified nodal points as a flattened vector, $N_{\mathrm{c}} = 3411 \times 3 = 10,233$. It is noted that all three *xyz* dimensions are considered for general applicability, although the present TMH example could actually be reduced to a two-dimensional (*xy*) load case.

The corresponding equation reads:

$$\widehat{\boldsymbol{y}}_{\mathrm{coords}} = \boldsymbol{\sigma}_1[\boldsymbol{W}_{\mathrm{c1}}\ \boldsymbol{\sigma}_0[\boldsymbol{W}_{\mathrm{c0}}\ \boldsymbol{x} + \boldsymbol{b}_{\mathrm{c0}}] + \boldsymbol{b}_{\mathrm{c1}}], \tag{3}$$

with the sigmoid function for neural network layer i being applied to each element of input argument vector $\boldsymbol{a}$:

$$\boldsymbol{\sigma}_i[\boldsymbol{a}] = \frac{1}{1 + e^{-\boldsymbol{a}}}. \tag{4}$$

The hidden and output layer weight matrices and bias vectors are $\boldsymbol{W}_{\mathrm{c0}} \in \mathbb{R}^{N_{\mathrm{n}} \times N_{\mathrm{f}}}$, $\boldsymbol{W}_{\mathrm{c1}} \in \mathbb{R}^{N_{\mathrm{c}} \times N_{\mathrm{n}}}$ and $\boldsymbol{b}_{\mathrm{c0}} \in \mathbb{R}^{N_{\mathrm{n}} \times 1}$ and $\boldsymbol{b}_{\mathrm{c1}} \in \mathbb{R}^{N_{\mathrm{c}} \times 1}$, respectively, where $N_{\mathrm{f}} = 2$ is the number of input features. The total number of free parameters of the neural network is $N_{\mathrm{n}} \times N_{\mathrm{f}} + N_{\mathrm{c}} \times N_{\mathrm{n}} + N_{\mathrm{n}} + N_{\mathrm{c}} = 174,009$.

As it will be shown subsequently, the proposed simple network architecture can well predict the parametrized unified initial coordinates for the present TMH example. However, it has to be noted that the choice of the network architecture and size depends on the complexity of the considered load case. More complex examples might require

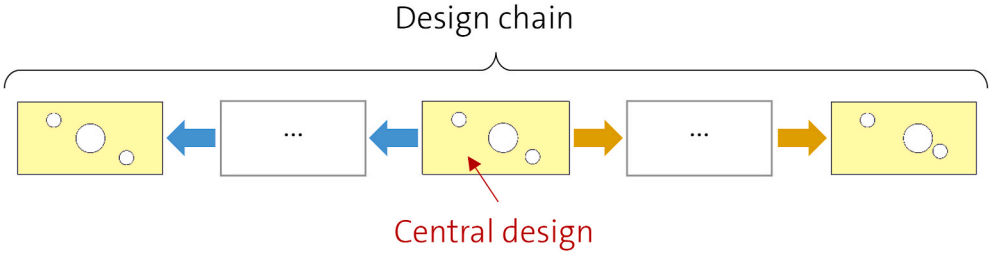


Fig. 2. Schematic of the design sequence for chained point cloud registration originating in the central design towards both ends of the design sequence.

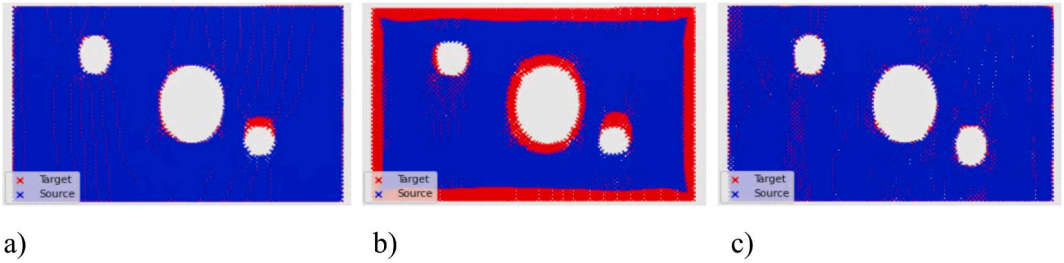


Fig. 3. Exemplary transformation of a source point cloud (blue crosses) to a target point cloud (red): a) initial state showing the geometrical difference of the designs; b) intermediate transformation; c) final fully transformed state.

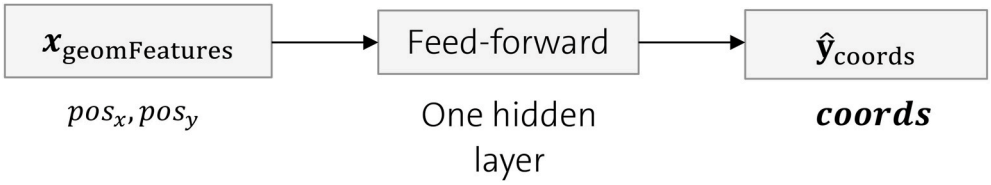


Fig. 4. Fully-connected feed-forward neural network architecture for the prediction of the initial coordinates.

additional hidden layers and/or an increased number of neurons.

The input feature vector $\boldsymbol{x}_{\text{geomFeatures}} \in \mathbb{R}^{N_f \times 1}$ and the corresponding coordinate target outputs $\boldsymbol{y}_{\text{coords}}$, are scaled in the range [0,1] prior to the training. For application, the corresponding prediction, $\hat{\boldsymbol{y}}_{\text{coords}}$, has to be transformed to the original value range by inverse scaling.

Gaussian dropout [30] available in Tensorflow [31] with a dropout probability of $rate = 0.05$ is applied after the input and hidden layers of the feed-forward network, introducing noise with a standard deviation of

$$S = \sqrt{\frac{rate}{1 - rate}}. \tag{5}$$

A train to test data split of 9:1 is used. During training, the train data is presented using a batch size including all designs. The training of the model is performed over 100,000 epochs using TensorFlow utilizing the Adam [32] optimizer with mean-squared-error loss function and an initial learning rate of 0.001, Fig. 5. No overfitting was observed for the validation set. The obtained coefficient of determination, R^2, is 0.9998. For the final training, all available data, including the validation set, has been used and the training was performed over the same number of epochs.

Model application: The results of the parametrized surrogate model for the initial coordinates is visualized for different values of the input features, Fig. 6. The holes follow a smooth path along the axes, confirming the expected movement.

The simple Euclidean distance selection applied after the CPD transportation does not guarantee unique one-to-one mappings. It is possible that several source nodes will be selected by the same target node and consequently, some target nodes will not be selected at all, which gradually introduces apparent gaps in the point clouds towards the end of the design sequence. This effect is visible in Fig. 6, where the raw initial coordinates are plotted without a surface representation. A feature parameter state close to the central design (Fig. 6, middle) leads to a more homogeneous point density than close to both ends of the design sequence (Fig. 6, left and right). It is emphasized that this effect does not cause data quality reduction of the selected data. For practical application, and also in the subsequent sections, the predicted initial shape of the TMH specimen is visualized as a surface using Delaunay triangulation instead of a point cloud.

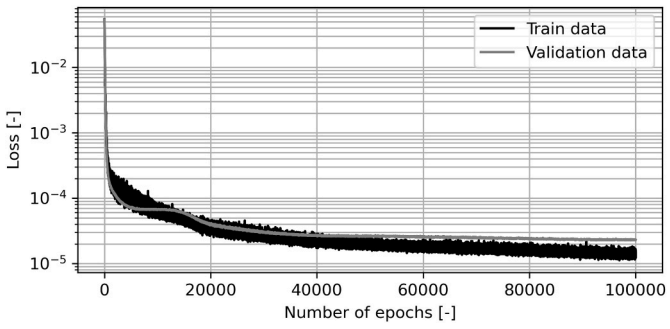


Fig. 5. Train and validation loss evolution over epochs for the training of the initial coordinates model.

3.2. Prediction of the time series

Two different time series prediction approaches are compared, namely the *node-by-node* and the *all-at-once* approach. For the *node-by-node* approach, the times series of each node is predicted independently, whereas for the *all-at-once* approach, the time series of all nodes are predicted conjointly. The learned predictions include the displacements (*DISP*), the maximum equivalent plastic strain (*EPMX*) field, and the post-necking (*PNCK*) field, indicating the onset of fracture.

Design of experiments (DOE): The investigated features and feature ranges are identical to the prediction of the initial coordinates in sub-section 3.1. However, only a moderate number of 10 simulations will be performed using Latin Hypercube space filling. The chosen number of designs is assumed to be close to the least number of designs required for properly capturing the complex deformation and fracture patterns within the chosen range of the input feature values. Usually, the number of designs is kept to a minimum, since running the FE simulations is computationally costly for real-life applications. The DOE table is provided in Table 2.

Pre-processing: The pre-processing is identical to the strategy for the initial coordinates (see section 3.1). Again, the design sequence is obtained by sorting with respect to the input feature values. It is important to note that the central design is still represented by the central design obtained for the initial coordinates. This leads to a consistent unified data representation, although the DOE data base for the initial coordinates and the time series data are different. The obtained data re-ordering selection is directly applied to all other outputs (*DISP*, *EPMX* and *PNCK*). It is noted that the original data for *EPMX* and *PNCK* is related to finite elements and not to the representing finite

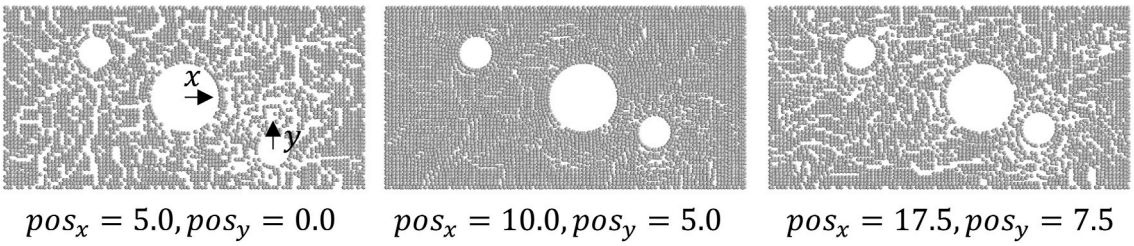


Fig. 6. Prediction of the raw initial coordinates for different input feature values illustrating the x-movement of the large hole and the y-movement of the small parametrized hole.

Table 2
DOE feature values of the time-series data.

Design Id	pos_x [mm]	pos_y [mm]
0	3.0	7.75
1	11	−1.25
2	9.0	3.25
3	17	6.25
4	1.0	0.25
5	15	−2.75
6	13	9.25
7	7.0	−4.25
8	5.0	1.75
9	19	4.75

element nodes. Hence, the elemental values are converted to the corresponding nodal values, where each node receives the corresponding contribution of all connected elements as a mean value.

3.2.1. The node-by-node approach

For the *node-by-node* approach, the time series of each node is predicted independently for each nodal coordinate. The data flow for the *node-by-node* model is shown in Fig. 7.

The basic global geometrical parameters of the parametrized FE model (pos_x, pos_y) represent the basic input features for each node. Together with the initial coordinates of each node, the input feature vector per node $\boldsymbol{x}_{\text{allFeatures},i}$ becomes

$$\boldsymbol{x}_{\text{allFeatures},i} = \left[pos_x, pos_y, coords_{x,i}, coords_{y,i}, coords_{z,i}\right]^{\text{T}}. \tag{6}$$

The input features are passed to a long short-term memory (LSTM) recurrent network [8], and are further propagated through a time-distributed dense layer for predicting the desired response $\hat{\mathbf{y}}_{t,i}^{\text{NodeByNode}} \in \mathbb{R}^{N_D \times 1}$ of node i represented by the initial coordinates $coords_{xyz,i}$ at time step t of the structure, where N_{D} is the output dimension (N_{D} =3 for *DISP* and $N_{\text{D}} = 1$ for *EPMX* and *PNCK*).

The data set contains $N_{\text{DOE}} = 10$ designs, where the number of nodes in the unified model is $N_{\text{n}} = 5426$. That is, for the *node-by-node* approach, the training set comprises $i \in [1..N_{\text{n}} \times N_{\text{DOE}}]$ nodes. The time step is denoted by $t \in [1..N_t]$, where the total number of time steps of the data set is $N_{\text{t}} = 11$.

A standard 4-gated LSTM cell is used, Fig. 8, with the forget gate $\boldsymbol{f}_t$, the input gate $\boldsymbol{i}_t$, the cell state gate $\boldsymbol{C}_t$, and the output gate $\boldsymbol{o}_t$. The corresponding equations are

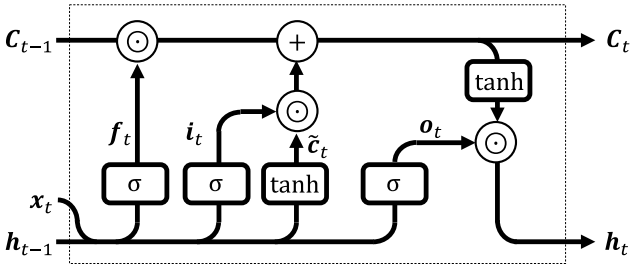


Fig. 8. Recurrent neural network architecture of the node-by-node approach.

$$\begin{aligned}
\boldsymbol{f}_t &= \boldsymbol{\sigma}\left[\boldsymbol{W}_f[\boldsymbol{x}_t, \boldsymbol{h}_t] + \boldsymbol{b}_f\right],\\
\boldsymbol{i}_t &= \boldsymbol{\sigma}[\boldsymbol{W}_z[\boldsymbol{x}_t, \boldsymbol{h}_t] + \boldsymbol{b}_z],\\
\tilde{\boldsymbol{c}}_t &= \mathbf{tanh}[\boldsymbol{W}_{\tilde{c}}[\boldsymbol{x}_t, \boldsymbol{h}_t] + \boldsymbol{b}_{\tilde{c}}],\\
\boldsymbol{C}_t &= \boldsymbol{f}_t \odot \boldsymbol{C}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t,\\
\boldsymbol{o}_t &= \boldsymbol{\sigma}[\boldsymbol{W}_o[\boldsymbol{x}_t, \boldsymbol{h}_t] + \boldsymbol{b}_o],\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \mathbf{tanh}[\boldsymbol{C}_t].
\end{aligned} \tag{7}$$

where $\boldsymbol{W}_g$ and $\boldsymbol{b}_g$, with $g \in \{f, z, \tilde{c}, o\}$ referring to the corresponding gate index, are the corresponding weight matrices and bias vectors containing the free parameters of the model.

The sigmoid activation function is denoted $\boldsymbol{\sigma}$, equation (3), and the hyperbolic tangent activation function being applied to each element of input argument vector $\boldsymbol{a}$ reads:

$$\mathbf{tanh}[\boldsymbol{a}] = \frac{e^{\boldsymbol{a}} - e^{-\boldsymbol{a}}}{e^{\boldsymbol{a}} + e^{-\boldsymbol{a}}}. \tag{8}$$

Symbol $\odot$ denotes the Hadamard product for element-wise multiplication. The output vector of the LSTM cell is denoted $\boldsymbol{h}_t$.

The chosen size of the output vector of the LSTM cell, $\boldsymbol{h}_t$, defines the size of the cell state parameters $\boldsymbol{C}_t$ and the size of the corresponding weights and biases. A Gaussian dropout layer is applied after the LSTM cell.

A subsequent time-distributed dense layer with weight matrix $\boldsymbol{W}_{\text{d}}$, bias vector $\boldsymbol{b}_{\text{d}}$, and activation function $\boldsymbol{f}_{\text{d}}$ converts the shape of the LSTM output vector $\boldsymbol{h}_t$ to the requested shape of the output vector $\hat{\mathbf{y}}_{t,i}^{\text{NodeByNode}}$:

$$\hat{\mathbf{y}}_{t,i}^{\text{NodeByNode}} = \boldsymbol{f}_{\text{d}}[\boldsymbol{W}_{\text{d}} + \boldsymbol{b}_{\text{d}}], \tag{9}$$

For the present analysis, $\hat{\mathbf{y}}_{t,i}^{\text{NodeByNode}}$ can represent one of the investigated outputs *DISP*, *EPMX*, or *PNCK*. It is noted that for training, the

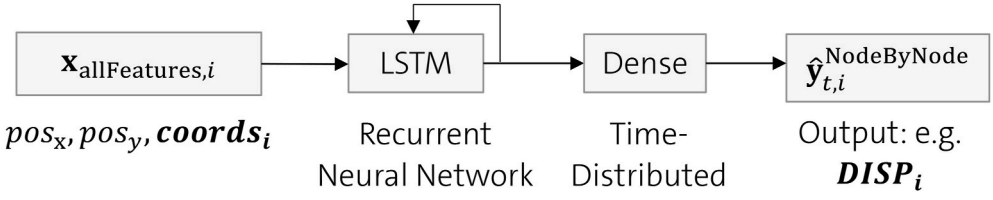


Fig. 7. Recurrent neural network architecture of the *node-by-node* approach.

initial coordinates of the performed DOE are used as an input, whereas for application, the initial coordinates obtained from the initial coordinates prediction model according to Fig. 4 and equation (2) are used as an input.

The proposed *node-by-node* network architecture features a compact input and output layer, which is independent of the size of the investigated structure. For the investigated TMH load case, the input feature vector only contains 5 entries, namely two parametrized hole positions and the initial *xyz* coordinates. The output vector contains the *xyz* displacements for the *DISP* output and a single output value for the *EPMX* and *PNCK* outputs. This compact network design leads to a small number of free weights and does not require the introduction of further (lossy) dimensionality reduction methods. Furthermore, the *node-by-node* method is topology-independent, since its prediction does not depend on the FE mesh topology or the order of the underlying nodal coordinates anymore.

Training preparation: For all predicted outputs, the LSTM utilizes 128 units, leading to an equivalent size of the output vector $\boldsymbol{h}_t$, and a linear output activation function f_{d} in equation (8). Accordingly, the neural network contains 68,608 free weights for the LSTM cell and 387 free weights for the dense layer for the *DISP* output (*xyz*). The dense layers for the *EPMX* and *PNCK* outputs contain 129 free weights due to the reduced (scalar) size of the outputs.

The DOE for the time series prediction comprises 10 designs. As mentioned before, for the proposed *node-by-node* network architecture, the time series of each node output is predicted independently. Hence, the DOE data comprising 10 FE simulations (designs) is re-arranged accordingly so that each node data represents one design, Fig. 7. Following this procedure, the formal number of designs changes from 10 (all evaluated FE simulations) to 47,848 (all nodes of all evaluated FE simulations), where the input feature vector of each node is extended by its initial coordinates. It is noted that the order of the re-arranged data set does not play a role for the *node-by-node* approach. The re-arranged input feature vectors $\boldsymbol{x}_{\mathrm{allFeatures},i}$ and the output target vector $\mathbf{y}_{t,i}^{\mathrm{NodeByNode}}$ are scaled into the range [0, 1] in advance. For application of the model, the predicted values have to be transformed back to the original value range via inverse scaling. The re-arranged DOE data is finally shuffled and split with a ratio of 9:1 into train and test data sets, where only the train data set is used for the training.

Training: Again, TensorFlow is used for training of the parametric solution model, utilizing the Adam optimizer with mean-squared-error loss function and an initial learning rate of 0.001. Gaussian dropout with a dropout probability of $rate = 0.1$ is applied. During training, the train data is presented using a batch size of 4096 data points. A converged state could be obtained for all outputs, where the losses over epoch evolution for the *DISP*, *EPMX* and *PNCK* outputs are shown in Fig. 9.

A noisy response, exhibiting some intermediate peaks, is observed, especially for the challenging strongly non-linear *PNCK* output, Fig. 9c. To study this behavior, the training of *PNCK* has been repeated using a larger batch size equivalent to the complete data set size (batch size = 47,848), Fig. 10. The noise frequency is significantly reduced, but the training time increases from 529s for a batch size of 4096 to 2029s for a batch size of 47,848 for obtaining a similar loss level. Hence, the smaller batch size is chosen for the present study.

For all outputs, a similar loss evolution is observed for the train and validation sets, indicating a good generalization capability of the *node-by-node* approach. The achieved R^2 values are summarized in Table 5, indicating a very good data representation of the *node-by-node* model. Finally, the *node-by-node* model is re-trained using all available data over the same number of epochs. The results using the *node-by-node* model are discussed in section 4.

3.2.2. The all-at-once approach

The *all-at-once* approach is investigated as a potential alternative to the *node-by-node* approach. The neural network architecture for the *all-at-once* approach is presented in, Fig. 11. In contrast to the *node-by-node* approach, the time series of all nodes of the considered unified FE structure are predicted conjointly for the *all-at-once* approach. Hence, the output vector $\widehat{\mathbf{y}}_t^{\mathrm{AllAtOnce}}$ can become large for real-life FE structures, and the corresponding size of the output layer weight matrix becomes large, too. For efficient training of the neural network, dimensionality reduction using Principal Component Analysis (PCA) is applied for compressing the target data $\mathbf{y}_t^{\mathrm{AllAtOnce}}$ to a truncated representation in a

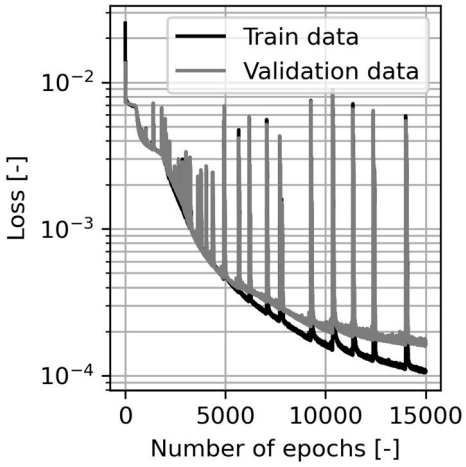


Fig. 10. Train and validation loss evolution over epochs for the training of the "node-by-node" model using a batch size equivalent to the complete data set size.

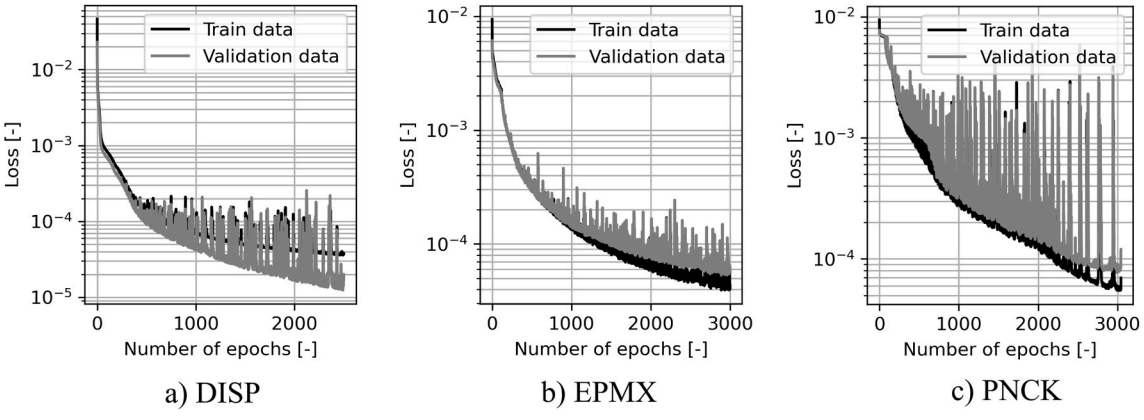


Fig. 9. Train and validation loss evolution over epochs for the training of the *node-by-node* model.

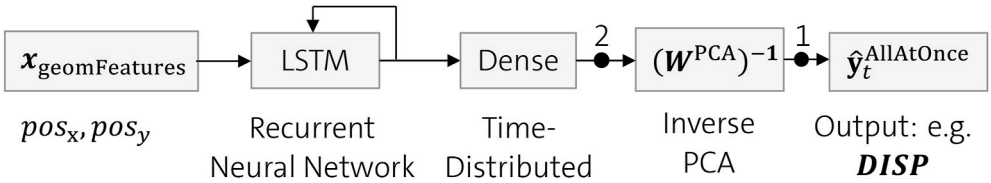


Fig. 11. Recurrent neural network architecture of the *all-at-once* approach.

pre-processing step. Then, the inverse weight matrix of the PCA is applied for reconstruction of the target data, as represented by the output layer $(W^{\text{PCA}})^{-1}$ in Fig. 11. It will be shown that the remaining number of free weights of the *all-at-once* approach are similar to the number of weights for the *node-by-node* approach.

Dimensionality reduction: In order to reduce the dimensionality of the training data, a linear dimensionality reduction method is applied prior to training for the *all-at-once* approach:

$$Y^{\text{PCA}} = Y\, W^{\text{PCA}}. \tag{10}$$

In equation (9), $Y_{n\,\text{x}\,p}$ is the design output matrix, where n is the number of designs and p is the number of parameters per design, the flattened *xyz* displacements (*DISP*) over all time steps. $Y^{\text{PCA}}_{n\,\text{x}\,L}$ is the truncated design matrix, where L is the reduced number of parameters with $L \ll p$. The orthogonal weight matrix $W^{\text{PCA}}_{p\,\text{x}\,L}$ is obtained by principal component analysis (PCA) [33], where the columns of the orthogonal transformation matrix W^{PCA} are the eigenvectors of the covariance matrix $Y^{\text{T}}Y$.

A standard scaler imposing zero mean and unit variance of the data points is applied to the Y data in advance. The corresponding inverse scaling process for the PCA is visualized by black dot (#1) in Fig. 11. The transformed data Y^{PCA} is then scaled to the range [0, 1] to ensure compatibility with the LSTM model. The corresponding inverse scaling is visualized by black dot (#2) in the propagation graph in Fig. 11.

The truncated transformation matrix W^{PCA} is calculated using the Scikit-learn PCA package [28], where the number of reduced parameters, L, is determined by truncating W^{PCA} after exceeding an overall explained variance of 0.999 for the *DISP*, *EPMX* and *PNCK* outputs.

The PCA is first performed using a train-to-test ratio of 9:1, revealing a satisfying R^2 value for the displacements (*DISP*), a sufficient R^2 value for the equivalent plastic strain field (*EPMX*), and an insufficient R^2 value for the post-necking field (*PNCK*). The obtained reduced data sizes and R^2 values are summarized in Table 3.

Further investigation of the PCA for *PNCK* using different designs for the validation set (manual cross validation) revealed R^2 values approximately ranging in between 0.5 and 0.75, indicating a generalization problem. Despite these potential issues and for the sake of completeness, the PCA is completed for all investigated outputs (including *PNCK*) and used for subsequent training of the *all-at-once* models. For the final PCA computation, the complete data set is used without keeping a validation set aside. The original target data can then be approximated by applying the inverse (=transpose) of the orthogonal transformation matrix to the truncated target data matrix:

$$Y' = Y^{\text{PCA}} \left(W^{\text{PCA}}\right)^{-1}. \tag{11}$$

Table 3
Summary of dimensionality reduction using PCA and a targeted explained variance of 0.999.

Output	Original data size (p) [−]	Reduced data size (L) [−]	R^2 (train set) [−]	R^2 (vali set) [−]
DISP	16278	15	0.9996	0.9939
EPMX	5426	38	0.9993	0.9175
PNCK	5426	44	0.9993	0.5613

Network architecture: For the *all-at-once* approach, the time series of all nodes of the represented unified FE structure are predicted at once, Fig. 11, where $x_{\text{geomFeatures}}$ represents the input feature vector.

$$x_{\text{geomFeatures},i} = \left[pos_x, pos_y\right]^{\text{T}}. \tag{12}$$

It is noted that in contrast to the *node-by-node* approach, the initial coordinates are not used as input features for the *all-at-once* approach. The architecture of the LSTM and time-distributed dense layers is identical to the *node-by-node* approach, where a Gaussian dropout layer is applied after the LSTM cell. The real values of the input feature vector $x_{\text{geomFeatures}}$ are scaled into the range [0, 1] in advance.

The output of the dense layer represents the latent space, which is decompressed to the final data space $\hat{y}_t^{\text{AllAtOnce}} \in \mathbb{R}^{N_n \cdot N_D \times 1}$, representing *DISP*, *EPMX* or *PNCK*, by applying the inverse transformation matrix $(W^{\text{PCA}})^{-1}$ according to equation (10).

Again it is noted that the chosen size of the output vector of the LSTM cell, h_t, defines the size of the cell state parameters C_t and the size of the corresponding weights and biases. For the *all-at-once* model, all time-series quantities to be learned from the DOE data, e.g. *DISP*, *EPMX* or *PNCK*, refer to the corresponding initial coordinate locations.

It has to be kept in mind that the predicted time-series will be visualized on top of the predicted initial coordinates. Consequently, a high accuracy of the initial coordinates prediction model is required, especially for the *all-at-once* approach, otherwise any predicted time-series would be displayed at a wrong position in space. The *node-by-node* approach is less sensitive to this error chain, since the *node-by-node* model uses the predicted initial coordinates for the subsequent time-series prediction without assuming that the real and predicted coordinates match perfectly.

Training preparation: For all predicted outputs, the LSTM utilizes 128 units (size of the output vector h_t) and a linear output activation function f_d. The DOE for the time series prediction comprises 10 designs. For the *all-at-once* network architecture, the time series of a complete design is predicted conjointly in one step.

The values of the input feature vector $x_{\text{geomFeatures}}$ are scaled in the range [0, 1]. A train-to-test ratio of 9:1 is imposed for the data set, that is, one design is used for validation.

Training: Again, TensorFlow is used for training of the parametric solution model, utilizing the Adam optimizer with mean-squared-error loss function and an initial learning rate of 0.001. During training, the train data is presented using a batch size containing the whole training set. The losses over epoch evolution for the *DISP*, *EPMX* and *PNCK* outputs are shown in Fig. 12. In contrast to the *node-by-node* model, the validation sets stagnate at a relatively high loss level, whereas the training sets still converge at a low loss level.

The overall coefficients of determination (R^2), representing the comparison of the final predictions, $\hat{y}^{\text{AllAtOnce}}$ and the corresponding target values $y^{\text{AllAtOnce}}$, are summarized in Table 5. A reasonable generalization can be achieved for *DISP* and *EPMX*. As indicated by the analysis of the dimensionality reduction process, section 3.2.2, the low overall value for the *PNCK* indicates the expected poor generalization of the trained network. The *all-at-once* model is re-trained using all available data over the same number of epochs. The results of the *all-at-once* model are discussed in section 4.

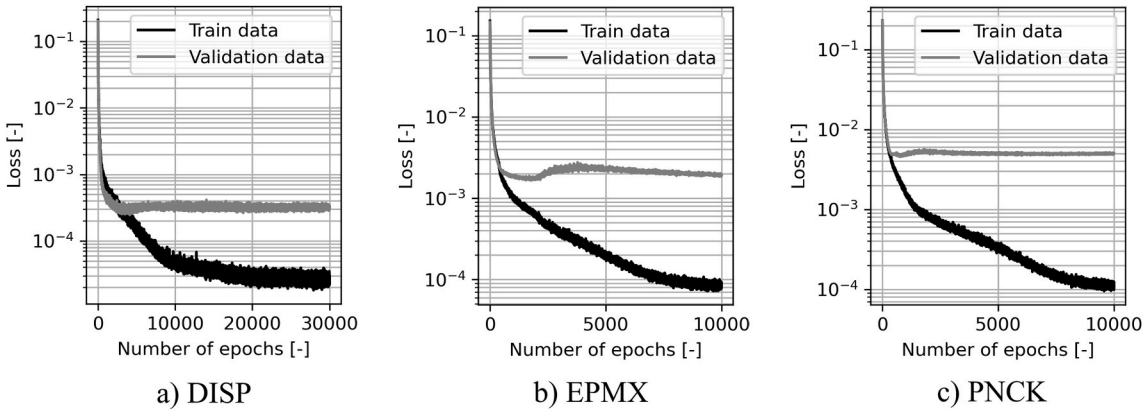


Fig. 12. Train and validation loss evolution over epochs for the training of the *all-at-once* model.

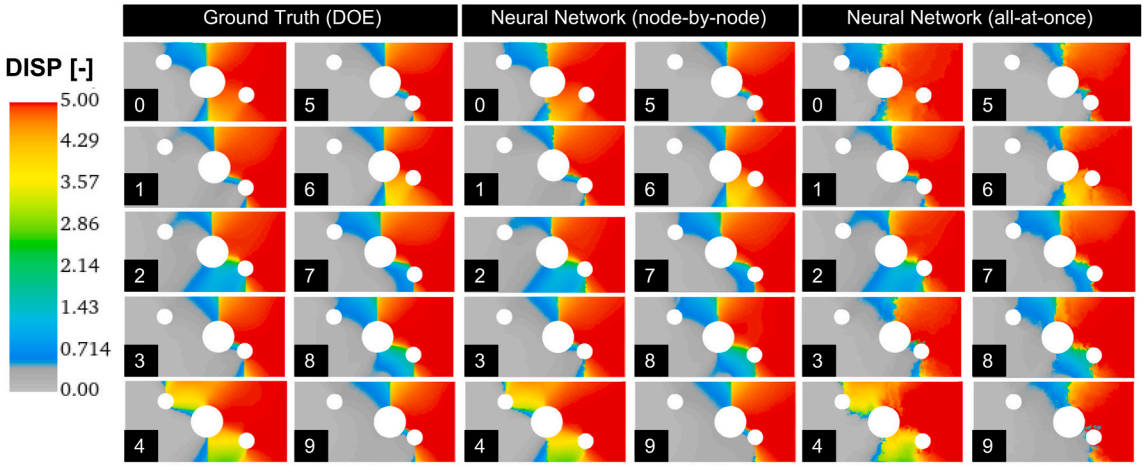


Fig. 13. Comparison of *DISP* output at the final time step: DOE results, "node-by-node" approach results and "all-at-once" approach results.

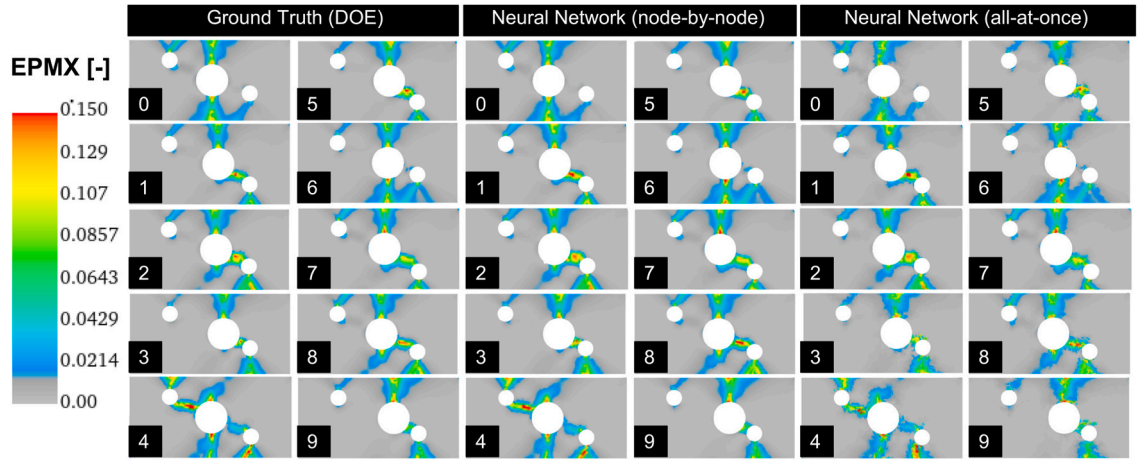


Fig. 14. Comparison of *EPMX* output at the final time step: DOE results, "node-by-node" approach results and "all-at-once" approach results.

4. Discussion of results

In this section, the presented modeling approaches are applied to the DOE data and unknown validation data for comparison of the predictability. Furthermore, the model evaluation times and the model training times are compared.

4.1. Comparison of model accuracy

Representation of trained data: In the previous sections, the architecture and training of parametric solution models for the *node-by-node* and the *all-at-once* approaches have been discussed. The trained output includes the nodal displacement (*DISP*) field, the equivalent plastic strain (*EPMX*) field, and the field of the post-necking indicator (*PNCK*). The *DISP*, *EPMX* and *PNCK* outputs of the *node-by-node* and *all-*

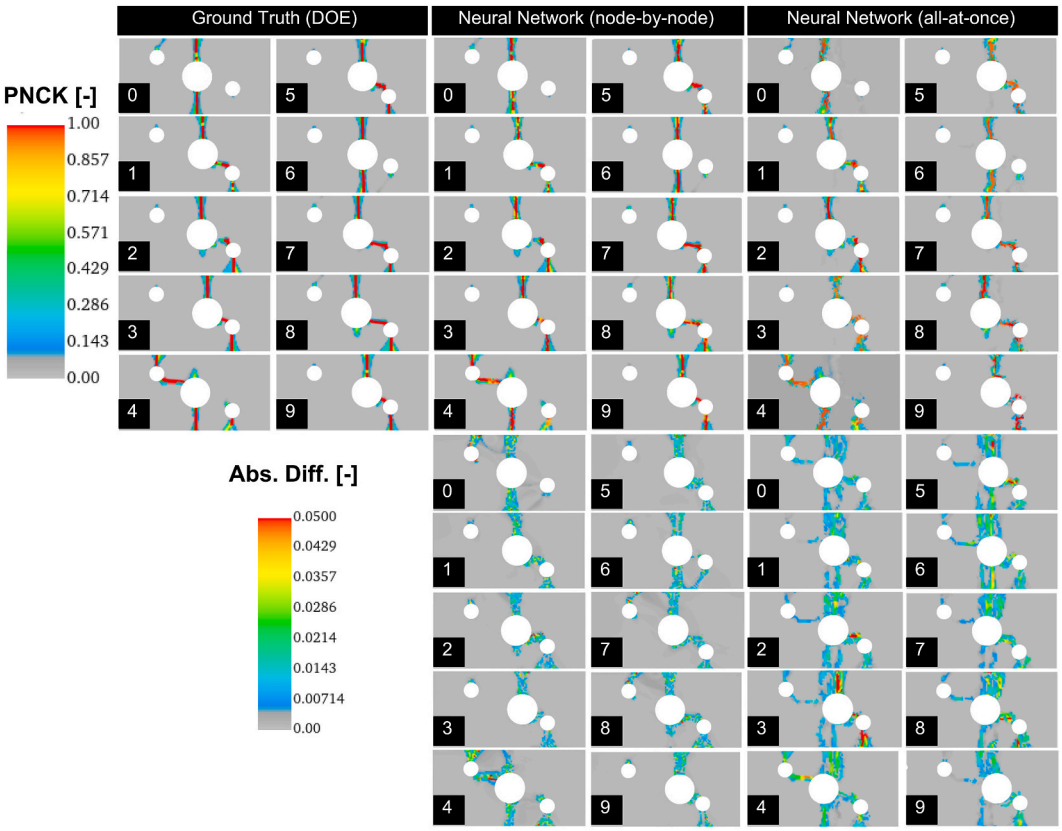


Fig. 15. Comparison of *PNCK* output at the final time step: DOE results, "node-by-node" approach results, "all-at-once" approach results and absolute differences.

at-once approaches, trained using the complete DOE data set (no validation set kept aside) are compared to the corresponding ground truth FE simulations obtained from the DOE in Fig. 13–15, where the predicted values for the *DISP* (Euclidean norm), *EPMX* and *PNCK* outputs are color-coded on top of the predicted initial coordinates and displacements. In the figures, the parametrized small hole is located on the right hand side. For visualization, the predicted initial coordinates are transformed to a surface using Delaunay triangulation [34].

For *DISP*, a very good representation of the DOE is achieved for both approaches. Only designs #6 (yellow-orange region) and #9 (around the small parametrized hole) of the *all-at-once* approach in Fig. 12 shows some minor mismatch of the displacement field compared to the ground truth results.

For *EPMX*, a good representation of the DOE is achieved for both approaches. The predicted responses using the *node-by-node* approach are slightly in better agreement with the ground truth responses than the *all-at-once* approach, compare e.g. the reddish region in design 4 in Fig. 14.

Both modeling approaches can reasonably represent the general patterns of the *PNCK* field of the 10 designs. It is however noted that the representation of *PNCK* is significantly sharper for the *node-by-node* approach than for the *all-at-once* approach, Fig. 15. For *PNCK*, the absolute difference of the predictions and the corresponding DOE values are plotted in Fig. 15, clearly revealing increased deviations for the *all-at-once* approach.

It can be shown that further increasing the number of epochs for the all-at-once approach does not improve the accuracy of the all-at-once approach. The training is intentionally stopped at the quasi-converged states shown in Fig. 12 in order to allow a fair comparison of the model training time. Furthermore, some minor overfitting is already observed for the present configuration. Hence, further increasing the number of free parameters by introducing a larger number of neurons is not desirable.

The poor *PNCK* prediction using the *all-at-once* approach could be explained by several remaining issues:

a) **Dimensionality reduction**: As mentioned in section 3.2.2, the dimensionality reduction for the *PNCK* output revealed a poor R^2 value for the validation set, indicating that the PCA does not create a generalized dimensionality reduction. The *PNCK* response of the two-moving-holes example is characterized by very narrow zones of high *PNCK* values surrounded by large domains of zero or close-to-zero values. A dimensionality reduction method aims to maintain the bulk of information, and the high *PNCK* values in the narrow zones are partially considered outliers and are eventually dropped in the truncated representation without losing a significant amount of explained variance.
b) **Number of designs representing the validation set**: an accurate and generalized prediction of the *PNCK* response for the *all-at-once* approach potentially requires more designs. In contrast to the *node-by-node* approach, complete designs are removed from the training set for validation for the *all-at-once* approach.
c) **Absence of initial coordinates data**: In contrast to the *node-by-node* approach, the *all-at-once* approach cannot make use of the local initial coordinates data, since the prediction is only based on the global geometrical input features pos_x and pos_y. As a consequence, the *all-at-once* approach can only establish global relations (interpolations) between the snapshots (designs) it has seen during the training In contrast, the *node-by-node* approach allows independent (uncoupled) learning of the time-series of individual nodes, depending on their initial coordinates.

Validation: For validation of the created parametric surrogate models for the *node-by-node* and *all-at-once* approaches, a sequence of additional FE simulation is evaluated, where the positions of the parametrized holes are varied along a specific path. The first simulation

is performed at feature values of $pos_x = 5.0$ mm and $pos_y = 5.0$ mm. Then, the position of the large hole is increased $pos_x = \{6.0, 7.0, 9.0, 10.0\}$ mm over a series of simulations while keeping the position of the small hole constant at $pos_y = 5.0\ mm$. Finally, the position of the large hole is kept constant at $pos_x = 10$ mm, and the position of the small hole is varied over a series of FE simulation: $pos_y = \{5.5, 6.0, 6.5, 7.0\}$ mm.

The final *PNCK* field of these validation FE simulations are shown in column "Ground Truth" in Fig. 16, where the *PNCK* values at the last time step are color-coded on the predicted initial coordinates.

In the initial configuration (state #0), a vertical crack passing through the large hole is present. As the parametrized large hole "moves" towards the parametrized small hole, the vertical crack switches (state #3) to a vertical-horizontal-vertical crack passing through both holes. When increasing the y-position of the parametrized small hole, starting at state #5, the crack switches back to a pure vertical line (state #7).

The corresponding predicted responses using the *node-by-node* and *all-at-once* approaches are also shown in Fig. 16, where critical predictions are marked with a flash symbol. A very good general prediction of the *PNCK* response is obtained by the *node-by-node* approach, where only the second switch from the vertical-horizontal-vertical crack to the pure vertical crack (state #6) is predicted slightly too early.

An overall poor prediction is obtained for the *all-at-once* approach. In the prediction, the different crack modes are not separated properly. Several crack patterns are predicted simultaneously and the level of the *PNCK* values is significantly lower than in the ground truth FE simulations.

The PNCK response of a specific point *P* (see Fig. 1) is investigated in further detail. The initial coordinates of the evaluation point *P* are close to a potential vertical crack occurring under the small moving hole. Since the location of an occurring vertical crack under the small vertically moving hole is independent from the feature values, the chosen evaluation coordinates can be readily used to establish a crack/no crack *PNCK* map at *P* as a function of the feature values, Fig. 17a. The feature value path according to the states shown in Fig. 16 are visualized as a black chain of arrows. By increasing pos_x according to the horizontal arrow in Fig. 17, the maximum *PNCK* value of the evaluation point close to the crack zone increases close to unity. Then, by increasing pos_y (vertical arrow in Fig. 17), the PNCK value decreases again. These observations are in agreement with the responses shown in Fig. 16, where a vertical crack under the small moving hole appears and disappears along the state path.

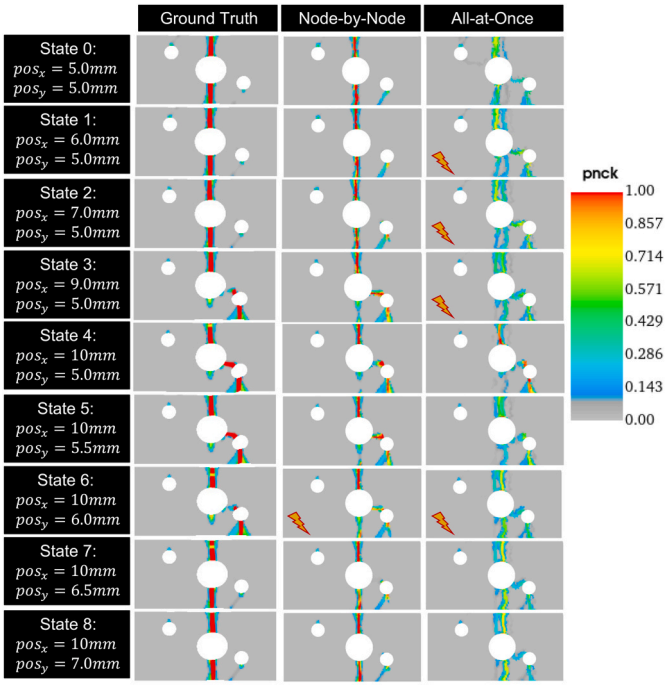


Fig. 16. Validation of a predicted response path (state 0–8): "Ground Truth" represents the FE simulations at the chosen feature values; *node-by-node* and *all-at-once* represent the predictions of the corresponding approaches.

The temporal *PNCK* response of the evaluation point *P* is compared to the response of the closest finite element for DOE design #8 in Fig. 17b. According to Table 2, the corresponding feature values vector is [5.0 mm, 1.75 mm]. This point is also highlighted in Fig. 17a by a black dot. It is again noted that the chosen evaluation point represents a point close to the crack and not a point in the center of the crack. This allow to obtain more non-zero data points in Fig. 17b for the comparison, since the *PNCK* response inside a crack is very steep and can step from zero to unity in one time step. The predicted time series is in excellent agreement with the FEM response.

Prediction of new yet unseen fracture modes: With the trained *node-by-node* model at hand, it is intriguing to check the feature parameter space for yet unseen crack paths. Manual examination reveals a new crack pattern at the feature values $pos_x = 2.75$ mm and $pos_y = -0.5$ mm, Fig. 18a. Accordingly, another FE simulation has been performed, using the identified parameter set, Fig. 18b. The crack pattern proposed by the *node-by-node* approach is in very good agreement with the subsequently performed FE simulation, indicating that the proposed parametric solution model has learned a generalized representation of the parametrized TMH load case.

4.2. Comparison of model evaluation time

It has to be noted that the evaluation time strongly depends on the underlying GPU hardware. Hence, the reported values should only be compared in a relative manner. The present analysis has been carried out on a NVIDIA® Quadro RTX™ 5000 graphic card.

The overall model evaluation time is the sum of the evaluation times for the initial coordinates of the nodes, the nodal displacements over time, and optionally additionally the evaluated nodal function, e.g. the post-necking indicator *PNCK*. For comparison, the models for the different outputs are evaluated several times and the recorded evaluation time is averaged accordingly, Table 4.

It is noted that the *node-by-node* and the *all-at-once* approaches use the same underlying model for prediction of the initial coordinates, denoted *COORD* in Table 4. For the *node-by-node* and *all-at-once* approaches, the evaluation time for *DISP* and *PNCK* are similar, where the prediction of the *DISP* output takes slightly longer than the prediction of *PNCK*. This is due to the fact that the output vector for DISP is 3 times larger than the *PNCK* output vector (*xyz* values versus single value per node).

It is again emphasized that each node is predicted independently by the *node-by-node* model. For small-scale FE models such as the investigated TMH example, the corresponding input vectors can be input to the network as a single batch and efficiently processed in parallel. For large-scale FE structures, the input vectors might have to be presented as a set of batches in order to fit into the available GPU memory and processed sequentially. Depending on the available hardware, this might increase the model evaluation time significantly.

4.3. Comparison of model training time

Compared to the costly evaluation time of the finite element simulations for obtaining the DOE results, the time for training of the parametric solution models is low. However, the potential need for fine-tuning network architecture parameters and other hyper-parameters might increase the overall training time significantly.

The training times (including validation sets) and the final training times (excluding validation sets) for the *node-by-node* and *all-at-once* approaches are summarized in Table 5. Again, it is emphasized that the

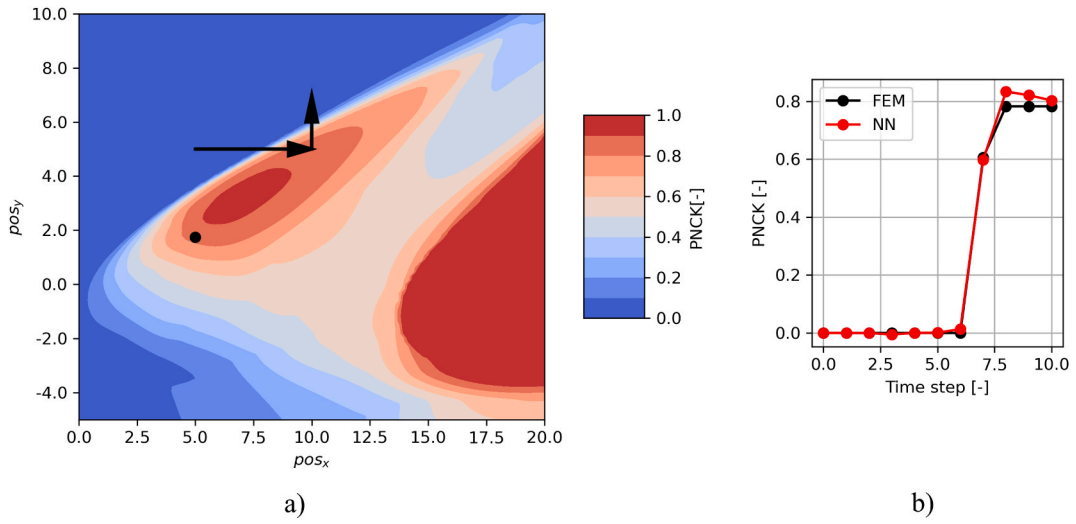


Fig. 17. a) Predicted maximum *PNCK* values over time of evaluation point *P* (close to a potential vertical crack occurring under the small moving hole, see Fig. 1) as a function of the input feature values. The black line represents the validation path discussed in Fig. 16; b) Time series prediction (curve NN) of the evaluation point *P* using the feature vector of DOE design #8 in comparison with the FEM simulation curve.

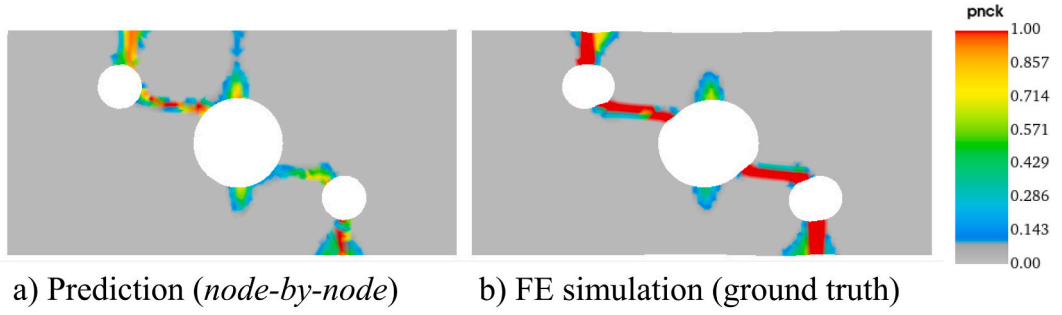


Fig. 18. a) New crack path suggested by the *node-by-node* approach at the feature values of $pos_x = 2.75$ mm and $pos_y = -0.5$ mm; b) Corresponding FE simulation response.

Table 4
Averaged model evaluation time of the *node-by-node* and *all-at-once* approaches.

Output	Approach	Evaluation time [s]	Remarks
COORD	all	0.0315	Averaged over 40 predictions
DISP	node-by-node	0.0400	Averaged over 20 predictions
DISP	all-at-once	0.0450	Averaged over 20 predictions
PNCK	node-by-node	0.0370	Averaged over 20 predictions
PNCK	all-at-once	0.0355	Averaged over 20 predictions

training times strongly depend on the underlying GPU hardware and that the reported values should only be compared in a relative manner.

A significant difference of training time with and without validation set is observed for the *all-at-once* approach, which is not present for the *node-by-node* approach.

Per epoch, the training of the *node-by-node* model takes longer than for the *all-at-once* model. However, for training of the *DISP* output the number of required epochs increases by one magnitude for the *all-at-once* compared to the *node-by-node* approach, eventually leading to a longer training time for the *all-at-once* model, when a validation set is included, Table 5.

For the *EPMX* and *PNCK* output, the difference of required epochs is less, where the training time for the *node-by-node* approach for reaching a converged loss state is significantly higher for the *PNCK* output.

All performed training runs for the *node-by-node* approach including a validation data set generalize well. That is, an additional final training using all available data is not necessarily required for the *node-by-node* approach. If the final training is omitted for the *node-by-node* approach, the training times are similar to the *all-at-once* approach (compare values with identical symbols in Table 5).

5. Conclusions

A parametrized finite element model of an academic load case, referred to as “two-moving-holes” (TMH) model, provided the data foundation of the presented analysis. Depending on the input parameter state, very different deformation and fracture patterns are observed in the underlying DOE using FE simulations. The investigated structural responses included various field quantities: the displacement (*DISP*), the equivalent plastic strains (*EPMX*) and the post-necking indicator (*PNCK*), representing the occurrence of cracks. The inhomogeneous FE structure as provided by the DOE is transformed to a unified data set using up-sampling and node re-ordering using the coherent point drift method. The parametric solution model for prediction of the initial coordinates was then created using a simple feed-forward neural network. Two different modeling approaches for predicting the parametrized time-dependent structural response of the TMH load case on top of the initial coordinates prediction have been investigated, referred to by the *node-by-node* and the *all-at-once* approach.

Both surrogate model approaches could well predict the responses for the *DISP* and *EPMX* fields, where the predictions for the *node-by-node* model turned out to be slightly more accurate. An accurate and generalized prediction of the strongly non-linear and narrow *PNCK* fields representing the crack patterns could only be achieved using the *node-*

Table 5
Averaged model evaluation time and R^2 values of the *node-by-node* and *all-at-once* approaches. The meaning of the value pairs identified by the symbols is explained in the text.

Output	Approach	Epochs [−]	Time w/ vali [s]	Time final [s]	Time sum [s]	R^2 [-]
DISP	Node-by-Node	2500	438 *	428	866	0.9993
	All-at-Once	30000	692	300	992 *	0.9978
EPMX	Node-by-Node	3000	516 □	501	1017	0.9942
	All-at-Once	10000	233	102	335 □	0.9769
PNCK	Node-by-Node	3000	531 ⊙	529	1060	0.9948
	All-at-Once	10000	233	102	335 ⊙	0.9268

by-node approach. The *all-at-once* approach did not provide satisfying predictions of *PNCK*. Investigation of the input feature parameter space using the *node-by-node* approach revealed the existence of a new crack pattern, which has not been observed in the underlying available DOE data. The new crack pattern could be confirmed by running an FE simulation using the identified feature parameter values, supporting the generalization of the trained *node-by-node* model.

In future work, the surrogate model, initially trained on an expected minimum of designs over the feature space, could be used for enriching the DOE data base iteratively. Recently [35], proposed adaptive surrogates for crash simulation using concepts of Uncertainty Quantification.

The model evaluation time of the *node-by-node* and the *all-at-once* approaches are similar for the investigated academic two-moving-holes example.

Considering the training time, the training time per epoch of the *node-by-node* model is longer than for the *all-at-once* model. However, the required number of epochs for the *node-by-node* approach is lower than for the *all-at-once* approach. Furthermore, the training of the *node-by-node* model including a validation set provides a good generalization, indicated by a similar loss evolution of the training and validation data sets. Hence, a subsequent training of the *node-by-node* model without keeping a validation set aside could potentially be skipped. In this case, the training times for both approaches become quite similar.

Limitations of the *all-at-once* approach: The initial coordinates, which are used as additional local input features for the *node-by-node* approach, provide crucial additional information for network generalization. These local features cannot be used for the presented global *all-at-once* approach, which can only establish global relations (interpolations) between the snapshots (designs) it has seen during the training. In contrast to the *all-at-once* approach, the *node-by-node* approach allows independent (uncoupled) learning of the time-series of individual nodes, depending on their initial coordinates.

Credit author statement

Lars Greve: Conceptualization, Methodology, Formal analysis, Software, Validation, Investigation, Visualization, Writing. Bram van de Weg: Software, Visualization, Writing- Reviewing and Editing.

Declaration of competing interest

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

A.1. Coherent point drift method (CPD)

Coherent point drift (CPD) [21] is an elastic point set registration method. It is used to transform a source point cloud to a similar, but not identical, target point cloud, while maintaining the underlying topological structure. A complete description is provided in Ref. [21]]. Below, the basic CPD algorithm is briefly revisited.

The moving point set (source) $\mathbf{Y}_{M\times D} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M)^{\mathrm{T}}$ with the nodal coordinates $\boldsymbol{y}_m$ represents the centroids of a GMM with M components and D dimensions. The fixed point set (target) $\mathbf{X}_{N\times D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)^{\mathrm{T}}$ with the nodal coordinates $\boldsymbol{x}_n$ is considered a set of N observations originating from a linearly combined Uniform + GMM model, where the uniform distribution $1/N$ is weighted by w and the GMM distribution is weighted by (1- w).

For the present work, the number of data points of source and target point sets are identical, $M = N$, and three-dimensional (*xyz)* nodal coordinates are used, $D=3$. The total mixture model reads:

$$p(\boldsymbol{x}) = w\frac{1}{N} + (1-w)\sum_{m=1}^{M} \Phi_m \mathcal{N}(\boldsymbol{x}, \boldsymbol{y}_m, \sigma), \tag{13}$$

where the means of the Gaussian distributions equal the centroids of the moving set,

$$\boldsymbol{\mu}_m = \boldsymbol{y}_m, \tag{14}$$

a common isotropic covariance is assumed,

$$\sigma_1^2 = \ldots = \sigma_M^2 = \sigma^2, \tag{15}$$

and equal mixing coefficients are assumed for the GMM:

$$\Phi_1 = \ldots = \Phi_M = 1/M. \tag{16}$$

For elastic CPD, the GMM centroids $\mathbf{y}_m$ are re-parametrized by elastic transformation $\mathcal{T}$:

$$\mathbf{y}_m = \mathcal{T}(\boldsymbol{Y}, \boldsymbol{W}) = \boldsymbol{Y} + \boldsymbol{G}\boldsymbol{W}, \tag{17}$$

with the matrix of coefficients $\boldsymbol{W}_{M\,\mathrm{x}\,D}$ and square and symmetric kernel matrix $\boldsymbol{G}_{M\,\mathrm{x}\,M}$. The iterative elastic point set registration algorithm is based on Expectation-Maximization (EM). The process steps are provided below:

Algorithm 1. Elastic coherent point drift

1. **Initialization** : $\boldsymbol{W} = 0,\ \sigma^2 = \frac{1}{DNM}\sum_{m,n=1}^{M,N}\|\boldsymbol{x}_n - \boldsymbol{y}_m\|^2$.

2. **Define parameters** : $0 \le w \le 1,\ \beta > 0,\ \alpha > 0$.

3. **Construct kernel matrix** $\boldsymbol{G}$: $G_{ij} = exp\Big(-\frac{1}{2\beta^2}\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2\Big)$

4. **EM** : loop until convergence of σ^2 is reached

   a) **Expectation** : calculate the elements of responsibility matrix $\boldsymbol{P}$.

$$P_{mn} = \frac{\exp(-\|\boldsymbol{x}_n - (\boldsymbol{y}_m + \boldsymbol{G}(m,\cdot)\boldsymbol{W}\|^2/(2\sigma^2))}{\sum_{k=1}^{M}\exp(-\|\boldsymbol{x}_n - (\boldsymbol{y}_k + \boldsymbol{G}(k,.)\boldsymbol{W}\|^2/(2\sigma^2)) + \frac{w}{1-w}(2\pi\sigma^2)^{D/2}\frac{M}{N}}$$

   b) **Maximization** : Solve for $\mathbf{W}$ and σ^2.

   Solve for $\boldsymbol{W}$: $(\boldsymbol{G} + \alpha\sigma^2 \mathrm{d}(\boldsymbol{P}1)^{-1})\boldsymbol{W} = \mathrm{d}(\boldsymbol{P}1)^{-1}\boldsymbol{P}\boldsymbol{X} - \boldsymbol{Y}$

   $N_p = 1^{\mathrm{T}}\boldsymbol{P}1,\ \boldsymbol{T} = \boldsymbol{Y} + \boldsymbol{G}\boldsymbol{W}$

   $\sigma^2 = \frac{1}{N_p D}[\mathrm{tr}(\boldsymbol{X}^{\mathrm{T}}\mathrm{d}(\boldsymbol{P}^{\mathrm{T}}1)\boldsymbol{X}) - 2\mathrm{tr}((\boldsymbol{P}\boldsymbol{X})^{\mathrm{T}}\boldsymbol{T}) + \mathrm{tr}(\boldsymbol{T}^{\mathrm{T}}\mathrm{d}(\boldsymbol{P}1)\boldsymbol{T})]$

5. **Result** : The aligned point set is $\mathcal{T}(\boldsymbol{Y}, \boldsymbol{W}) = \boldsymbol{Y} + \boldsymbol{G}\boldsymbol{W}$.

In Algorithm 1, the free model parameters are the ratio of uniform/GMM mixing, w, the variance of the Gaussian kernel matrix, β, and the smoothness regularization parameter α.

$\boldsymbol{G}(m,\cdot)$ refers to the m-th row of kernel matrix $\boldsymbol{G}$, $\mathrm{d}(\cdot)^{-1}$ is the inverse diagonal matrix, and 1 is a unit column vector.

References

[1] ESI Group. Virtual performance solution. 2021. Available online at: https://www.esi-group.com/.

[2] Bock Hans-Georg, Hoog Frank de, Friedman Avner, Gupta Arvind, Neunzert Helmut, Pulleyblank William R, et al. Model order reduction: theory, research aspects and applications. Berlin, Heidelberg: Springer Berlin Heidelberg; 2008. 13.

[3] Braconnier T, Ferrier M, Jouhaud J-C, Montagnac M, Sagaut P. Towards an adaptive POD/SVD surrogate model for aeronautic design. Comput Fluid 2011;40(1):195–209. https://doi.org/10.1016/j.compfluid.2010.09.002.

[4] Mohammed Auwalu I, Bartlett Mark, Oyeneyin Babs, Kayvantash Kambiz, Njuguna James. An application of FEA and machine learning for the prediction and optimisation of casing buckling and deformation responses in shale gas wells in an in-situ operation. J Nat Gas Sci Eng 2021;95:104221. https://doi.org/10.1016/j.jngse.2021.104221. 2000.

[5] Le Guennec Y, Brunet J-P, Daim F-Z, Chau M, Tourbier Y. A parametric and non-intrusive reduced order model of car crash simulation. Comput Methods Appl Mech Eng 2018;338:186–207. https://doi.org/10.1016/j.cma.2018.03.005.

[6] Hernandez Quercus, Badías Alberto, González David, Chinesta Francisco, Cueto Elías. Deep learning of thermodynamics-aware reduced-order models from data. Comput Methods Appl Mech Eng 2021;379(4):113763. https://doi.org/10.1016/j.cma.2021.113763.

[7] Hernández Quercus, Badías Alberto, González David, Chinesta Francisco, Cueto Elías. Structure-preserving neural networks. J Comput Phys 2021;426(12):109950. https://doi.org/10.1016/j.jcp.2020.109950.

[8] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9(8):1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.

[9] Cho, Kyunghyun; van Merrienboer, Bart; Gulcehre, Caglar; Bahdanau, Dzmitry; Bougares, Fethi; Schwenk, Holger; Bengio, Yoshua: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. Available online at: http://arxiv.org/pdf/1406.1078v3.

[10] Gorji Maysam B, Mozaffar Mojtaba, Heidenreich Julian N, Cao Jian, Mohr Dirk. On the potential of recurrent neural networks for modeling path dependent plasticity. J Mech Phys Solid 2020;143(7):103972. https://doi.org/10.1016/j.jmps.2020.103972.

[11] van de Weg BP, Greve L, Andres M, Eller TK, Rosic B. Neural network-based surrogate model for a bifurcating structural fracture response. Eng Fract Mech 2021;241(2):107424. https://doi.org/10.1016/j.engfracmech.2020.107424.

[12] Kohar Christopher P, Greve Lars, Eller Tom K, Connolly Daniel S, Inal Kaan. A machine learning framework for accelerating the design process using CAE simulations. An application to finite element analysis in structural crashworthiness. Comput Methods Appl Mech Eng 2021;385(4):114008. https://doi.org/10.1016/j.cma.2021.114008.

[13] Tancogne-Dejean Thomas, Gorji Maysam B, Zhu Juner, Mohr Dirk. Recurrent neural network modeling of the large deformation of lithium-ion battery cells. Int J Plast 2021;146:103072. https://doi.org/10.1016/j.ijplas.2021.103072.

[14] Im Sunyoung, Lee Jonggeon, Cho Maenghyo. Surrogate modeling of elasto-plastic problems via long short-term memory neural networks and proper orthogonal decomposition. Comput Methods Appl Mech Eng 2021;385(7):114030. https://doi.org/10.1016/j.cma.2021.114030.

[15] Bonatti Colin, Mohr Dirk. One for all: universal material model based on minimal state-space neural networks. Sci Adv 2021;7(26). https://doi.org/10.1126/sciadv.abf3658.

[16] van de Weg Bram, Greve Lars, Rosic Bojana. Long short-term relevance learning. 2021. Available online at: http://arxiv.org/pdf/2106.12694v1.

[17] Thomas Anoop Ebey, Guevelou Simon, Di Pasquale, Edmondo, Chambard Anne, Duval Jean-Louis, Chinesta Francisco, et al. Shape parametrization & morphing in sheet-metal forming. Procedia Manuf 2020;47:702–6. https://doi.org/10.1016/j.promfg.2020.04.216.

[18] Scarselli Franco, Gori Marco, Tsoi Ah Chung, Hagenbuchner Markus, Monfardini Gabriele. The graph neural network model. IEEE Trans Neural Network 2009;20(1):61–80. https://doi.org/10.1109/TNN.2008.2005605.

[19] Qi Charles R, Su Hao, Mo Kaichun, Guibas Leonidas J. PointNet: deep learning on point sets for 3D classification and segmentation. 2016. Available online at: http://arxiv.org/pdf/1612.00593v2.

[20] Wikipedia contributors: point set registration. 2021.

[21] Myronenko Andriy, Song Xubo. Point-set registration: coherent point drift. IEEE Trans Pattern Anal Mach Intell 2010;32(12):2262–75. https://doi.org/10.1109/TPAMI.2010.46.

[22] BETA CAE Systems. Ansa. 2021. Available online at: https://www.beta-cae.com/ansa.htm. [Accessed 9 September 2021]. checked on.

[23] Greve L, Eller TK, Medricky M, Andres M. Hardness-based plasticity and fracture model for quench-hardenable boron steel (22MnB5). In: AIP conference proceedings, 12/31/2013; 2013 (1567) : https://aip.scitation.org/doi/abs/10.1063/1.4850038. Available online at.

[24] Eller TK, Greve L, Andres MT, Medricky M, Hatscher A, Meinders VT, van den Boogaard AH. Plasticity and fracture modeling of quench-hardenable boron steel with tailored properties. J Mater Process Technol 2014;214(6):1211–27. https://doi.org/10.1016/j.jmatprotec.2013.12.015.

[25] Hill R. On discontinuous plastic states, with special reference to localized necking in thin sheets. J Mech Phys Solid 1952;1(1):19–30. https://doi.org/10.1016/0022-5096(52)90003-3.

[26] Stören S, Rice JR. Localized necking in thin sheets. J Mech Phys Solid 1975;23(6):421–41. https://doi.org/10.1016/0022-5096(75)90004-6.

[27] McKay MD, Beckman RJ, Conover WJ. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 1979;21(2):239. https://doi.org/10.2307/1268522.

[28] Pedregosa Fabian, Varoquaux Gaël, Gramfort Alexandre, Michel Vincent, Thirion Bertrand, Grisel Olivier, et al. Scikit-learn: machine learning in Python. J Mach Learn Res 2011;12(Oct):2825–30.

[29] Siavashk. Siavashk/Pycpd: pure numpy implementation of the coherent point drift algorithm. 2021. Available online at: https://github.com/siavashk/pycpd. [Accessed 1 May 2022]. checked on.

[30] Srivastava Nitish, Hinton Geoffrey, Krizhevsky Alex, Sutskever Ilya, Salakhutdinov Ruslan. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 2014;15(1):1929–58.

[31] Abadi Martín, Agarwal Ashish, Barham Paul, Brevdo Eugene, Chen Zhifeng, Citro Craig, et al. TensorFlow: large-scale machine learning on heterogeneous distributed systems. 2016. Available online at: http://arxiv.org/pdf/1603.04467v2. updated on 2016.

[32] Kingma Diederik P, Ba Jimmy. Adam: a method for stochastic optimization. 2014. Available online at: http://arxiv.org/pdf/1412.6980v9. updated on 2014.

[33] Pearson Karl. LIII. On lines and planes of closest fit to systems of points in space. Lond Edinburg Dublin Philosoph Magaz J Sci 2010;2(11):559–72. https://doi.org/10.1080/14786440109462720.

[34] Delaunay BN. Sur la sphère vide. Bull Acad Sci USSR 1934;1934(6):793–800.

[35] Rocas Marc, García-González Alberto, Larrayoz Xabier, Díez Pedro. Adaptive surrogates of crashworthiness models for multi-purpose engineering analyses accounting for uncertainty. 2021. Available online at: http://arxiv.org/pdf/2103.16202v1.