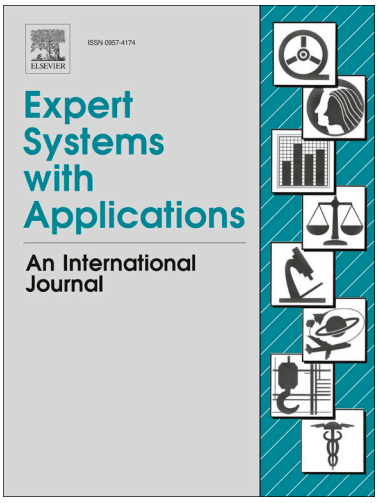
Journal Pre-proofs



DeepFEA: Deep learning for prediction of transient finite element analysis solutions

Georgios Triantafyllou, Panagiotis G. Kalozoumis, George Dimas, Dimitris K. Iakovidis

PII: S0957-4174(24)03210-X
DOI: https://doi.org/10.1016/j.eswa.2024.126343
Reference: ESWA 126343

To appear in: *Expert Systems with Applications*

Received Date: 15 April 2024
Revised Date: 19 December 2024
Accepted Date: 27 December 2024

Please cite this article as: Triantafyllou, G., Kalozoumis, P.G., Dimas, G., Iakovidis, D.K., DeepFEA: Deep learning for prediction of transient finite element analysis solutions, *Expert Systems with Applications* (2024), doi: https://doi.org/10.1016/j.eswa.2024.126343

DeepFEA: Deep Learning for Prediction of Transient Finite Element Analysis Solutions

Georgios Triantafyllou[a], Panagiotis G. Kalozoumis[a], George Dimas[a], and Dimitris K. Iakovidis[a,*]

[a]Dept. Computer Science and Biomedical Informatics, University of Thessaly, 2-4 Papasiopoulou st. 35131, Lamia, Greece

{gtriantafyllou, pkalozoumis, gdimas, diakovidis}@uth.gr

Abstract: Finite Element Analysis (FEA) is a computationally intensive method for simulating physical phenomena. Recent advancements in machine learning have introduced surrogate models that can accelerate FEA. However, there are limitations in developing surrogates of transient FEA models that can handle dynamic input and multi-dimensional output. Motivated by this research gap, this study proposes DeepFEA, a deep learning-based framework for predicting the solutions of transient FEA simulations. Key contributions of this study include i) a multilayer Convolutional Long Short-Term Memory (ConvLSTM) network branching in two parallel convolutional neural networks, enabling the prediction of both node- and element-related output (displacement, stress and strain), ii) a novel adaptive learning algorithm, called Node-Element Loss Optimization (NELO) tailored for minimizing accumulated error produced by recursive predictions of the network, and iii) publicly available reference datasets generated by 2D and 3D FEA model simulations in the context of structural mechanics. Key study outcomes regarding DeepFEA include i) improved performance over relevant state-of-the-art methods; ii) less than 3% normalized mean and root mean squared error; iii) inference times two orders of magnitude faster than traditional FEA; and iv) demonstration in a real-life biomedical application, which confirmed its effectiveness for accurate and efficient predictions of transient FEA simulations.

Keywords: Finite Element Analysis; Deep Learning; Convolutional Long Short-Term Memory Networks; Surrogate Models; Scheduled Sampling Method

1 Introduction

Physical phenomena can often be described by a system of equations, such as Partial Differential Equations (PDEs) or Ordinary Differential Equations (ODEs). PDEs are equations containing partial derivatives that encapsulate physical quantities of natural processes. The solution of these equations is provided by a function, the exact solution of which cannot be analytically calculated or is not unique (Evans, 2022). In research fields such as mechanical and biomedical engineering, the behavior of objects that are subject to a system of PDEs is simulated via computational modeling tools that are able to solve these PDEs in a computationally efficient way (Arzani et al., 2022; Calzolari & Liu, 2021; Samadian et al., 2024). Finite Element Analysis (FEA) is often used to divide such objects into a finite number of discrete elements and nodes, enabling dedicated FEA software to solve the PDEs in a more efficient manner (Ŝolín, 2005). FEA-based simulations rely on iteratively solving PDEs that correspond to specific physical models. Usually, the duration of the problem to be simulated needs to be discretized into a number of timesteps; a large number of timesteps results in a solution of a higher resolution in the time domain and it usually entails a higher computational cost. At each timestep, FEA relies on the predictions of the previous state to predict the solution for the current state (Hughes, 2012). These simulations can also become computationally expensive as the order of the PDEs and the density of the mesh increase. Therefore, the need to accelerate the process of estimating high-resolution FEA solutions by replacing conventional solver-based approaches serves as the initial motivation for the research scope of this work.

Reduced-order modelling (ROM) and surrogate models have been considered to address this problem. ROM solutions are mostly employed to reduce the dimensionality of high-order models (Masoumi-Verki et al., 2022), while surrogate models are used to replace FEA. Traditional ROM

* Corresponding author.

methods, *e.g.,* the proper orthogonal decomposition method and the principal component analysis method (Xiang et al., 2023), are constrained by their linear nature; thus, AI-based solutions, such as Artificial Neural Networks (ANNs) that include Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs), are preferred for reducing the dimensionality of non-linear physical models (Du et al., 2022; Liang et al., 2020). There has been an increasing interest in AI-based surrogate models that utilize ANNs (Phellan et al., 2021), which is due to their inherent ability to approximate linear and non-linear functions that are embedded in their weights (Hu et al., 2020). Based on their application, AI-based surrogate models can be divided into three main categories, *i.e.*, steady-state, physics-informed, and transient analysis models. ANN architectures, such as MLPs (Du et al., 2022) and CNNs (Pfeiffer et al., 2019) have been used as surrogate models for steady-state analyses. Steady-state simulations utilize only the input and initial conditions provided at a certain time-point without accounting for previous states or iteratively predicting the intermediate states that are necessary in traditional FEA (Cook & others, 2007). Despite their potential for steady-state predictions, these methods lack temporal awareness; thus, they are not suitable for modeling scenarios where all the states of a simulation need to be predicted. Since the next state of the mesh (deformation) depends on the previous timesteps, *i.e.,* the node coordinates affect the predicted deformation in the next timestep, the error propagates from the previous to the current timestep, leading to large errors toward the end of the simulation (Hughes, 2012). Long Short-Term Memory (LSTM) (Q. Chen et al., 2021), Gated Recurrent Unit (GRU) (Wu et al., 2020), Graph Neural Networks (GNNs) (Maurizi et al., 2022), Non-Linear Autoregressive Exogenous Models (NARXs) (X. Zhao et al., 2021), CNN-LSTMs (Totounferoush et al., 2021; Wang et al., 2021), and Convolutional LSTMs (Yan et al., 2023) have been used as models for transient analysis. These methods mainly consider the prediction of one or more states based on previous timesteps. However, recent methods require input from FEA during inference, they are constrained to 2D domains, or they cannot simultaneously predict outputs related to both the nodes and elements of the FEA model (Maurizi et al., 2022; Totounferoush et al., 2021; Wang et al., 2021). Physics-informed methods, such as Physics-Informed Neural Networks (PINNs) (Raissi et al., 2019), have also been used to predict the solutions of FEA-based simulations. Nonetheless, most of these methods can only predict the behavior of physical processes described by specific PDEs (considered during their training phase), tailored to the problem under investigation.

Although various surrogate models have been proposed to predict FEA solutions, current models have several limitations: they are often limited to steady-state predictions; they cannot completely replace FEA; they are incapable of simultaneously predicting outputs for both nodes and elements; they can only be trained for one simulation scenario at a time; they are PDE-specific; they are limited to the 2D domain; they rely on an initial amount of ground truth data. Furthermore, recurrent models, which rely on previous predictions to provide future solution estimations, usually result in error propagation and accumulation over time (Franke & Wagner, 2024). To address these limitations, this study introduces DeepFEA, a novel deep learning framework designed to replace FEA by leveraging a CNN-based architecture with dual output branches for multi-dimensional output predictions and an adaptive optimization algorithm for dynamic input predictions. In contrast, relevant state-of-the-art methods can either handle only dynamic input in 2D simulations by combining CNNs with an appropriate optimization algorithm such as in (Wang et al., 2021) or only multi-dimensional output using temporal GNNs without dynamic input capabilities such as in (Maurizi et al., 2022).

More specifically, the innovative contributions of this study include:

- A novel deep ANN architecture, called Node-Element Prediction (NEP) network, that is composed of a multilayer ConvLSTM and two parallel CNN branches. Unlike relevant state-of-the-art architectures, it considers as input the coordinates of the mesh nodes, the initial and boundary conditions, and the external load, to dynamically predict the solutions of 2D and 3D transient FEA simulations. The predictions for the nodes and the elements of the FEA model are made separately in the form of tensors.
- A novel adaptive learning algorithm, called Node-Element Loss Optimization (NELO), inspired from a technique used in the context of Natural Language Processing (NLP), called Scheduled Sampling Method (SSM) (Bengio et al., 2015). The SSM algorithm has been

adapted by utilizing a novel loss function co-considering the node- and element-level errors to enable a progressive incorporation of the previous predictions to each subsequent timestep, and the minimization of the accumulated error over several timesteps.
- A total of three, publicly available, reference datasets generated by 2D and 3D FEA model simulations in the context of structural mechanics, enabling comparisons with surrogate models.
- An experimental study that uses these datasets to assess the proposed framework both quantitatively and qualitatively.
- To the best of our knowledge, this is the first time that a deep learning framework capable of both multi-dimensional output and dynamic input predictions is proposed as a surrogate model for transient FEA simulations.

The remainder of this paper is organized as follows: Section 2 provides a literature review of the state-of-the-art surrogate models for FEA simulation; Section 3 describes the proposed framework; Section 4 presents its evaluation process for different 2D and 3D cases; Section 5 discusses the results and the key findings, and Section 6 summarizes the conclusions of this study and future work.

2 Related work

Early MLP-based steady-state prediction approaches were used to predict parts of the solution assisted by FEA (Bender et al., 2019; Burghardt et al., 2021). In later studies, FEA was completely bypassed and MLPs were utilized to incorporate the FEA-related parameters in the input feature vectors (Du et al., 2022; Farajtabar et al., 2022; Liang et al., 2020; Pellicer-Valero et al., 2020; Ti et al., 2021). Although this approach can be simple to implement, it can become computationally expensive for large meshes. A drawback of MLPs is that they lack spatial awareness, in the sense that they do not embed information related to the spatial location from which the features are extracted. Studies have attempted to address this problem by incorporating spatial information in the input vector, *e.g.*, geometry-related parameters (Farajtabar et al., 2022; Liu et al., 2022; Pellicer-Valero et al., 2020). CNNs inherently offer spatial awareness and they have been applied for predicting the outcome of FEA simulations, *e.g.*, deformation, stress, and strain (Ibragimova et al., 2022; Mendizabal et al., 2020; Pant et al., 2021; Pfeiffer et al., 2019). CNNs can scale better when considering larger meshes that require deeper networks. Bayesian CNNs have also been explored as surrogate models, providing adaptability in cases where a FEA simulation involves external loads (Deshpande et al., 2022; Krokos et al., 2022). Notably, these methods utilize CNN architectures that are similar to U-Nets. Other CNN-based applications include multi-grid CNNs (Le & Ooi, 2021) and image coloring techniques for 2D microstructure-based FEA simulations, where the predictions are based on the pixel intensity of the microstructure in grayscale (Khanolkar et al., 2021). CNNs are suitable for structured meshes, which limit their capabilities in scenarios with high geometric complexity. To address this problem, the use of zero-padding has been proposed, where zero values are added to form an evenly structured tensor (Deshpande et al., 2022; Krokos et al., 2022; Mendizabal et al., 2020).

Physics-informed methods have also been explored in the context of approximating FEA simulations (Bolandi et al., 2023; H. Chen et al., 2023; W. Chen et al., 2021; Haghighat et al., 2021; Haubner et al., 2024). The most popular category are PINNs, where the input contains spatial variables, *i.e.*, coordinates of the mesh nodes, and a time variable (Raissi et al., 2019). In relation to the examined problem, PINNs have been used to predict the dynamic stress of FEA models under various loading conditions with static input (Bolandi et al., 2023). Implicit neural representations (INRs) have also been utilized in a similar way as PINNs, where the boundary conditions are included in the loss function by incorporating spatiotemporal information based on the gradients of previous timesteps in the loss function (H. Chen et al., 2023). These methods have recently gained popularity due to their ability to incorporate PDEs in the loss function by utilizing the automatic differentiation process of ANNs

achieved via backpropagation. These approaches are excellent choices for interpolation, since they can learn to generate results over the whole domain of the experiment based only on a few training data.

In the context of transient analysis models, MLPs with the assistance of NARXs have been explored (X. Zhao et al., 2021; Y. Zhao et al., 2021). Nonetheless, they require a substantial number of timesteps to initialize the NARX model and can only be trained to predict the solutions of one simulation scenario at a time. Recurrent neural networks (RNNs) embed the temporal information of time-series in their weights, as well as in dedicated variables that contain information about the output of the previous timesteps. State-of-the-art methods that utilize RNNs, such as LSTM or GRU, have been used as surrogate models for PDE-based simulations (G. Chen, 2021), but they can partially predict parts of the solution based on certain calculated output parameters provided by FEA (Q. Chen et al., 2021; Ghavamian & Simone, 2019; Wu et al., 2020). It should be noted that, in such simulations, the input of the current timestep is not affected by the output of the previous timestep and there is no need for a spatially aware methodology. Therefore, the expected output can be easily predicted based only on the time-coupling that RNNs provide. More recently, MLP-LSTM hybrid networks, also known as Deep Operator Networks, have been used to predict the solutions of FEA simulations based on time-dependent loads (He et al., 2024).

Transient FEA simulations depend on spatiotemporal correlations between input conditions and predicted output parameters, thus the combination of CNNs with RNNs can be beneficial. The fusion of CNNs and LSTMs (CNN-LSTMs) has been adopted for such tasks (Arcones et al., 2022; Chijioke et al., 2022; Totounferoush et al., 2021; Wang et al., 2021). Nevertheless, these methods are limited to 2D scenarios, where only one output parameter of the solution is predicted. CNN-LSTMs have also been applied in Fluid-Structure Interaction (FSI) simulations, where the output of the structural solution is used as input for the fluid solution and *vice versa* (Arcones et al., 2022; Chijioke et al., 2022; Totounferoush et al., 2021). Graph Neural Networks (GNNs) have been recently used to predict FEA simulation solutions, since they have a similar spatial awareness capability to CNNs. GNNs can be combined with Auto Encoders (AEs), CNNs, GRUs, and LSTMs to create spatiotemporal-aware GNNs (Maurizi et al., 2022; L. Zhao et al., 2019). Despite their benefits, these methods do not consider the dynamic nature of the input parameters, and they are limited to inferring solutions only for a few timesteps. In addition, GNNs are limited in terms of scalability and can become computationally expensive for graphs with dynamic characteristics (Sharma et al., 2023), which is a key feature of transient FEA simulations. Convolutional LSTMs (ConvLSTMs) are another type of CNN and LSTM fusion, which have been used as surrogate models of FEA simulations (Shi et al., 2015). ConvLSTMs can extract spatiotemporal information from a time-series of images, such as videos, without requiring complex ANN or RNN architectures, thereby providing more scalability and flexibility. This flexibility can be particularly advantageous when dealing with FEA simulations, which often involve diverse mesh structures and varying levels of complexity (Wang et al., 2022). Methods that utilize ConvLSTMs have been applied to microstructure-based FEA simulations in the 2D domain (Frankel et al., 2020), material design (Yan et al., 2023), and surface heat-flux distribution (Wang et al., 2022). However, in the context of transient FEA simulations, they have only been used for predicting the evolution of output parameters based on an initial amount of ground truth data (Frankel et al., 2020; Yan et al., 2023). Variational AEs utilizing GRUs have also been used to predict time-dependent output parameters of 2D microstructure-based FEA simulations (Kim et al., 2023). More recently, a study aimed at addressing the problem of recursively predicting output parameters in FEA simulations proposed a hybrid model combining traditional ROM with LSTMs (Franke & Wagner, 2024). Nevertheless, this approach can only be applied to one node of the mesh and is limited to predicting node displacements. An overview of the state-of-the-art methods, highlighting the pros and cons of each category, is provided in Table 1.

A variety of methods related to the prediction of FEA simulation solutions has been proposed; yet, only a few studies have investigated surrogate models aimed at transient FEA simulations. Most of them have focused on addressing specific application scenarios, and only two of them were relevant to the problem under investigation. The choice was based on key features required, that include the capability of the method to predict both node- and element-related outputs for multiple timesteps, to cope with dynamic input and both 2D and 3D simulation scenarios (Table 2). It is evident that, although

several surrogate models have been proposed for predicting the output of transient FEA simulations, there is still a lack of methods that can completely replace FEA and have all the identified key features. DeepFEA is proposed as a solution to cover this gap and effectively address the limitations of the current methods.

Table 1 A summary of state-of-the-art surrogate models for FEA simulations.

Category	Model Type	Strengths	Limitations	Methods
Steady-state	MLPs	Simplified implementation	Computationally expensive for large meshes, lacks spatial awareness, error accumulation for dynamic input prediction	(Du et al., 2022; Farajtabar et al., 2022; Liang et al., 2020; Pellicer-Valero et al., 2020; Ti et al., 2021)
	CNNs	Spatial awareness, scale better for larger meshes	Error accumulation for dynamic input prediction	(Deshpande et al., 2022; Ibragimova et al., 2022; Krokos et al., 2022; Mendizabal et al., 2020; Pant et al., 2021; Pfeiffer et al., 2019)
Physics-informed	PINNs INRs	Boundary conditions are incorporated in the loss function, good for training with few data points	Can be trained on PDE-specific problems, cannot handle dynamic input predictions	(Bolandi et al., 2023; H. Chen et al., 2023; W. Chen et al., 2021; Haghighat et al., 2021; Haubner et al., 2024)
Transient analysis	NARXs	Capture temporal dependencies	Requires extensive ground truth data during inference, can be trained on one problem at a time	(X. Zhao et al., 2021; Y. Zhao et al., 2021)
	RNNS, LSTMs, GRUs	Capture temporal dependencies	No spatial awareness, error accumulation for dynamic input prediction, rely on ground truth data for initial predictions	(Q. Chen et al., 2021; Ghavamian & Simone, 2019; He et al., 2024; Wu et al., 2020)
	Temporal GNNs, CNN-LSTM ConvLSTM	Spatiotemporal awareness	Error accumulation for dynamic input predictions, rely on ground truth data for initial predictions, limited to 2D applications	(Arcones et al., 2022; Chijioke et al., 2022; Frankel et al., 2020; Kim et al., 2023; Maurizi et al., 2022; Totounferoush et al., 2021; Wang et al., 2021, 2022; Yan et al., 2023; L. Zhao et al., 2019)

Table 2 Key features of DeepFEA and relevant state-of-the-art methods.

Method	Node and element output predictions	Predictions across multiple timesteps	Tailored for dynamic input	2D simulations	3D simulations
StressNet (Wang et al., 2021)	×	✓	✓	✓	×
Temporal GNN (Maurizi et al., 2022)	×	×	×	✓	✓
DeepFEA	✓	✓	✓	✓	✓

3 Methods

DeepFEA is a deep learning framework designed to simultaneously predict node- and element-related output across multiple timesteps, given a set of initialization parameters of a FEA simulation. Its core is a novel, trainable network architecture (NEP), which can provide very accurate predictions after being trained with NELO, which is a bespoke training algorithm especially designed to minimize error accumulation in recursive predictions of transient FEA solutions.

In transient FEA, the behavior of the system at each timestep t is governed by the following dynamic equilibrium equation:

$$\mathbf{M}\ddot{\boldsymbol{U}}_{\boldsymbol{t}} + \mathbf{C}\dot{\boldsymbol{U}}_{\boldsymbol{t}} + \mathbf{K}\boldsymbol{U}_{\boldsymbol{t}} = \mathbf{F}_{\boldsymbol{t}} \quad (1)$$

where $\ddot{\boldsymbol{U}}_{\boldsymbol{t}}$, $\dot{\boldsymbol{U}}_{\boldsymbol{t}}$, and $\boldsymbol{U}_{\boldsymbol{t}}$ correspond to the acceleration, velocity, and displacement tensors at timestep t, $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{K}$ are the mass, damping, and stiffness matrices, representing inertial, damping, and elastic effects, respectively, and $\mathbf{F}_{\boldsymbol{t}}$ represents the applied forces at timestep t. The displacement tensor can also be expressed as the change in position of the nodes over time:

$$\boldsymbol{U}_{\boldsymbol{t}} = \boldsymbol{Q}' - \boldsymbol{Q} \quad (2)$$

where $\boldsymbol{Q}'$ and $\boldsymbol{Q}$ denote the nodal positions at the current and previous timesteps, respectively.

Once $\boldsymbol{U}_{\boldsymbol{t}}$ is computed, it is used to calculate strains and stresses through spatial derivatives of the displacement field (Wriggers, 2008). For instance, the strain can be computed as a gradient of the displacement field, while the stress is derived via material constitutive laws, *e.g.*, Hooke's law for linear elasticity. Since DeepFEA serves as a unifying surrogate for a complex system of equations in transient FEA, it can be described by:

$$\Phi(\boldsymbol{I_t}) = \boldsymbol{Y_t} \quad (3)$$

where Φ is the function learned by the NEP network, $\boldsymbol{I_t}$ is the input to the FEA model, which depends on $\mathbf{M, C, K, F_t}$, and $\boldsymbol{Q}$ (Eqs. (1) and (2)), and $\boldsymbol{Y_t}$ is the predicted output, defined in this study as follows:

$$\boldsymbol{Y_t} = (\boldsymbol{Q'}, \boldsymbol{\Sigma}, \boldsymbol{E}) \quad (4)$$

where $\boldsymbol{\Sigma}$ denotes the effective stress tensor and $\boldsymbol{E}$ the effective strain tensor. These are typically scalar quantities (equivalent von Mises stress and strain) that are widely used in engineering applications to assess material behavior under complex loading conditions.

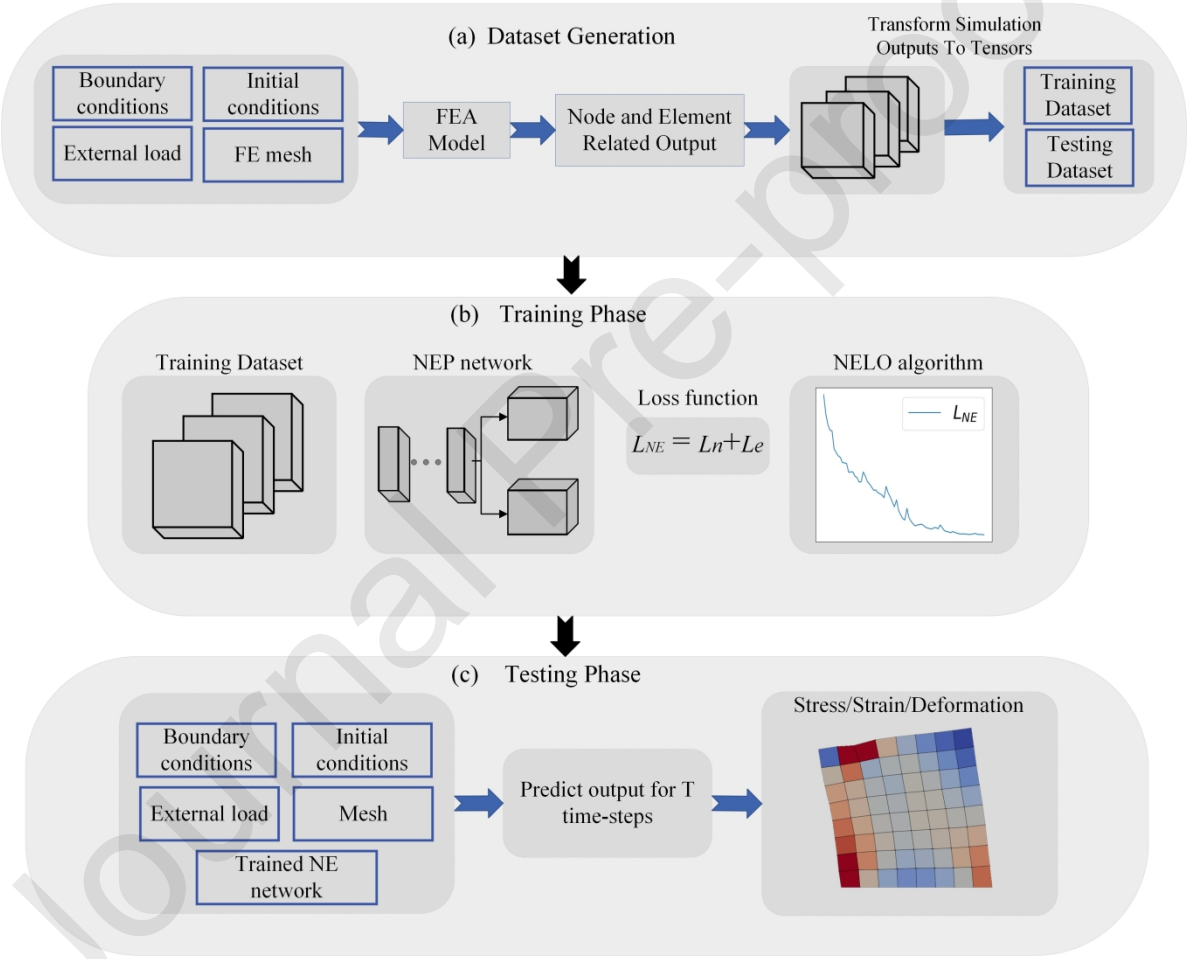


Figure 1. DeepFEA framework.

An overview of the proposed framework is presented in Fig. 1, comprising three phases: the dataset generation phase (Fig. 1a), training phase (Fig. 1b), and testing phase (Fig. 1c). The first phase of the DeepFEA framework (Fig. 1a) involves the generation of a transient FEA simulation dataset. Boundary and initial conditions, mesh structure (FE mesh), and varying maximum external loads are selected to create a dataset with representative transient FEA simulations that will enable NEP to learn the behavior of the examined FEA model. Then, a commercial FEA simulation software, such as ANSYS LS-DYNA, is used to calculate the solutions for each simulation and a pre-processing step maps the node- and element-related outputs of every timestep to tensors that will be used as input and output for the network. Subsequently, (Fig. 1b) a portion of the dataset is used to train the network

based on the NELO algorithm (Sub-section 3.2.1) which aims to minimize the loss function $\mathcal{L}_{NE}$ (Sub-section 2.2.2).

Once the network is trained (Fig. 1c), it can predict the solutions of simulations on the same mesh structure that have not been included in the construction of the training dataset. This approach allows for recurrent estimation of the simulation output in the time domain. In the following subsections the details of the architecture and of the training phase are described.

3.1 Node-Element-based Network Architecture

The NEP network architecture of DeepFEA consists of two modules, a Feature Extraction Module (FExM) and a Prediction Module (PM). FExM is implemented using a multilayer ConvLSTM network, which receives as input the FEA model characteristics and the initialization parameters for the FEA in the form of a tensor. FExM is tasked with automatically extracting spatiotemporal features in the form of multiple feature maps extracted over consecutive timesteps.

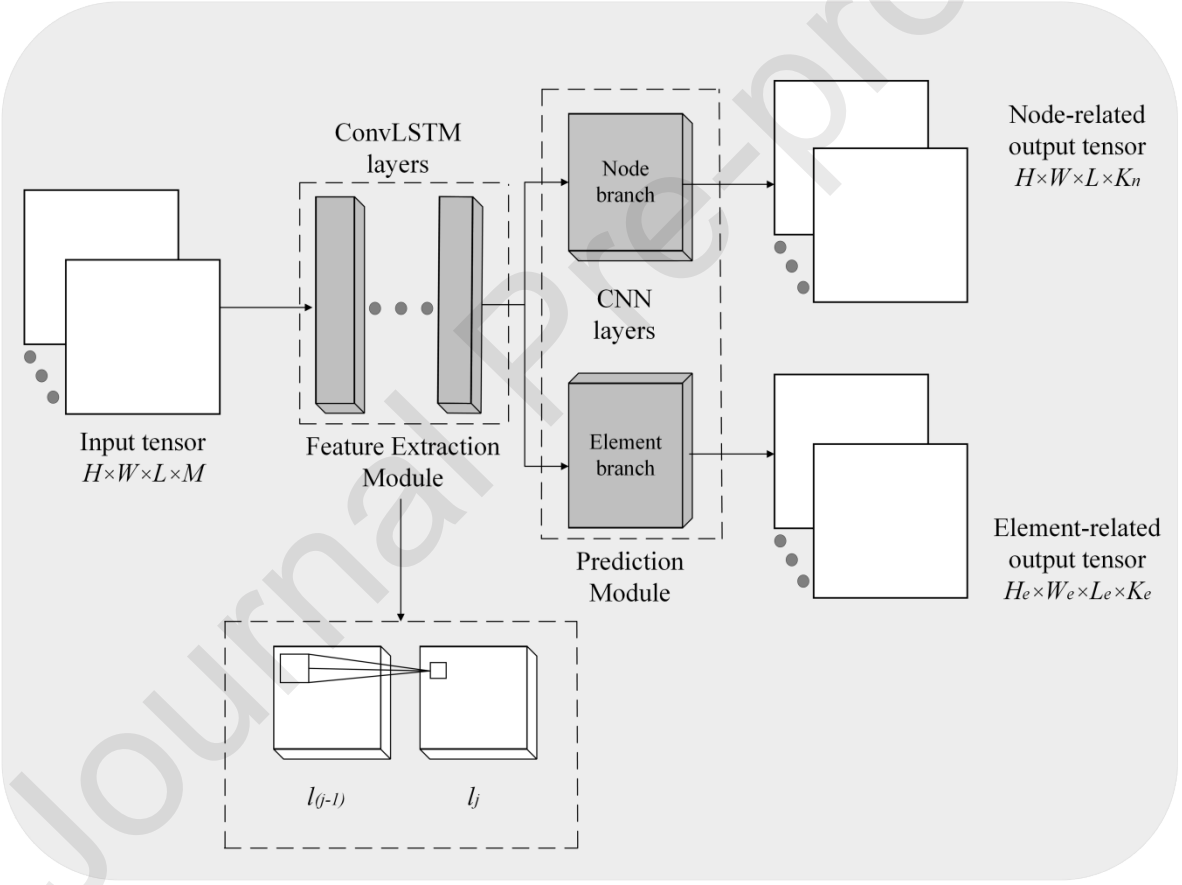


Figure 2. The DeepFEA NEP network architecture.

More specifically, the feature maps encode spatial information regarding the nodes and elements of the FEA model along with the respective physical properties considered in the FEA simulation. This information is propagated through each layer of the ConvLSTM network, allowing it to progressively extract complex spatial and physics-related information of the simulation in various degrees of abstraction. The output of the final ConvLSTM layer encapsulates the overall spatial structure of the mesh, the information retained from previous timesteps, as well as extracted information about the physics of the simulation, serving as a high-level representation of the FEA simulation. Given a set of extracted features by FExM, PM predicts the output parameters of the simulation. PM comprises two

parallel CNN branches, producing two output tensors that follow the structure of the input tensor. These output tensors have different dimensions corresponding to the node- and element-related parts of the FE mesh.

3.1.1 Input and Output Tensors

The geometry of the structure to be analyzed with FEA is first discretized into a digital representation, called FE mesh, which is composed of a finite number of nodes (discretized points) and elements (area or volume that connects the nodes of the mesh). Each node and element of the mesh stores information related to the FEA model, *e.g.*, coordinates or stress/strain in each node or element, respectively. The FEA model is created by specifying a mesh structure, the initial conditions, the boundary conditions, and the externally applied load. In this study, the coordinates of the mesh are considered as the initial conditions, the positions of the constrained nodes as the boundary conditions, and the forces applied on the outer nodes as the externally applied load. A FEA simulation is tasked with predicting the output parameters for each node and element of the mesh based on the initial conditions corresponding to each node, external load, and boundary conditions.

In Fig. 2, the input tensor size is $H\times W\times L$, where H, W, and L denote the height, width, and length of each input feature map, respectively, whereas M denotes the number of the different parameters considered for solving a specific problem. For a 3D mesh, $H\times W\times L$ correspond to the dimensions of the input mesh. The notation $H_e\times W_e\times L_e$ refers to the dimensions of a tensor that contains information regarding the elements of the mesh. In the 2D mesh case, only the height and the width of the mesh are considered, *i.e.*, H and W. The output tensor of the node branch is of shape $H\times W\times K_n$ or $H\times W\times L\times K_n$ for 2D and 3D models, respectively. In this paper, the 2D and 3D FEA models are meshed with quadrilateral and hexahedral elements, respectively. Hence, the output tensor of the element branch is of shape $H_e\times W_e\times K_e$ for 2D or $H_e\times W_e\times L_e\times K_e$ for 3D FEA models, with $H_e = H$-1, $W_e = W$-1, and $L_e = L$-1. Variables K_n and K_e denote the number of different output targets estimated by the dedicated node and element branches, respectively, *e.g.*, displacement across the x-axis, displacement across the y-axis, *etc.* For example, considering the input referring to the position of each node for a 2D FE mesh, *i.e.*, a pair of x and y coordinates, the information of the x and y coordinates is structured into two different matrices, as illustrated in Fig. 3. In contrast to traditional CNN-based frameworks that employ padding techniques to accommodate structural changes in the input tensor, DeepFEA utilizes the FE mesh coordinates as input features. This approach enables the prediction of FEA solutions whilst the structure of the mesh changes due to deformation.

首先，待分析的结构的几何形状被离散化为数字表示，称为 FE 网格，它由有限数量的节点（离散化的点）和单元（连接网格节点的面积或体积）组成。网格的每个节点和单元存储与 FEA 模型相关的信息，例如，每个节点或单元的坐标或应力/应变信息。FEA 模型是通过指定网格结构、初始条件、边界条件和外部施加的载荷来创建的。在本研究中，网格的坐标被视为初始条件，约束节点的位置作为边界条件，施加在外部节点上的力作为外部载荷。FEA 仿真旨在根据每个节点的初始条件、外部载荷和边界条件，预测网格中每个节点和单元的输出参数。

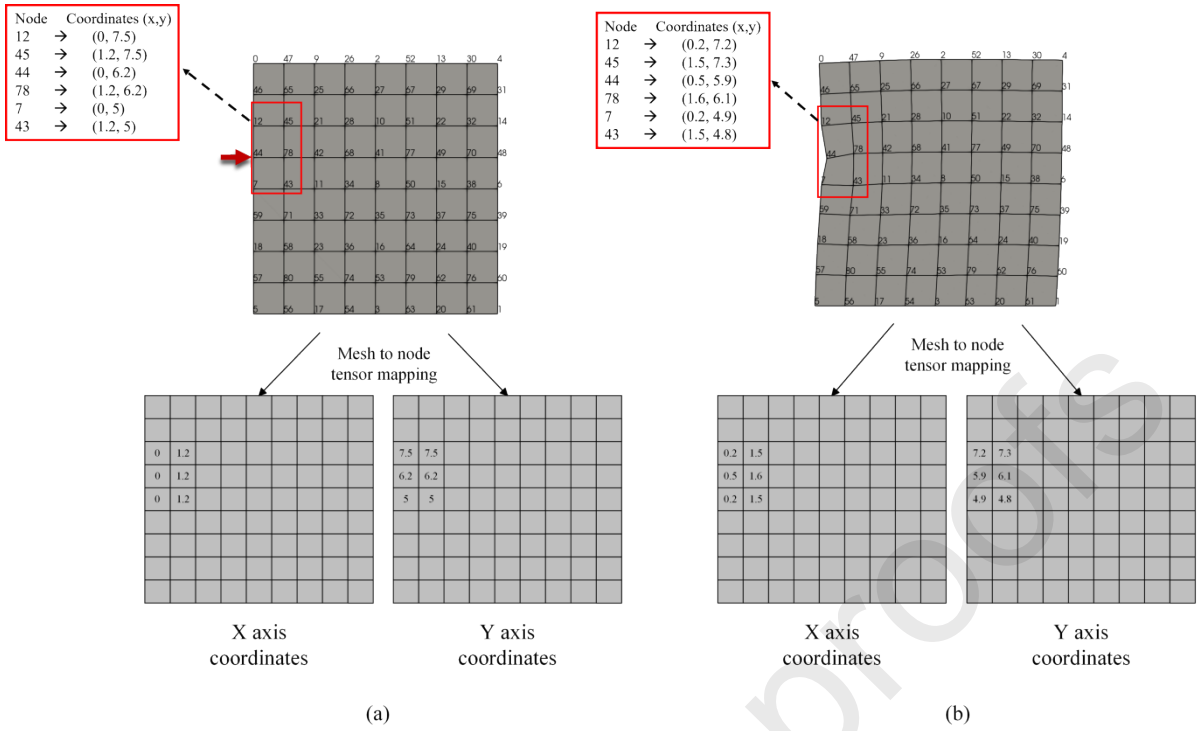


Figure 3. Mapping of a 2D FE mesh to input and output node tensors. (a) Before and (b) after the application of a load. The red arrow indicates the position and direction of the applied force.

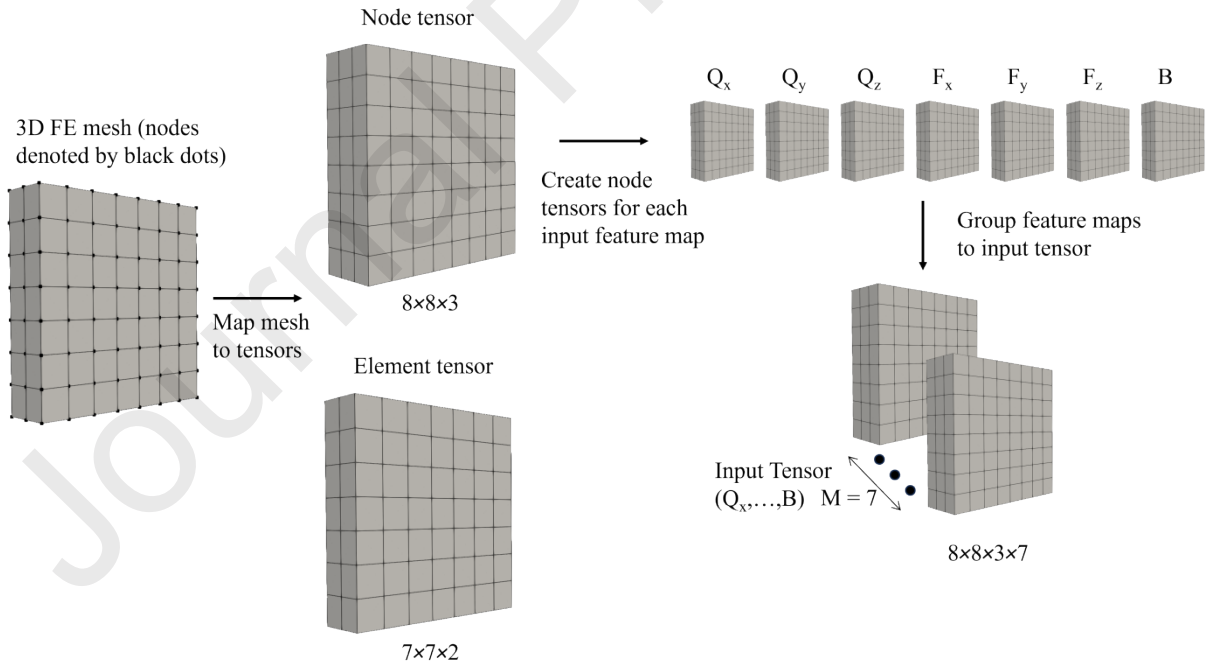


Figure 4. Mapping process of a 3D FE mesh into node and element tensors.

An example of how the input tensor of DeepFEA is formed for a 3D FEA model is presented in Fig. 4. The input tensor $\boldsymbol{X_t} = (\boldsymbol{Q_x}, \boldsymbol{Q_y}, \boldsymbol{Q_z}, \boldsymbol{F_x}, \boldsymbol{F_y}, \boldsymbol{F_z}, \boldsymbol{B})$ consists of 7 input feature maps, where ($\boldsymbol{Q_x}, \boldsymbol{Q_y}, \boldsymbol{Q_z}$) is a triplet of feature maps denoting the coordinates of the mesh, ($\boldsymbol{F_x}, \boldsymbol{F_y}, \boldsymbol{F_z}$) is a triplet of feature maps denoting the initial force vectors applied to the mesh, and $\boldsymbol{B}$ is a feature map denoting the

constrained nodes for the timestep *t*. In this case, the dimensions of an input tensor $\boldsymbol{X_t}$ are *8×8×3×7*, where *8×8×3* are the dimensions of the input mesh and *7* is the number of input features. Thus, each input feature map of $\boldsymbol{X_t}$ can be considered as a copy of the initial node tensor with shape *8×8×*3, containing feature-related information. Furthermore, the fixed nodes in the feature map $\boldsymbol{B}$ are denoted with 0, whereas non-constrained nodes are denoted with 1. This feature essentially works as a flag that minimizes the effect of the fixed nodes inside the feedforward process of the NEP network and enables it to differentiate between constrained and non-constrained nodes. Regarding the forces applied on the nodes of the mesh, at each timestep *t*, only the external load is considered as input. Hence, the ($\boldsymbol{F_x}$, $\boldsymbol{F_y}$, $\boldsymbol{F_z}$) channels contain information regarding only the forces applied externally during the simulation.

The output parameters considered in this study are the displaced coordinates of the mesh and the effective stress and strain for the node and element branches, respectively. Therefore, the output tensor of the node branch for a 3D FEA model can be defined as $\boldsymbol{Y}^{n}_{t+1} = (\boldsymbol{Q'_x}, \boldsymbol{Q'_y}, \boldsymbol{Q'_z})$, where ($\boldsymbol{Q'_x}$, $\boldsymbol{Q'_y}$, $\boldsymbol{Q'_z}$) corresponds to the new coordinates of the mesh. These coordinates are subsequently utilized as input in *t + 1* to predict the output parameters of the mesh for the next timestep, *i.e.*, for *t + 2*. Hence, the displaced coordinates are considered as Recurrent Parameters (RPs). In the example case presented in Fig. 4, the dimensions of the node branch output tensor would be *8×8×3×3,* where *8×8×3* are the dimensions of the output tensor mesh and *3* is the number of output features. As regards the element branch, the output tensor can be defined as $\boldsymbol{Y}^{e}_{t+1} = (\boldsymbol{\Sigma},\boldsymbol{E})$, where $\boldsymbol{\Sigma}$ corresponds to the effective stress tensor and $\boldsymbol{E}$ to the effective strain tensor. Moreover, the element branch output tensor would be of shape *7×7×2×2*, where *7×7×2* are the dimensions of the element tensor and 2 denotes the number of output parameters. Since $\boldsymbol{Y}^{e}_{t+1}$ is not used as input for the next timestep predictions, the effective stress and strain are considered as Non-Recurrent Parameters (NRPs).

3.1.2 Feature Extraction Module

The FExM utilizes *r* ConvLSTM layers, each responsible for extracting important spatiotemporal features of the FEA model that are identified by its kernels. Contrary to traditional CNN-based methods, the dimensionality of the input tensor is not reduced after each layer, thus the extracted feature map of each layer has the same dimensions as the input tensor with different number of features.

A ConvLSTM layer consists of three gates, *i.e.*, the input gate $\boldsymbol{i}_t$, the forget gate $\boldsymbol{f}_t$, and the output gate $\boldsymbol{o}_t$ (Fig. 5). These three gates are used to predict the cell state $\boldsymbol{C}_t$ and the hidden state $\boldsymbol{H}_t$ for the current timestep *t*. The input gate $\boldsymbol{i}_t$ is tasked with incorporating useful information from the input of the current timestep and the forget gate $\boldsymbol{f}_t$ is tasked with determining which piece of information should be retained from the memory cell status of the previous timestep. The output gate $\boldsymbol{o}_t$ determines whether the memory cell status of the current timestep will be propagated to the hidden state.

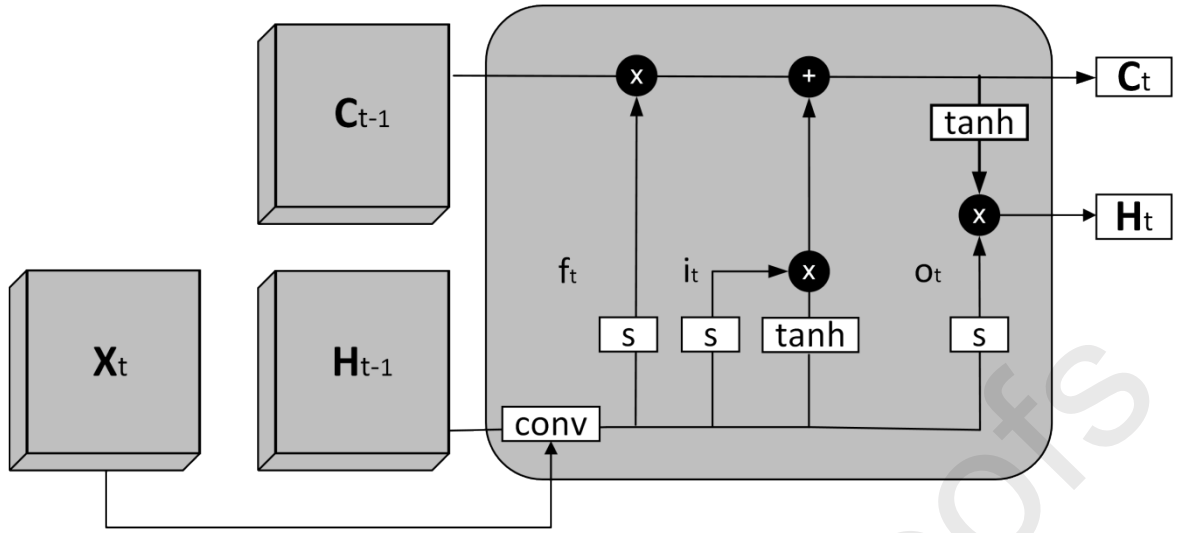


Figure 5 Core architecture of a ConvLSTM layer.

The cell of a ConvLSTM layer can be described by the following system of equations:

$$g(\cdot;\boldsymbol{W},\boldsymbol{b}) = \begin{cases} \boldsymbol{i}_t = s(\boldsymbol{W}_{xi} * \boldsymbol{X}_t + \boldsymbol{W}_{hi} * \boldsymbol{H}_{t-1} + \boldsymbol{W}_{ci} \circ \boldsymbol{C}_{t-1} + \boldsymbol{b}_i) \\ \boldsymbol{f}_t = s(\boldsymbol{W}_{xf} * \boldsymbol{X}_t + \boldsymbol{W}_{hf} * \boldsymbol{H}_{t-1} + \boldsymbol{W}_{cf} \circ \boldsymbol{C}_{t-1} + \boldsymbol{b}_f) \\ \boldsymbol{C}_t = \boldsymbol{f}_t \circ \boldsymbol{C}_{t-1} + \boldsymbol{i}_t \circ \tanh(\boldsymbol{W}_{xc} * \boldsymbol{X}_t + \boldsymbol{W}_{hc} * \boldsymbol{H}_{t-1} + \boldsymbol{b}_c) \\ \boldsymbol{o}_t = s(\boldsymbol{W}_{xo} * \boldsymbol{X}_t + \boldsymbol{W}_{ho} * \boldsymbol{H}_{t-1} + \boldsymbol{W}_{co} \circ \boldsymbol{C}_t + \boldsymbol{b}_o) \\ \boldsymbol{H}_t = \boldsymbol{o}_t \circ \tanh(\boldsymbol{C}_t) \end{cases} \quad (5)$$

where ' $*$ ' denotes the convolution operation, ' $\circ$ ' is the operator of the Hadamard product, $s(\cdot)$ denotes the sigmoid function, $\boldsymbol{W} = (\boldsymbol{W}_{xi}, \boldsymbol{W}_{hi}, \boldsymbol{W}_{xf}, \boldsymbol{W}_{hf}, \boldsymbol{W}_{xo}, \boldsymbol{W}_{ho}, \boldsymbol{W}_{xc}, \boldsymbol{W}_{hc})$ denotes the weight parametrization of $\boldsymbol{g}$, and $\boldsymbol{b}$ denotes its biases $(\boldsymbol{b}_i, \boldsymbol{b}_f, \boldsymbol{b}_o, \boldsymbol{b}_c)$. The tensors $(\boldsymbol{W}_{xi}, \boldsymbol{W}_{hi}, \boldsymbol{W}_{ci}, \boldsymbol{b}_i)$, $(\boldsymbol{W}_{xf}, \boldsymbol{W}_{hf}, \boldsymbol{W}_{cf}, \boldsymbol{b}_f)$, $(\boldsymbol{W}_{xo}, \boldsymbol{W}_{ho}, \boldsymbol{W}_{co}, \boldsymbol{b}_o)$, and $(\boldsymbol{W}_{xc}, \boldsymbol{W}_{hc}, \boldsymbol{b}_c)$ are the weight and bias tensors for the $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, and $\boldsymbol{o}_t$ gates and the cell state $\boldsymbol{C}_t$, respectively. Moreover, $\boldsymbol{X}_t$ is the input tensor for the current timestep, $\boldsymbol{H}_t$ and $\boldsymbol{H}_{t-1}$ are the hidden states of the current and previous timesteps, respectively, and $\boldsymbol{C}_t$ and $\boldsymbol{C}_{t-1}$ are the memory cell statuses of the current and previous timesteps, respectively.

Each ConvLSTM layer $\boldsymbol{l}^j$ of the FExM, utilizes the predicted hidden state $\boldsymbol{H}_t^{j-1}$ of the previous layer. Based on Eq. (5), a layer $\boldsymbol{l}^j$ can be formally expressed as:

$$\boldsymbol{l}^j = g\left(\boldsymbol{H}_t^{j-1}, \boldsymbol{H}_{t-1}^j, \boldsymbol{C}_{t-1}^j; \boldsymbol{W}^j, \boldsymbol{b}^j\right) \; for \; j = 2,\ldots,r \quad (6)$$

where $g(\cdot;\boldsymbol{W}^j,\boldsymbol{b}^j)$ is Eq. (5) expressed for layer $\boldsymbol{l}^j$ and r denotes the total number of ConvLSTM layers. In addition, $\boldsymbol{H}_{t-1}^j$ and $\boldsymbol{C}_{t-1}^j$ denote the hidden and cell states, respectively, for layer $\boldsymbol{l}^j$ and timestep t-1. For $j = 1$, Eq. (6) becomes:

$$\boldsymbol{l}^1 = g\left(\boldsymbol{X}_t, \boldsymbol{H}_{t-1}^1, \boldsymbol{C}_{t-1}^1; \boldsymbol{W}^1, \boldsymbol{b}^1\right) \quad (7)$$

Let $\boldsymbol{H}_t^r$ be the hidden state of timestep t as estimated by the last ConvLSTM layer $\boldsymbol{l}^r$ of the FExM. The hidden state $\boldsymbol{H}_t^r$ is then propagated through each branch of the PM to predict the node- and element-related output parameters. This procedure is conducted sequentially for each simulation timestep, starting from the initial condition at $t_o = 0$ until the end of the simulation for a total number of T timesteps.

3.1.3 Prediction Module

PM utilizes two CNN branches in a parallel topology. Each CNN branch comprises one or more CNN layers; in this study, one CNN layer is utilized for each branch. One branch is used to map the extracted feature maps to node tensors and the other is used to map the extracted feature maps to output tensors representing the elements of the FEA model. Furthermore, the CNN layers of each PM branch utilize the same kernel dimensions as the ConvLSTM layers of the FExM. Each branch is dedicated to predicting node- and element-related parameters. The distinction between the branches dedicated to node- and element-related predictions is mandatory, since they refer to outputs that are represented by tensors of different sizes. Therefore, for a 3D FEA model with hexahedral elements, the CNN layer of the node branch would utilize a $1\times1\times1\times K_n$ kernel, whereas that of the element branch would utilize a $2\times2\times2\times K_e$ kernel, with K_n and K_e denoting the number of output features for each branch, respectively.

The activation function of the CNN layers of each branch is determined by the data normalization process. For instance, if the input data are normalized within the interval of [-1, 1], then the hyperbolic tangent (*tanh*) can be considered as an appropriate activation function for each output layer of each branch. This is a requirement for DeepFEA, since the output of the NEP network at the timestep t is used as input for the next timestep t+1.

3.2 Training Phase

The NEP network is trained using data generated from conventional FEA simulations using commercial software, such as ANSYS LS-DYNA, or other in-house software, to simulate the behavior of a structure under different conditions, *e.g.*, applying forces with different magnitudes and angles on a structure of a given material. The training dataset contains representative FEA simulations of the examined model that will enable the NEP network to learn the examined behavior without being trained on the whole spectrum of possible FEA simulations. Subsequently, the trained NEP network can be used to predict the solutions for new FEA simulations of the examined FEA model with varying conditions, such as external load applied at different nodes of the mesh. Furthermore, the training phase relies on a NELO algorithm to address the error accumulation problem that occurs through the traditional training approach of deep learning models and a novel additive loss function to account for both node- and element-related errors produced by the NEP network.

3.2.1 Node-Element Loss Optimization Algorithm

The NELO algorithm is inspired by the SSM, which is an NLP technique used for the training of sequence-to-sequence models such as RNNs. These models generate an output text sequence (e.g., a translation or a text prediction) based on an input sequence (e.g., a source sentence). During training, the model uses the ground truth tokens (words or characters) at each step to predict the next token. However, during inference (generation of output sequences), the model uses its own predictions as inputs, which may lead to errors accumulating over time, similar to surrogate models of transient FEA simulations. Therefore, the NELO algorithm adopts an SSM optimization approach adapted to the domain of transient FEA simulations and tackles the error accumulation problem by gradually transitioning from using ground truth data to using model predictions as inputs during training. The NELO algorithm can be described as follows:

Algorithm 1 NELO algorithm

Set $P_s \longleftarrow 1$***,*** $k \longleftarrow$ *number of epochs to decrease* P_s*,* $S \longleftarrow$ *total epochs,* $T \longleftarrow$ *total timesteps,* $\kappa \longleftarrow 0$

for $j = 0$ *to* S *epochs*

    Procedure Train model

    for $b = 0$ ***to*** B *total batches*

        for $t = 0$ ***to*** T *total timesteps*

            $P_r \longleftarrow$ *random [0,1)*

            if $P_r > P_s$ ***and*** $t > 0$

                Then

                Replace ground truth input with predicted output

            Predict output for next timestep

        End

        Procedure Backpropagation, Optimizer step

    End

    if $j \,\%\, k == 0$

        Then

        $\kappa \longleftarrow \kappa + 1$

        $P_s \longleftarrow$ *Decrease by preferred scheme*

15: ***if*** $j < \beta_p$

Then

16: $P_s == 0$

17: **End**

where $\beta_p \in (0, 1)$ is a constant and P_s is the probability of using the ground truth data as input for the next timestep, which is defined as:

$$P_s = \gamma^{\kappa} \quad (8)$$

where $\gamma \in (0, 1)$ is a constant and $\kappa \in \mathbb{Z}^+$ can be described as an incremental factor, *i.e.*, an integer that is incremented by 1 every k epoch. The γ factor along with the incrementation rate κ are adjusted according to the nature of the problem.

The training process is illustrated in Fig. 6, where, for each batch of FEA simulations with T timesteps, the NEP network is tasked with recurrently predicting the output of each timestep t, starting from the initial timestep t_o. The predicted RPs (yellow tensors in Fig. 6) from timestep t are used as input in timestep t+1 with a probability of 1-P_s ($P_s > P_r$, where P_r is a random number $\in$ [0,1)). Based on Algorithm 1, at the beginning of training, the NEP network is forced to rely heavily on the ground truth. As training progresses, the NEP network starts using its own predictions more frequently and at the last stage of the training phase the NEP network uses only the predicted output as input for the next timesteps. This final stage simulates the testing phase, where only the force, boundary conditions, and the predictions of the NEP from the prior timestep are available, thereby enabling the NEP network to build robustness against accumulating error over long sequence predictions. The gradual transition from ground truth RPs to NEP-predicted RPs serves a dual purpose in training. Initially, by relying on ground truth data, the NEP network can focus on learning a baseline understanding of the material or structural behavior under varied conditions. As the training progresses, the network begins to use its own predictions for subsequent timesteps, shifting the emphasis towards handling the transient nature of long time-sequence simulations. This staged transition enables the NEP network to balance short-term accuracy with long-term stability, effectively minimizing cumulative errors that may arise from sequential predictions in transient conditions. This way, NEP can predict long output sequences without relying on any further updating mechanisms to maintain prediction precision.

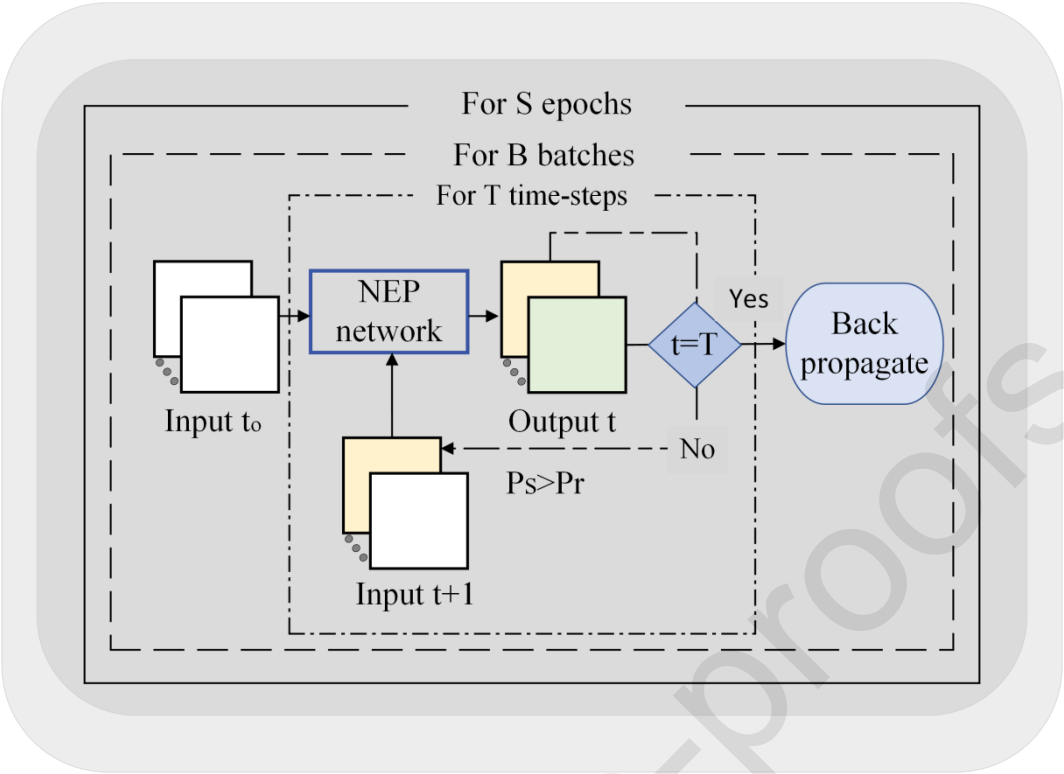


Figure 6 Overview of the training phase.

3.2.2 Node-Element Loss Function

Considering the nature of the PM in the NEP network, the error of both the node and element branch need to be incorporated in the same loss function. Hence, a novel additive loss function, hereinafter called Node-Element (NE) loss, is devised. The formulation of the NE loss is inspired by the Mean Squared Error (MSE) metric; thus, it places greater emphasis on larger errors, accommodating discrepancies across the entire value spectrum of the output parameters and the squaring effect balances the impact of outliers, proving beneficial for handling noisy or irregularly distributed FEA simulation data (Liang et al., 2020). The NE loss function employed in this study can be described as follows:

$$\mathcal{L}_{total} = \zeta_n \cdot \left(\frac{\sum_{n=1}^{N}\sum_{t=1}^{T}(y_{nt}-\hat{y}_{nt})^2}{T\cdot N}\right) + \zeta_e \cdot \left(\frac{\sum_{e=1}^{E}\sum_{t=1}^{T}(y_{et}-\hat{y}_{et})^2}{T\cdot E}\right) \tag{9}$$

where ζ_n and ζ_e are the scaling factors for the node- and element-related output, respectively, N is the total number of nodes, E is the total number of elements, T is the total number of timesteps, $\hat{y}_{nt}$ and y_{nt} correspond to the predicted and ground truth output related to nodes, and $\hat{y}_{et}$ and y_{et} correspond to the predicted and ground truth output related to elements.

By incorporating the predicted output tensor of both branches into an additive loss function, the network can concurrently learn to predict the output parameters for both nodes and elements without being limited by model-specific properties.

4 Experiments and Results

4.1 Datasets

DeepFEA is a generalized framework that can be applied to various simulation problems, such as structural mechanics problems. For the purposes of this study, three different datasets of FEA simulations were used to predict the deformation of an object under different conditions. Given that relevant datasets are not publicly available, we aimed to create datasets that can serve as benchmarks and facilitate comparisons with future studies. In addition, their simplicity allows for the quick assessment of different configurations, *e.g.*, network architectures and parameters. All problems involved in the datasets are governed by the basic equations of continuum mechanics, which are detailed in (Wriggers, 2008). Since these equations are not integral to the proposed architecture, they are not analytically described in this paper. First, two 2D model datasets utilizing material models with linear elastic (LEM) and hyperelastic (HM) characteristics, as well as one 3D LEM dataset were developed. The objects in each dataset were subjected to an external load applied at different outer nodes of the mesh under different angles and magnitudes over a period of 1 s. The software utilized for the generation of the datasets was ANSYS LS-DYNA v11.1. The outputs of all simulations were recorded every 5 ms, resulting in datasets comprising 200 datapoints, *i.e.*, 200 timesteps. The generated meshes contained 9×9 nodes and 64 elements for the 2D cases and 9×9×4 nodes and 192 elements for the 3D case. The force was randomly applied on one external node of each mesh and was linearly increased until the end of the simulation. The force was applied at four different angles {0°, 45°, 90°, 135°} and three different maximum force magnitudes were considered $\{5\times10^5, 10^6, 2\times10^6\}$ N. The non-constrained nodes of the 2D and 3D meshes had 4 and 6 degrees-of-freedom (DOFs), respectively. The bottom nodes of the meshes were set as constrained. The LEM and HM had a mass density of 1200 kg/m^3 each. The Young's modulus and Poison ratio of LEM were 5×10^6 Pa and 0.495, respectively. Regarding HM, the Ogden model was adopted with a Poisson's ratio, first shear modulus, and first exponent parameters of 0.495, 5.978×10^4 Pa, and 12.97, respectively. At each timestep, FEA was tasked with predicting the deformation, *i.e.*, node displacement, and the effective stress and strain of the objects. For the dataset generation, all combinations of angles and magnitudes were considered for all the external non-constrained nodes of the meshes. This resulted in three different datasets containing 450 2D LEM, 450 2D HM, and 2,256 3D LEM simulation cases (Table 3). Thus, these reference datasets contain 3,156 experiments in total with 2,706 different loading conditions (450 different loading conditions for the 2D datasets and 2,256 for the 3D due to the two additional degrees of freedom), two different geometries and two different sets of material properties. As regards the 2D LEM dataset, the effective stress and strain values resided in the intervals $[1.9\times10^{-22}, 2.9\times10^6]$ Pa and $[3.07\times10^{-29}, 5.6\times10^{-1}]$, respectively.

Table 3 Overview of the mesh characteristics, loading conditions, and material properties used for generating each dataset.

Parameters		**2D LEM**	**2D HM**	**3D LEM**
Mesh	Structure	9×9	9×9	9×9×4
	Element No.	64	64	192
	Node No.	81	81	324
Loading Conditions	Maximum applied load	$\{5\times10^5, 10^6, 2\times10^6\}$ N		
	Application	Random external node, load linearly increased		

	DOFs	4	4	6
	Angles	{0°, 45°, 90°, 135°}	{0°, 45°, 90°, 135°}	{0°, 45°, 90°, 135°}
Material properties	Mass density	1,200 kg/m^3	1,200 kg/m^3	1,200 kg/m^3
	Young's modulus	5×10^6 Pa	N/A	5×10^6 Pa
	Poisson's ratio	0.495	0.495	0.495
	First shear modulus	N/A	5.978×10^4 Pa	N/A
	First exponent	N/A	12.97	N/A
Total number of simulated cases under different loading conditions		450	450	2,256
Total number of samples		90,000	90,000	451,200

Note: N/A denotes the non-applicable parameters

The displacement values ranged within [-4.6×10^{-2}, 7.2×10^{-2}] m for the x-axis and [-5.2×10^{-2}, 2.3×10^{-2}] m for the y-axis. As regards the 2D HM dataset, the effective stress and strain values resided in the intervals [5.3×10^{-6}, 2.9×10^{6}] Pa and [3.1×10^{-20}, 4.3×10^{-1}], respectively. The displacement values ranged within [-9.3×10^{-2}, 9.3×10^{-2}] m for the x-axis and [-5.8×10^{-2}, 5.8×10^{-2}] m for the y-axis. In this case of the 3D LEM dataset, the additional dimension (2 more DOFs) provided a larger dataset with compression forces applied to non-constrained nodes on the surface of the mesh, under the same angles and force magnitudes considered in the 2D dataset. The effective stress and strain values resided in the intervals [4.86×10^{-13}, 5.38×10^{6}] Pa and [9.7×10^{-20}, 1.7×10^{-1}], respectively. The displacement values ranged within [-0.13, 0.13] m for the x-axis, [-1.3×10^{-1}, 9×10^{-2}] m for the y-axis, and [-1.7×10^{-2}, 1.9×10^{-1}] m for the z-axis. These intervals, particularly the maximum effective strain reaching up to 5.6×10^{-1}, indicate that the datasets contain FEA simulations with large deformations, thereby enhancing their value as reference datasets.

In this study, the effective stress and strain are calculated based on the von Mises equation that can be described as:

$$\sigma = \sqrt{\frac{(\sigma_{xx} - \sigma_{yy})^2 + (\sigma_{yy} - \sigma_{zz})^2 + (\sigma_{zz} - \sigma_{xx})^2 + 6(\sigma_{xy}{}^2 + \sigma_{yz}{}^2 + \sigma_{zx}{}^2)}{2}} \quad (10)$$

$$\varepsilon = \frac{2}{3}\sqrt{\frac{(\varepsilon_{xx} - \varepsilon_{yy})^2 + (\varepsilon_{yy} - \varepsilon_{zz})^2 + (\varepsilon_{zz} - \varepsilon_{xx})^2 + 6(\varepsilon_{xy}{}^2 + \varepsilon_{yz}{}^2 + \varepsilon_{zx}{}^2)}{2}} \quad (11)$$

where σ denotes the effective stress, (σ_{xx}, σ_{yy},σ_{zz},σ_{xy},σ_{yz}, σ_{zx}) are the stress tensor components, ε denotes the effective strain, and (ε_{xx}, ε_{yy},ε_{zz},ε_{xy},ε_{yz}, ε_{zx}) are the strain tensor components for each element of the FE mesh. In the 2D scenario, the stress components (σ_{zz},σ_{yz}, σ_{zx}) and strain components (ε_{zz},ε_{yz}, ε_{zx}) were zero.

4.2 Evaluation metrics

To quantify the effectiveness of DeepFEA, the R^2 metric, known as coefficient of determination, has been selected, since it is widely used for the evaluation of regression models (Leach & Henson, 2007). R^2 is a statistical measure that indicates the goodness of fit of a regression model, *i.e.*, how well the model's predictions align with the ground truth values. R^2 is usually confined in the [0,1] interval, where 1 indicates that the predictions of a model perfectly fit the ground truth data, whereas 0 indicates a baseline model that always predicts the mean of the ground truth data. R^2 can take negative values in cases where the model is worse than the baseline model. R^2 is defined as:

$$\mathrm{R}^2 = 1 - \frac{\sum_j (y_j - \hat{y}_j)^2}{\sum_j (y_j - \bar{y}_j)^2} \quad (12)$$

where y_j is the j^{th} ground truth value, $\hat{y}_j$ is the j^{th} predicted value, and $\bar{y}$ is the mean of all ground truth values.

The analysis of output parameters, such as stresses and strains, presents a challenge due to their varying order of magnitude, *e.g.*, stress values may span from Pa to GPa. Hence, the normalized Mean Absolute Error (NMAE) and Root Mean Squared Error (NRMSE) have been incorporated into our evaluation study. NMAE and NRSMSE have been reported to be more robust compared to MAE and RMSE when assessing the performance of predictors in similar scenarios (Liang et al., 2020).

$$\mathrm{NMAE}_p = \frac{1}{A}\sum_{j=1}^{A} \frac{\sum_{\mu=1}^{\Gamma} |y_{\mu j} - \hat{y}_{\mu j}|}{\Gamma \cdot (max\{Y_j^p\} - min\{Y_j^p\})} \times 100\% \quad (13)$$

$$\mathrm{NRMSE}_p = \frac{1}{A}\sum_{j=1}^{A} \frac{\sqrt{\sum_{\mu=1}^{\Gamma} (y_{\mu j} - \hat{y}_{\mu j})^2}}{\sqrt{\Gamma} \cdot (max\{Y_j^p\} - min\{Y_j^p\})} \times 100\% \quad (14)$$

where $y_{\mu j}$ is the ground truth output of the μ^{th} node or element for the j^{th} simulation, $\hat{y}_{\mu j}$ is the predicted value of the μ^{th} node or element for the j^{th} simulation, A is the total number of simulations, Γ is the total number of ground truth output values for simulation j, max$\{Y_j^p\}$ and min$\{Y_j^p\}$ are the maximum

and minimum values, respectively, of the set Y containing the output values of the j^{th} simulation for the parameter p.

4.3 Experimental Setup

The performance of the method was evaluated in the context of predicting the effective stresses and strains and the displacement (d_x, d_y, d_z) in each Cartesian axis (x, y, and z (for 3D)). The displacements predicted for each axis were also utilized to calculate the resultant displacement (R_d) of the nodes. An ablation study was conducted based on the 2D LEM dataset to determine the best combination of layers and channels using a 3×3 kernel size for the convolutional layers. The best architecture was selected based on the quantitative evaluation metrics presented in Table 4. Subsequently, the selected architecture was used to demonstrate the capabilities of DeepFEA across a variety of FEA simulation scenarios. More specifically, DeepFEA was trained on the 2D LEM, 3D LEM, and 2D HM datasets, and the results were assessed both quantitatively and qualitatively. Consequently, the experiments conducted for these datasets examined the behaviors of DeepFEA for two different structure types, two different material properties and loading conditions, resulting in total experiments. The NEP network trained on the 2D and 3D datasets comprised 4.7M and 16.7M trainable parameters, respectively.

To evaluate the performance of DeepFEA, the datasets were divided into training and testing subsets, following the guidelines suggested in (Samadian et al., 2024). An ablation study was performed using different proportions of training and testing subsets and the results are summarized in Table 6 and Fig. 7. It can be noticed that the best results were obtained when 80% of the data were used for training and 20% for testing; therefore, this proportion was chosen for the subsequent experiments. The training set was split into 80% for actual training and 20% for validation. This internal validation set was used to fine-tune hyperparameters and to perform early stopping for preventing overfitting. The networks for the 2D and 3D FEA simulation predictions were trained with a batch size of 32 and 16, respectively. The batch size for the 3D simulations was lower due to limitations related to the GPU memory. The optimizer used in the experiments was the Adam optimizer with a variable learning rate associated with the value of the NELO factor P_s. The scaling factors ζ_n and ζ_e of the loss function were set to 10^4. Moreover, the γ factor of the NELO algorithm was set to 0.7, β_p was set to 0.01, and the P_s was reduced every 40 epochs. The model was implemented using the PyTorch v1.12 framework (Paszke et al., 2019) and it was trained with a GeForce RTX 3090 24GB.

Furthermore, DeepFEA was compared with the two related studies identified in Section 2 (Table 2), *i.e.*, StressNet proposed in (Wang et al., 2021) and the hereinafter temporal GNN proposed in (Maurizi et al., 2022). Both methods were trained on the 2D datasets under the same conditions as DeepFEA. For the 3D dataset, DeepFEA was only compared with the temporal GNN (Maurizi et al., 2022), since StressNet (Wang et al., 2021) cannot be applied in 3D problems. Since these methods are not capable of predicting both node- and element-related parameters, they were trained to predict only the deformation of nodes. Therefore, node deformation was selected as the examined parameter, since the goal of this comparison was to assess the ability of these methods to accurately predict dynamic input across multiple timesteps. In the results presented in Sub-section 4.5, the metrics for output parameters that could not be predicted by the other methods, *i.e.*, σ and ε, are denoted as non-applicable (N/A).

4.4 Ablation study

An ablation study was conducted to identify the best architecture through extensive experimentation. DeepFEA was trained on a subset of the 2D LEM dataset with different numbers of layers of varying channel sizes, whereas the kernel size was kept constant to 3×3. As can be seen in Table 4, the overall best model comprised three ConvLSTM layers with 64, 128, and 256 channels. It can be observed that DeepFEA can consistently predict the effective stress and strain of the FEA simulations with great accuracy across all timesteps, with the best model achieving an R^2 of ~0.99.

Table 4 *Quantitative results of the ablation study for models with different ConvLSTM layers trained on a subset of the 2D LEM dataset. Best performance is indicated in bold.*

Architectures		1 layer 64 (3×3)	1 layer 128 (3×3)	1 layer 256 (3×3)	2 layers 64 (3×3) - 64 (3×3)	2 layers 64 (3×3) - 128 (3×3)	2 layers 128 (3×3) - 128 (3×3)	2 layers 128 (3×3) - 256 (3×3)	3 layers 64 (3×3) - 128 (3×3) - 256 (3×3)	4 layers 64 (3×3) - 64 (3×3) - 128 (3×3) - 256 (3×3)	4 layers 64 (3×3) - 128 (3×3) - 128 (3×3) - 256 (3×3)
R^2 ↑	σ	0.964	0.990	0.990	0.991	0.982	0.988	**0.994**	0.993	0.991	0.993
	ε	0.962	0.989	0.989	0.990	0.981	0.988	**0.994**	0.993	0.991	0.993
	d_x	0.445	0.858	0.918	0.927	0.833	0.934	0.949	0.963	0.954	**0.965**
	d_y	0.694	0.893	0.939	0.903	0.898	0.954	0.957	**0.975**	0.969	0.974
	R_d	0.774	0.935	0.967	0.961	0.931	0.971	0.980	**0.984**	0.976	0.980
NMAE (%) ↓	σ	1.509	0.802	0.755	0.698	1.035	0.778	0.559	**0.511**	0.544	0.535
	ε	0.916	0.858	0.806	0.740	1.090	0.833	0.601	**0.544**	0.610	0.570
	d_x	5.324	2.866	2.264	2.225	3.289	2.078	1.778	1.513	1.764	**1.481**
	d_y	3.645	2.124	1.628	2.206	2.106	1.437	1.381	**0.986**	1.088	1.005

	R_d	3.618	2.009	1.554	1.690	2.199	1.455	1.201	1.034	1.679	**1.019**
NRMSE (%) ↓	σ	2.268	1.175	1.169	1.092	1.573	1.226	0.878	**0.853**	0.964	0.904
	ε	2.37	1.256	1.225	1.136	1.644	1.297	0.914	**0.887**	0.989	0.955
	d_x	8.109	4.082	3.224	3.049	4.500	2.915	2.449	2.194	2.397	**2.135**
	d_y	5.049	2.917	2.280	2.836	2.944	2.016	1.858	**1.451**	1.606	1.491
	R_d	5.434	2.880	2.195	2.272	3.062	2.036	1.656	1.506	1.679	**1.473**

Note: ↑ indicates better performance for larger values and ↓ for smaller ones.

It is worth noting that, according to the results presented in Table 4, the node displacement prediction is a more complex task and thus the utilization of a network with higher complexity benefits the overall performance. Nevertheless, based on the same results, it can be observed that an architecture with more layers is not necessarily beneficial for the performance of the method, *i.e.*, the three-layer variant outperforms the four layers ones. To further evaluate the proposed method, the ConvLSTM layers of DeepFEA in the FExM were replaced with simple CNN layers, Bi-LSTM layers, and a variation of the GNN-GRU. Different numbers of layers were considered for the examined ANN layers similarly to the ConvLSTM layers in Table 4 and the results of the best models, *i.e.*, CNN with 7 layers, Bi-LSTM with 4 layers, and GNN-GRU with 6 layers, are presented in Table 5. These networks failed to provide comparable results for every parameter and metric. More specifically, the CNN- and GNN-GRU-based networks achieved worse R^2 values ($R^2 < -1$) than the baseline ($R^2 = 0$) and yielded counterproductive results according to the NMAE and NRMSE metrics (NMAE and NRMSE >100%) for all the output parameters. The effective strain and stress predicted by the Bi-LSTM-based model ($R^2 = \sim0.3$, NMAE = ~6%, and NRMSE = ~9%) were slightly better than those predicted by the other networks, but significantly worse than those predicted by DeepFEA ($R^2 = \sim0.99$, NMAE = ~0.6%, and NRMSE = ~0.9%). As regards the predicted displacement, the performance of Bi-LSTM was similar to that of the other two ANNs ($R^2 < -1$, NMAE & NRMSE >100%). Since these ANNs performed poorly, the following subsections will focus on the results produced by DeepFEA for each dataset using ConvLSTM layers in the FExM.

Table 5 Comparison between state-of-the-art ANN and ConvLSTM layers in the FExM.

ANN Layers		CNN 7 layers	Bi-LSTM 4 layers	GNN-GRU 6 layers	ConvLSTM 3 layers
R^2 ↑	σ	<-1	0.354	<-1	**0.993**

	ε	<-1	0.335	-0.687	**0.993**
	d_x	<-1	<-1	-1	**0.963**
	d_y	<-1	<-1	<-1	**0.975**
	R_d	<-1	<-1	<-1	**0.984**
NMAE (%) ↓	σ	>100	6.093	> 100	**0.511**
	ε	>100	6.290	11.481	**0.544**
	d_x	>100	73.768	>100	**1.513**
	d_y	>100	>100	>100	**0.986**
	R_d	>100	>100	>100	**1.034**
NRMSE (%) ↓	σ	>100	9.517	>100	**0.853**
	ε	>100	9.684	17.994	**0.887**
	d_x	>100	> 100	>100	**2.194**
	d_y	>100	>100	>100	**1.451**
	R_d	>100	>100	>100	**1.506**

Note: ↑ indicates better performance for larger values and ↓ for smaller ones.

Deep learning models require a significant amount of training samples and may perform poorly in terms of generalizability. For transient FEA simulations, it is essential that once the deep learning model has been trained, it can accurately predict the solution of new simulation cases (not included in the training set) for the same FEA model. To determine the optimal trade-off between accuracy and dataset generation time, DeepFEA was trained with different ratios of training and testing samples of the 2D LEM dataset.

Table 6 Quantitative results of the ablation study for training subsets with different percentages (2D LEM dataset). Best performance is indicated in bold.

Metrics	% of Dataset

		20	40	60	80
R^2 ↑	σ	0.873	0.945	0.977	**0.993**
	ε	0.878	0.944	0.976	**0.993**
	d_x	0.614	0.692	0.890	**0.963**
	d_y	0.569	0.792	0.926	**0.975**
	R_d	0.765	0.874	0.945	**0.984**
NMAE (%) ↓	σ	2.122	1.341	0.882	**0.511**
	ε	2.193	1.440	0.937	**0.544**
	d_x	4.776	3.656	2.635	**1.513**
	d_y	3.836	2.687	1.637	**0.986**
	R_d	3.738	2.701	1.833	**1.034**
NRMSE (%) ↓	σ	3.943	2.385	1.518	**0.853**
	ε	3.969	2.499	1.589	**0.887**
	d_x	7.214	5.511	3.695	**2.194**
	d_y	5.899	4.015	2.502	**1.451**
	R_d	5.627	3.942	2.649	**1.506**

Note: ↑ indicates better performance for larger values and ↓ for smaller ones.

The results presented in Table 6 and Fig. 7 indicate that DeepFEA is capable of accurately predicting the effective stress and strain even when trained on 40% of the dataset. It can also be observed that the performance of DeepFEA regarding the predicted displacements is negatively impacted when the NEP network is trained on less than 60% of the dataset.

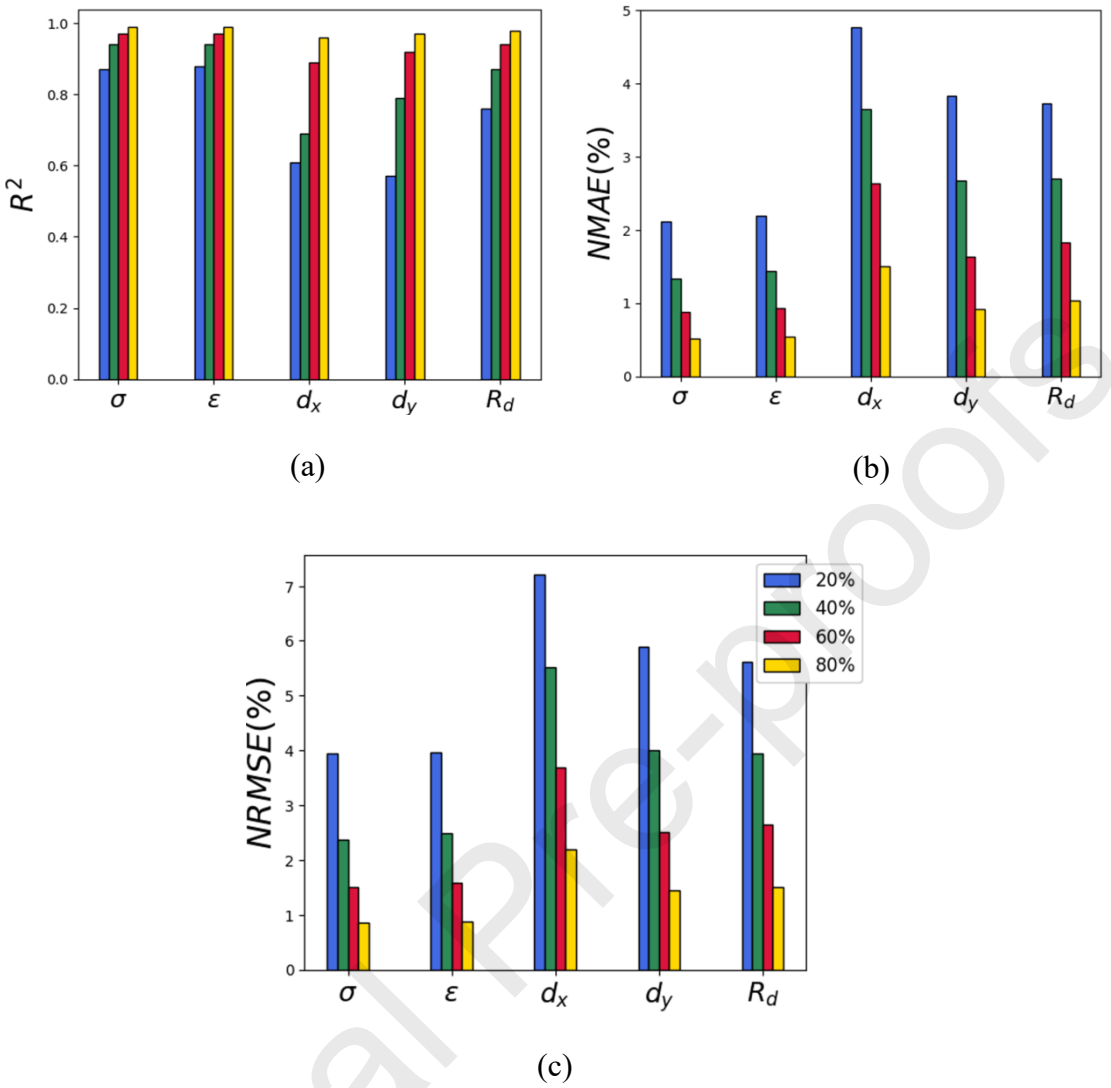


Figure 7 Plots of the results presented in Table 6 for a) R^2, b) NMAE and c) NRMSE under training subsets of 20%, 40%, 60%, and 80%.

As part of the ablation study, DeepFEA was trained on 9 different instances based on randomly sampled consecutive timesteps of varying sequence length of the FEA simulations. Similarly to the other evaluation procedures, the train/test proportions was set to 80%/20%. Then, during inference, DeepFEA was tasked to predict the solutions for the entire simulation. The sequence length for each training instance ranged from 20 to 180 timesteps with a step of 20. For example, during the first training instance sequences of 20 consecutive timesteps were sampled throughthought the simulation duration, *e.g.*, a training sample comrpised a sequence from timestep 25 to timestep 45.

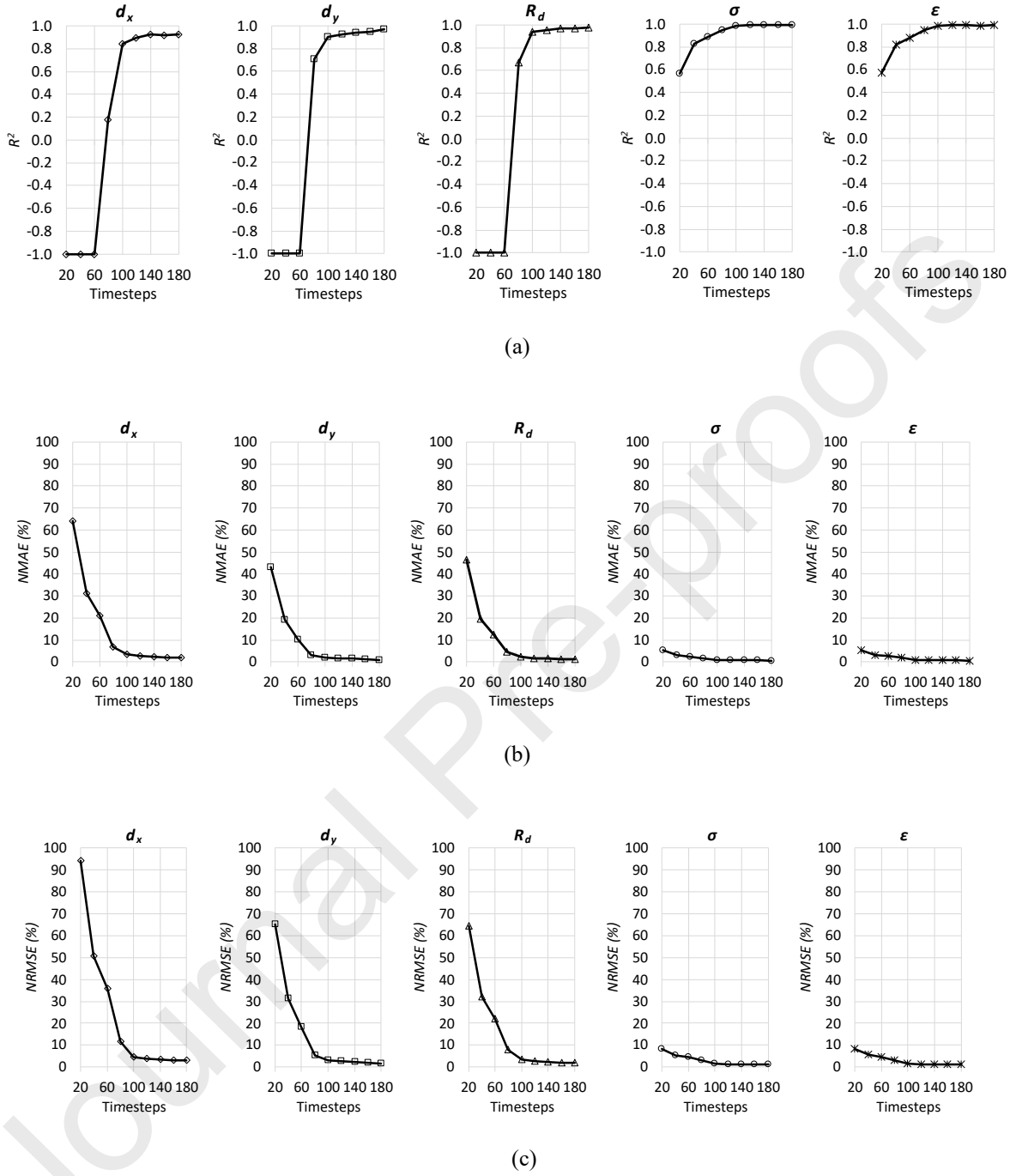
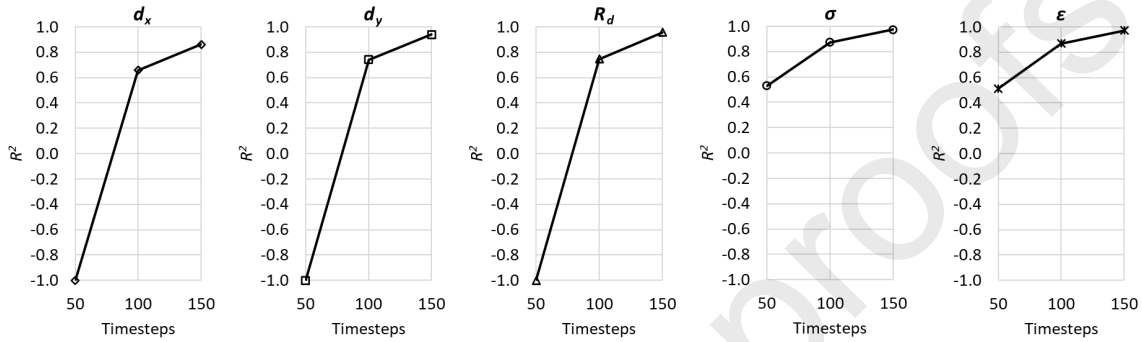


Figure 8 Plots of the prediction performance of DeepFEA for a) R^2, b) NMAE, and c) NRMSE under fixed randomly sampled training sequences ranging from 20 to 180 timesteps.

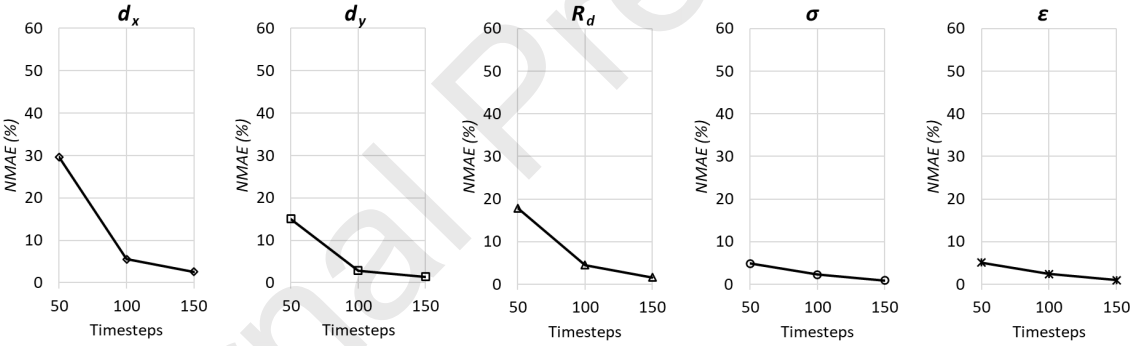
The results in Fig. 8 indicate that DeepFEA was capable of accurately predicting the solutions of the FEA simulations during inference, even when trained with sequences of 100 timesteps, *i.e.*, 50% of the simulation duration, with $R^2 > 0.84$, NMAE < 3.4, and NRMSE = < 4.7% for the deformation and $R^2 = \sim 0.98$, NMAE = ~1%, and NRMSE = ~1.6% for the other parameters, *i.e.*, d_y, R_d, σ, and ε.

To further assess the resilience of DeepFEA to performance degradation on unseen data, it was trained with train/test proportions of 80%/20% to predict parts of the FEA simulation. Instead of using training samples from the entirety of the simulation duration, *i.e.*, 200 timesteps in total, DeepFEA was

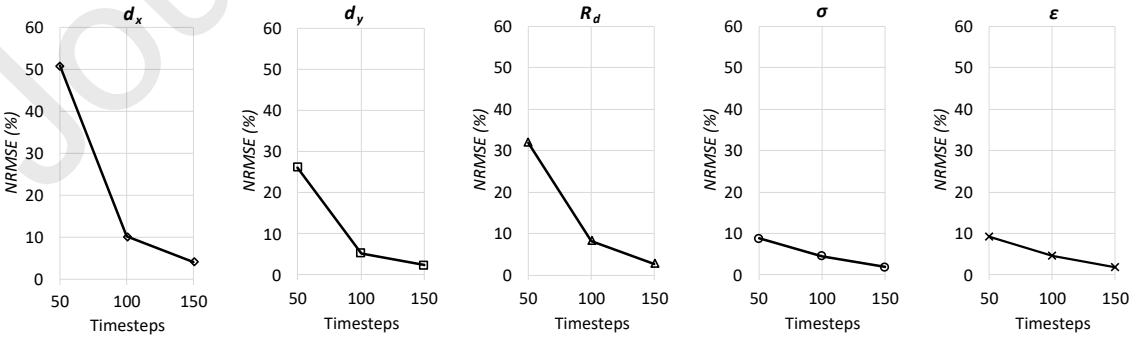
also trained using only the first 50, 100, and 150 timesteps. Then, during inference, DeepFEA was tasked with predicting the entire FEA simulation duration. DeepFEA produced accurate results when trained on only the first 150 timesteps of the simulation with R^2 = ~0.86, NMAE = ~2.6%, and NRMSE = ~4% for d_x and R^2 > 0.94, NMAE < 2%, and NRMSE < 2.6% for the other parameters, *i.e.*, d_y, R_d, σ, and ε. However, DeepFEA experienced a minor performance degradation when trained only on the initial 100 timesteps (R^2 = ~0.66, NMAE = ~5.5%, NRMSE = ~10% for d_x, NMAE = ~ 4.5% and NRMSE = ~8.2% for R_d, and R^2 > 0.74, NMAE < 3%, NRMSE < 5.3% for the other parameters) and experienced further degradation when trained on the first 50 timesteps. As depicted in Fig. 9, this task is more challenging, especially as regards the deformation of the FE mesh.



(a)



(b)



(c)

Figure 9 Plots of the prediction performance of DeepFEA for a) R^2, b) NMAE, and c) NRMSE when trained on the first 50, 100, and 150 timesteps of the FEA simulation duration.

4.5 2D Datasets (shell elements)

4.5.1 LEM Dataset

Based on the results of the ablation study, the model with the best architecture using 80% of the dataset was used for the quantitative and qualitative evaluation of the method.

Table 7 Prediction performance of DeepFEA, StressNet and the temporal GNN trained on the 2D LEM dataset. The metrics were obtained by comparison to the ground truth.

Methods		StressNet	Temporal GNN	DeepFEA
R^2 ↑	σ	N/A	N/A	**0.993**
	ε	N/A	N/A	**0.993**
	d_x	**0.982**	-0.630	0.963
	d_y	-1.833	0.537	**0.975**
	R_d	0.677	-0.212	**0.984**
NMAE (%) ↓	σ	N/A	N/A	**0.535**
	ε	N/A	N/A	**0.570**
	d_x	**0.895**	31.209	1.513
	d_y	13.724	16.588	**0.986**
	R_d	5.884	19.601	**1.034**
NRMSE (%) ↓	σ	N/A	N/A	**0.904**
	ε	N/A	N/A	**0.955**
	d_x	**1.940**	37.805	2.194

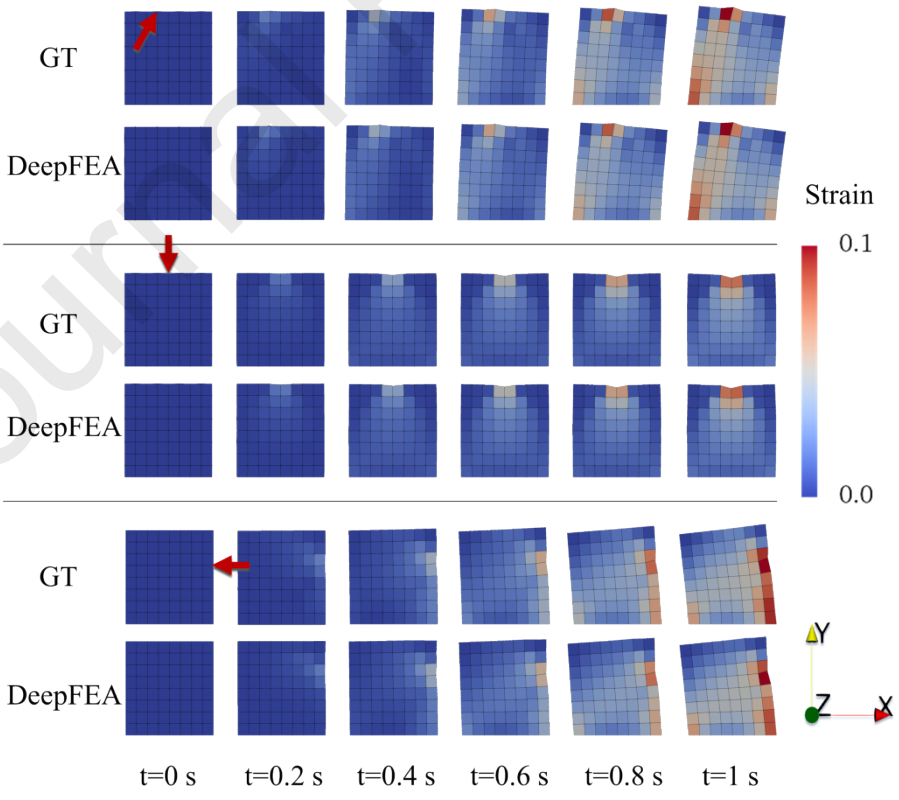
	d_y	29.284	20.292	**1.451**
	R_d	10.807	24.909	**1.506**

Note: ↑ indicates better performance for larger values and ↓ for smaller ones.

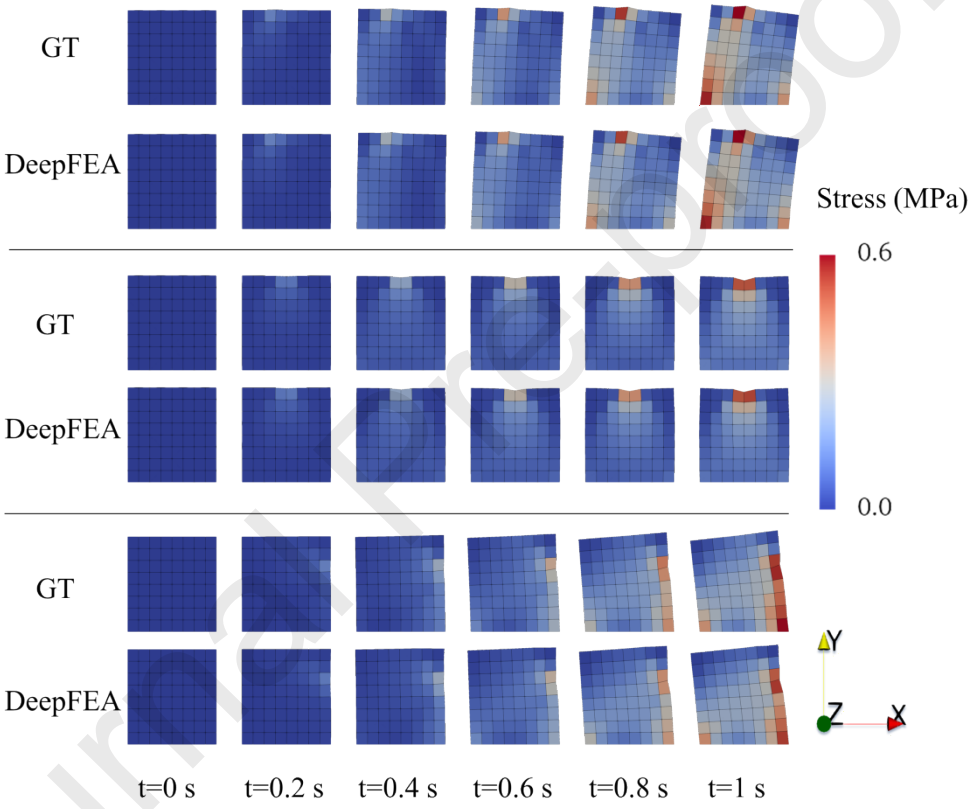
The results in Table 7 indicate that DeepFEA was able to predict all the parameters with great accuracy, exhibiting great consistency throughout the simulation ($R^2 > 0.95$). The values of d_x were predicted with less accuracy compared to those of d_y, which can be attributed to the wider displacement range in the x-axis. Although the predictions for R_d were similar to those for d_y, the normalized errors were higher due to the increased errors in the x-axis.

In Table 7, it can be observed that the use of spatiotemporal ANNs, *i.e.*, StressNet and DeepFEA, can be beneficial when paired with an optimization algorithm tailored for recursive predictions yielding better results than the temporal GNN with $R^2 < 0.6$, NMAE and NRMSE > 16%. Despite that the StressNet yielded better results for d_x ($R^2 = 0.982$, NMAE < 1% and NRMSE < 2%), its performance was worse than that of DeepFEA for d_y ($R^2 < -1$, NMAE = ~14 % and NRMSE = ~30%) and R_d ($R^2 = \sim 0.68$, NMAE = ~5.8% and NRMSE = ~10.8 %). Compared to StressNet and the temporal GNN, DeepFEA achieves better performance, particularly for d_y and R_d, where it demonstrated significantly lower error margins, showcasing its capability to handle both node- and element-related outputs simultaneously.

To qualitatively evaluate the predictions of DeepFEA, three simulation cases were randomly selected from the dataset. Figure 10 presents representative snapshots (five timesteps) of the output parameters (deformation, strain, and stress evolution) predicted for each simulation case.



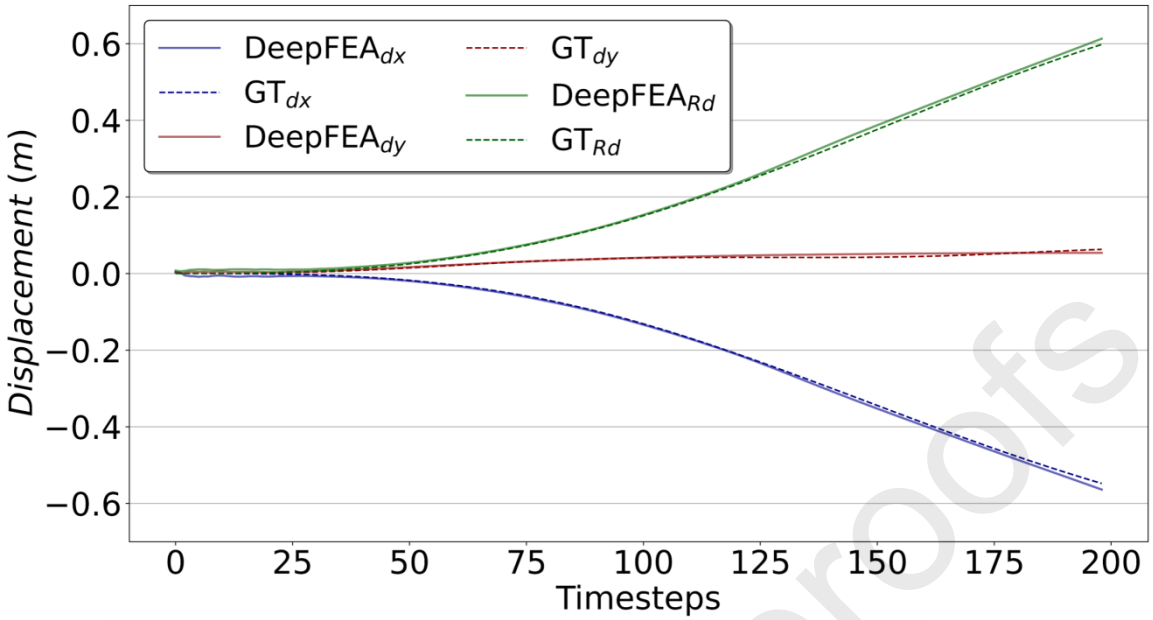
(a)



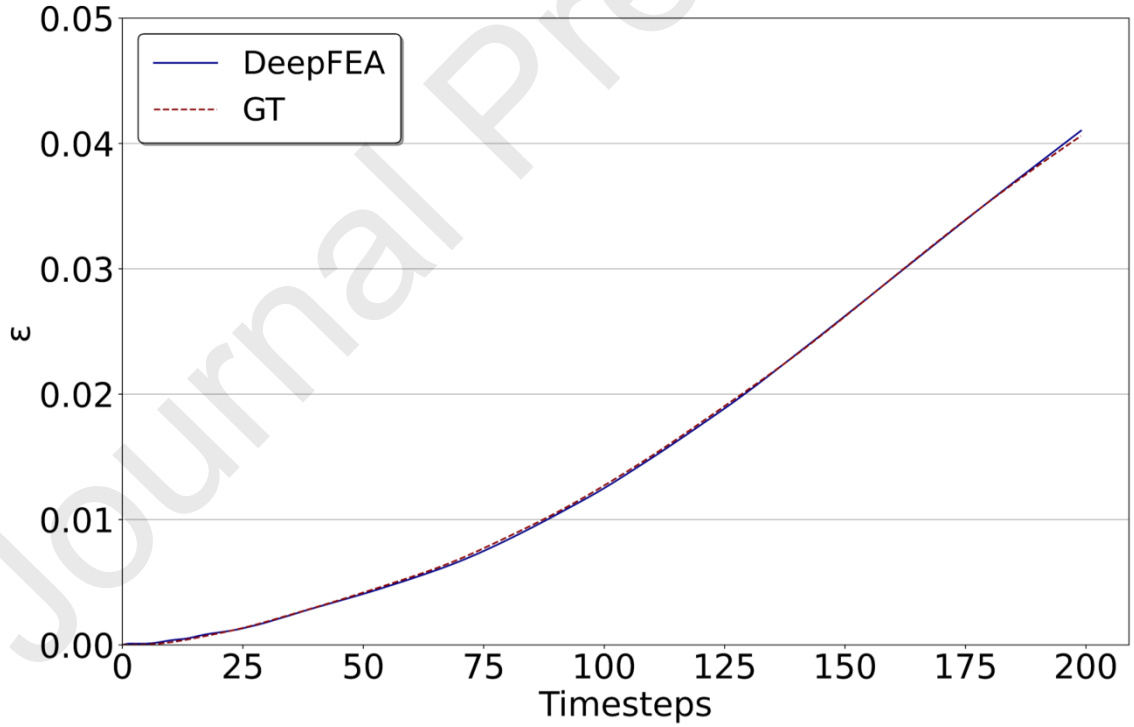
(b)

Figure 10. Mesh deformation and (a) strain and (b) stress contours of three randomly selected simulation cases (2D LEM dataset). The red arrows indicate the position and direction of the applied force. GT denotes ground truth.

In addition, the evolution of the predicted average displacement (Fig. 11a), strain (Fig. 11b), and stress (Fig. 11c) of all nodes and elements for the entire simulation duration were plotted against the ground truth (dashed lines). Figure 11 depicts the average output of the third simulation case in Fig. 10. Specifically, the node displacement (Fig. 11a), effective stress (Fig. 11b) and strain (Fig. 11c). This case was selected as a representative example and further demonstrates that DeepFEA produces accurate predictions for every timestep of the simulation as they align with the ground truth for every examined parameter. Overall, the results demonstrate the ability of DeepFEA to predict the solutions of FEA simulations for different ranges of motion generated by forces applied at different external nodes of the 2D mesh.



(a)



(b)

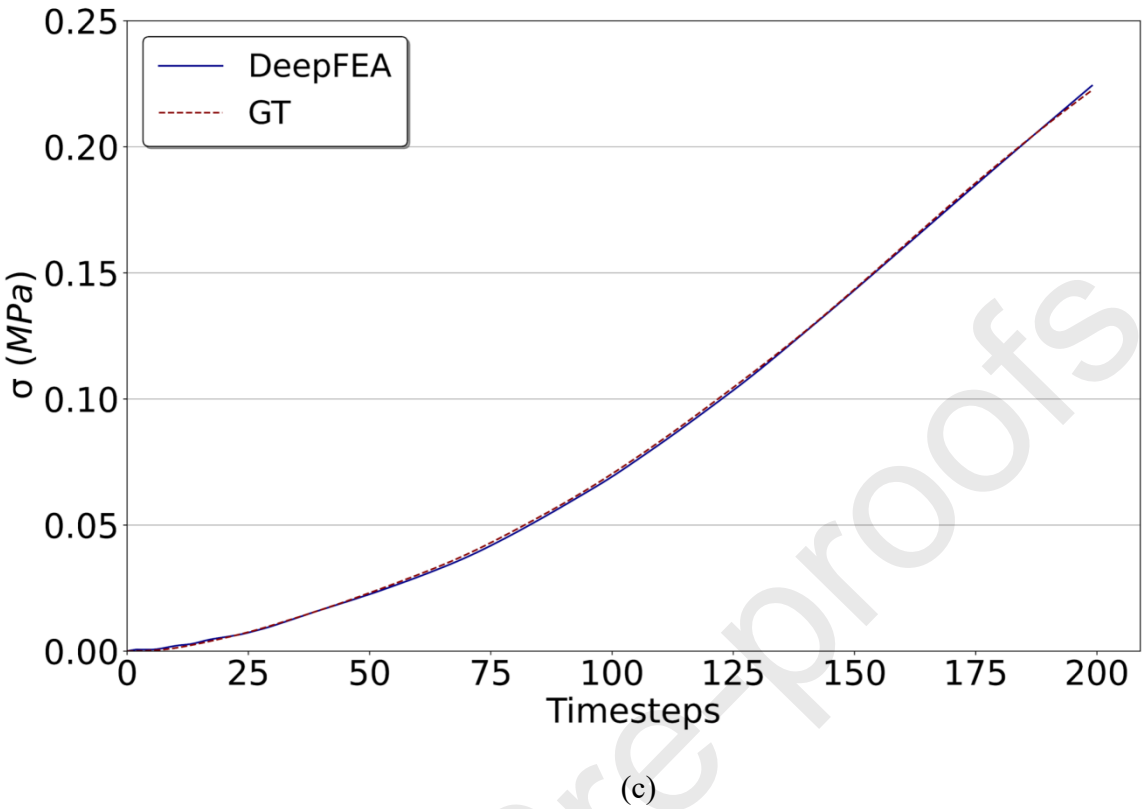


Figure 11. Plots of (a) average displacement, (b) strain, and (c) stress vs. time for the third simulation case in Fig. 9 (2D LEM dataset).

4.5.2 HM Dataset

DeepFEA was also tasked with predicting the solutions of simulation cases from the 2D HM dataset. The quantitative results in Table 8 further validate the ability of DeepFEA to accurately predict the simulation outcome of FEA models with non-linear material properties for both node- and element-related output parameters. Notably, the predicted displacement was on par with the results in Table 8, with the only exception that the normalized errors for the displacement in the x- and y-axes were slightly higher.

Table 8 Quantitative results for DeepFEA, StressNet and the temporal GNN trained on the 2D HM dataset. The metrics were obtained by comparison to the ground truth.

Methods		StressNet	Temporal GNN	DeepFEA
R^2 ↑	σ	N/A	N/A	**0.981**
	ε	N/A	N/A	**0.988**
	d_x	0.449	-1.507	**0.959**

	d_y	0.137	0.765	**0.950**
	R_d	0.820	-0.624	**0.976**
NMAE (%) ↓	σ	N/A	N/A	**0.651**
	ε	N/A	N/A	**1.109**
	d_x	3.073	34.950	1.663
	d_y	6.130	11.282	**1.405**
	R_d	4.021	22.221	**1.270**
NRMSE (%) ↓	σ	N/A	N/A	1.278
	ε	N/A	N/A	**1.732**
	d_x	12.695	43.334	2.587
	d_y	16.054	13.846	**2.323**
	R_d	7.917	28.294	**1.982**

Note: ↑ indicates better performance for larger values and ↓ for smaller ones.

For this dataset, the normalized errors for the effective stress and strain differed from those in Table 7. More specifically, the task of predicting the effective strain in the HM dataset appeared to be more challenging than that in the LEM dataset. Despite that, the effective stress predicted by DeepFEA for the HM dataset achieved only a slightly lower R^2 and higher NRMSE compared to those for the LEM dataset (Table 7). Furthermore, according to Table 8, DeepFEA was more accurate compared to the other methods for all metrics and parameters. In addition, the errors of StressNet with NMAE < 7% and NRMSE < 13% were lower compared to those of the temporal GNN approach with NMAE > 10% and NRMSE > 13%.

4.6 3D LEM Dataset (solid elements)

To assess the capability of the proposed method in the 3D domain, DeepFEA was also evaluated on the 3D LEM dataset. The quantitative evaluation is summarized in Table 9. Despite the increased complexity of the problem due to the additional DOFs, DeepFEA was able to accurately predict σ and ε, with R^2 = ~0.99, NMAE = ~0.49%, and NRMSE = ~0.78%. DeepFEA inferred the resultant displacement of the nodes with high accuracy. Notably, the predicted d_y was less accurate in terms of R^2 compared to the other displacement axes, while the normalized errors were comparable. In addition,

the predicted R_d demonstrated a closer alignment with the ground truth for all metrics. Thus, it can be inferred that, while the predicted displacement along a specific axis might deviate from the ground truth, DeepFEA can accurately capture the resultant deformation. Compared to DeepFEA, the temporal GNN produced worse results with $R^2 < 0.2$, NMAE > 20 %, and NRMSE > 25%.

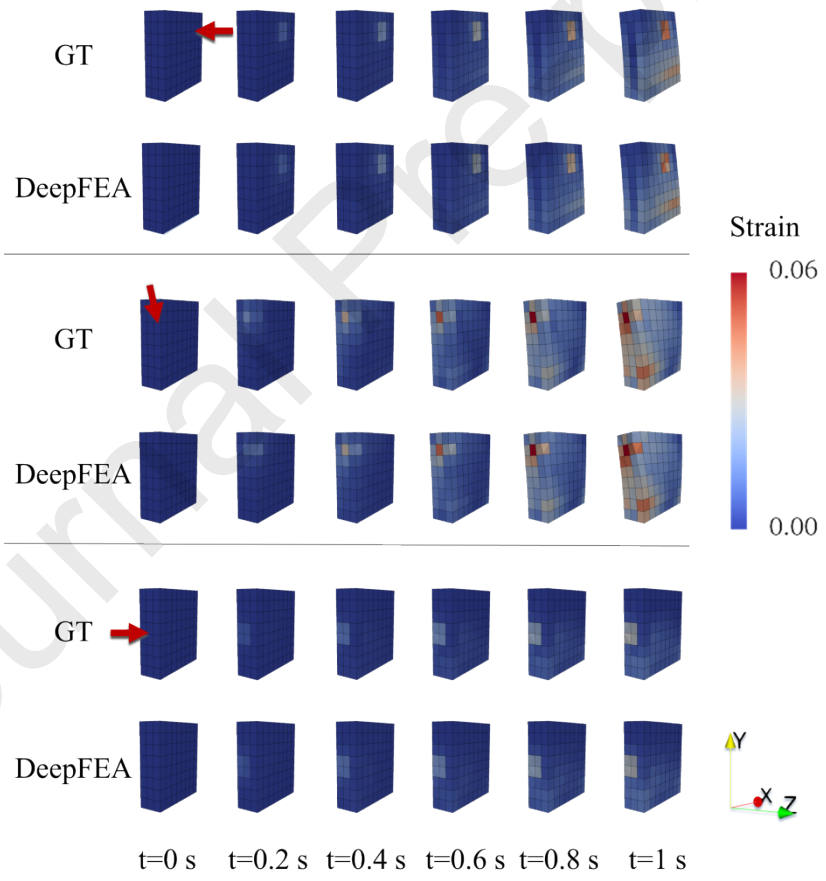
Figure 12 illustrates representative snapshots (five timesteps) of the predicted output parameters (deformation, strain, and stress evolution) of three simulation cases that were randomly selected from the dataset. In the first case, a compression force was applied on an outer node of the mesh with magnitude of 10^6 N, and angle of 0° relative to the z-axis. In the second case, a compression force was applied with a magnitude of 10^6 N and angle of 135° with respect to the x- and y-axis. In the third case, a compression force was applied with a magnitude of 5×10^5 N and angle of 0° with respect to the x-axis.

Table 9 Quantitative results for DeepFEA and the temporal GNN trained on the 3D LEM dataset. The metrics were obtained by comparison to the ground truth.

Methods		Temporal GNN	DeepFEA
R^2 ↑	σ	N/A	**0.976**
	ε	N/A	**0.976**
	d_x	-0.559	**0.829**
	d_y	0.206	**0.725**
	d_z	-11.547	**0.862**
	R_d	-0.280	**0.941**
NMAE (%) ↓	σ	N/A	**0.660**
	ε	N/A	**0.657**
	d_x	32.651	**1.985**
	d_y	22.230	**1.730**
	d_z	68.070	**1.253**
	R_d	21.956	**1.034**

NRMSE (%) ↓	σ	N/A	**1.173**
	ε	N/A	**1.174**
	d_x	39.008	**2.780**
	d_y	28.104	**2.448**
	d_z	105.897	**2.449**
	R_d	25.892	**1.737**

Note: ↑ indicates better performance for larger values and ↓ for smaller ones.



(a)

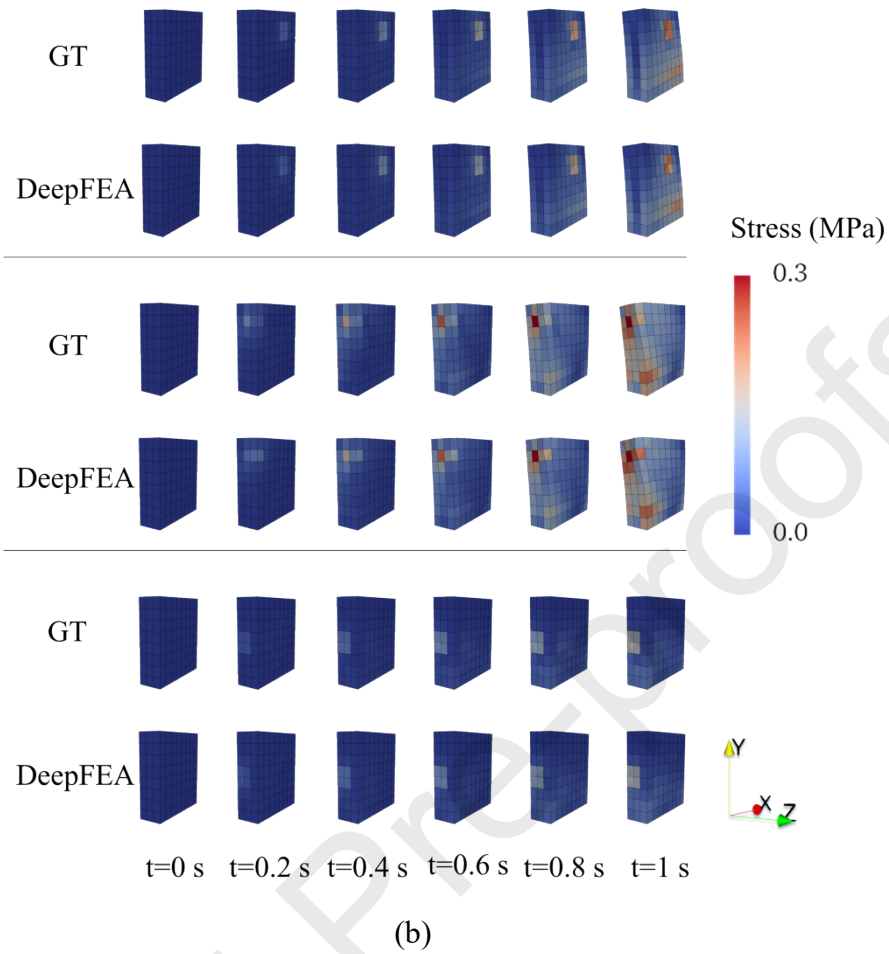
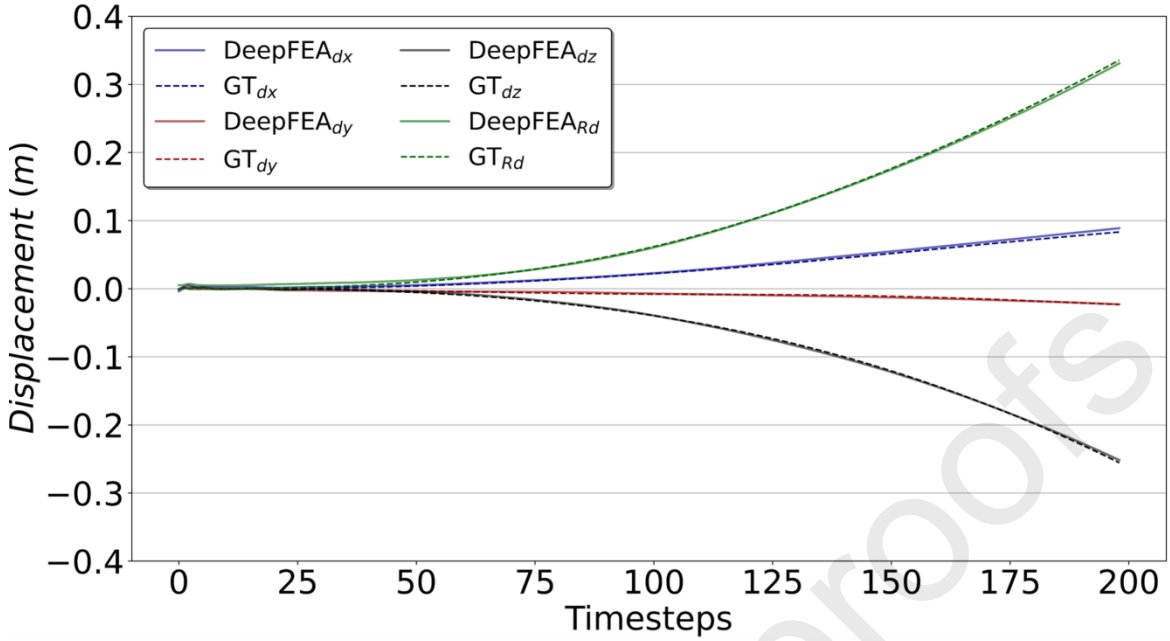
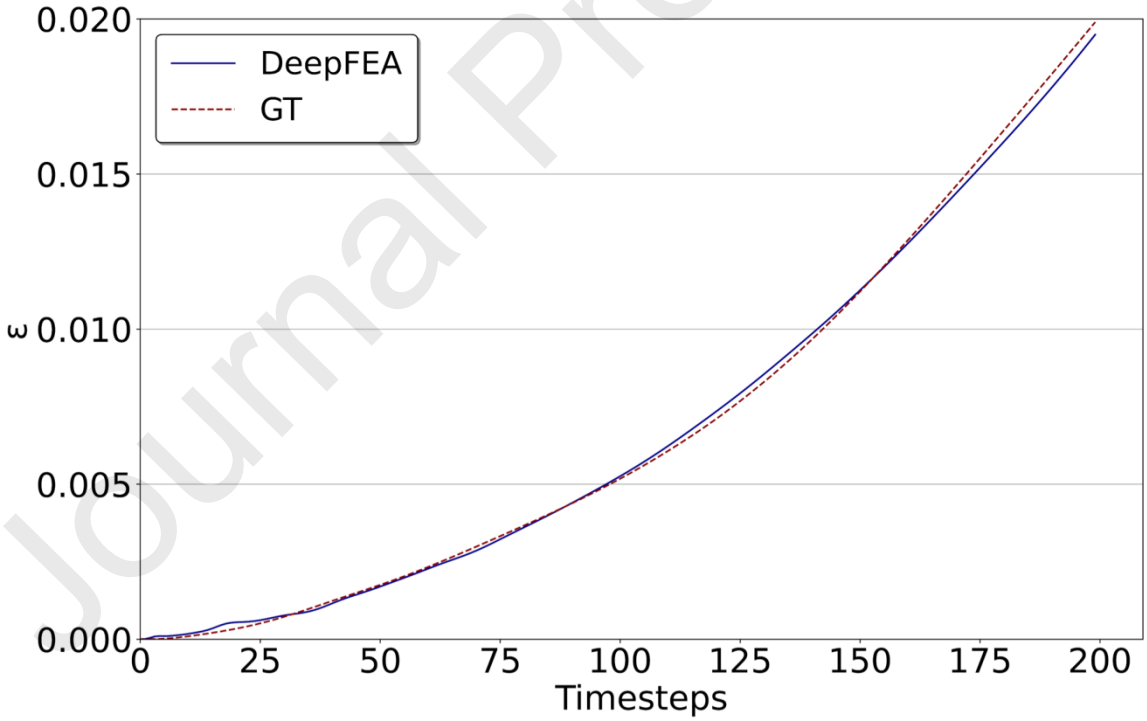


Figure 12. Mesh deformation and (a) strain and (b) stress contours of three randomly selected simulation cases (3D LEM dataset). The red arrows indicate the position and direction of the applied force. GT denotes ground truth.

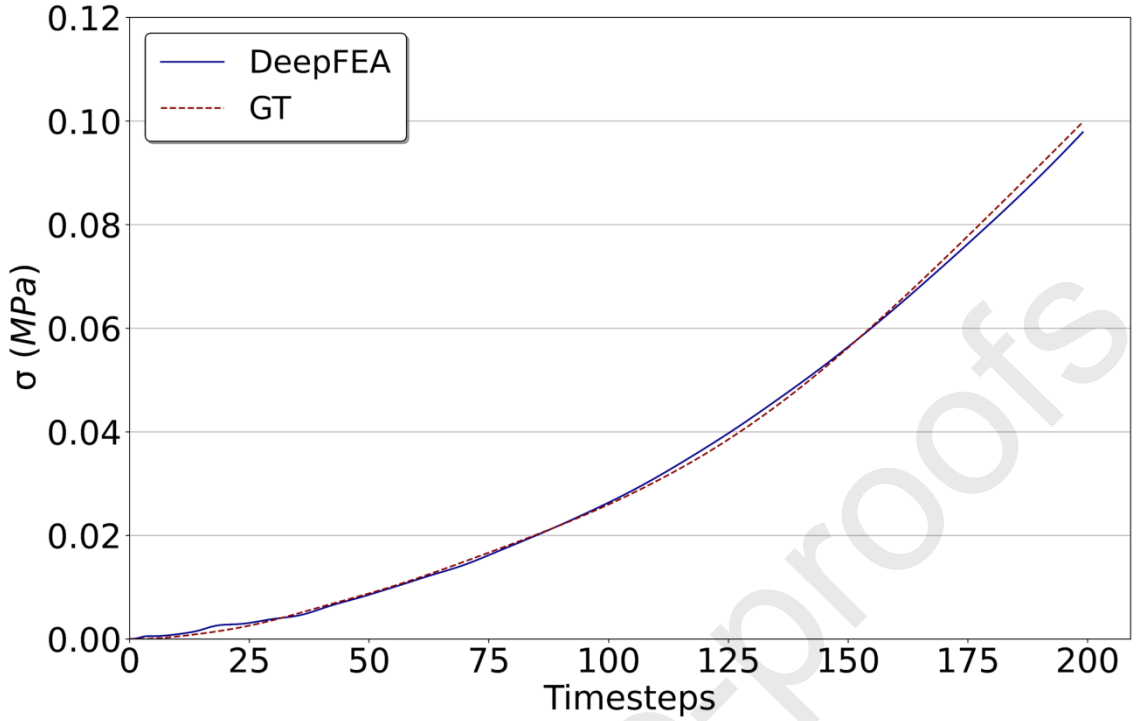
Furthermore, the second simulation case presented in Fig. 13 was selected as an example and the evolution of the average displacement (Fig. 13a), strain (Fig. 13b), and stress (Fig. 13c) of all nodes and elements for the entire simulation duration predicted by DeepFEA was plotted against the ground truth. The effect of the constrained nodes on the y-axis displacement can be observed in Figs. 12 and 13a. The deformation along the y-axis is smaller than that in the x- and z-axes due to the limited range of motion.



(a)



(b)



(c)

Figure 13. Plots of (a) average displacement, (b) strain, and (c) stress vs. time for the second simulation case in Fig. 11 (3D LEM dataset).

This could explain the discrepancy between the R^2 and normalized errors in Table 4 for the y-axis. Moreover, the minor error margins of the predicted σ and ε in Table 4 can be attributed to overestimations of the values of certain elements near the applied force position at the last timesteps of the simulation or due to marginal deviations throughout the entire simulation. Nevertheless, the predictions of DeepFEA align closely with the ground truth for every examined parameter.

4.7 Inference Time

The results have demonstrated the performance of DeepFEA in terms of accuracy. Hence, a comparative analysis of the average CPU- and GPU-enabled inference times of DeepFEA and the CPU-enabled data generation time of FEA for each experimental scenario was conducted (Table 10).

Table 10 Average CPU and GPU enabled inference time (in seconds) of DeepFEA compared to CPU enabled generation time of FEA for one simulation.

	Datasets					
Methods	2D LEM		2D HM		3D LEM	
	CPU (s)	*GPU (s)*	*CPU (s)*	*GPU (s)*	*CPU (s)*	*GPU (s)*

FEA	22	*N/A*	25	*N/A*	40	*N/A*
DeepFEA	2.35	0.20	2.35	0.20	15.30	0.23

The resulted average inference times were obtained on a computer system with an Intel(R) Core(TM) i7-9750H CPU at 2.60GHz, a GeForce RTX 3090 24GB, and a 16 GB RAM. As regards the 2D datasets, DeepFEA could predict the solution of a FEA simulation in 0.20 s and 2.35 s on average using a GPU and CPU, respectively, whereas FEA needed 24 s. As regards the 3D LEM dataset, the inference time of DeepFEA for one simulation was 0.23 s (GPU) and 15.30 s (CPU) on average, whereas FEA required 40 s. This indicates that under the same conditions and the utilization of a GPU, the proposed method is up to two orders of magnitude faster than FEA for the 2D and 3D scenarios. Furthermore, the average inference time of DeepFEA was consistent for both the 2D and 3D domains when a GPU was employed, in contrast to the average solution time of FEA that exhibited a two-fold increase.

4.8 Application study

To demonstrate the applicability of DeepFEA in real-life scenarios, an example case study was conducted in the context of SoftReach[1], which is a European Project under which this study is performed. The research involves attaching a payload (special cells or drugs) to the wall of the brain ventricles. The payload is attached to the brain wall under a load exerted by a catheter. To ensure secure attachment while avoiding tissue damage, the load magnitude and its application orientation (angle) need to be optimized. While this can be assessed preoperatively, unpredictable changes that may occur intraoperatively, *e.g.*, new position or catheter orientation, may require readjustment of the plan and simulations under the new situation. Since this would require fast calculations and decision making, DeepFEA constitutes a pioneering approach towards providing such solutions rapidly during the operation. However, it should be noted that this is only the first step towards the project's endeavor.

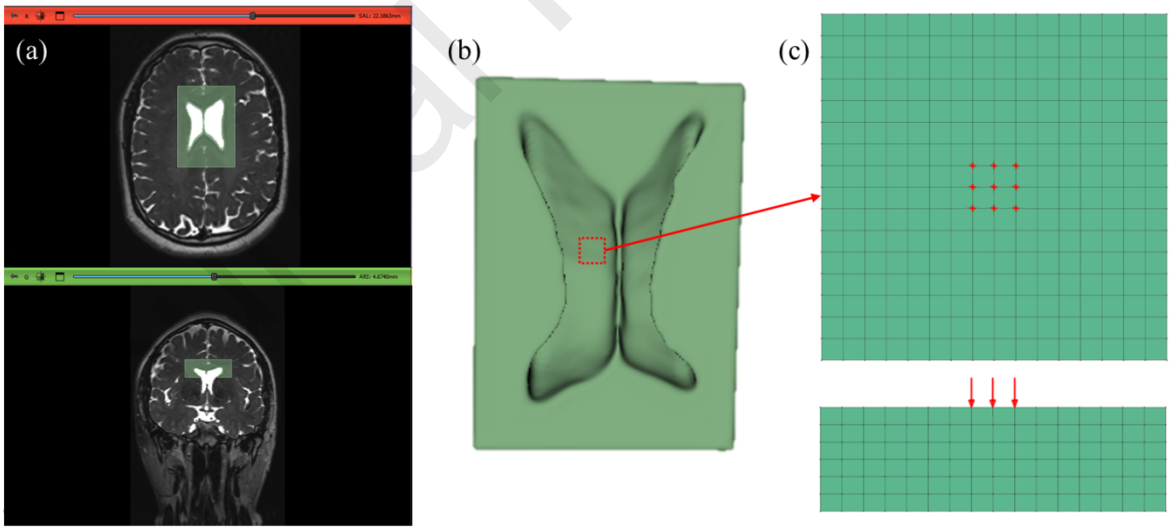


Figure 14 (a) Segmentation of the region of interest from a brain MRI dataset; (b) 3D model of the brain tissue proximal to the upper portion of the lateral ventricles; (c) Meshed 3D model (top and side views) of the isolated region where the payload will be attached. The red points and arrows indicate the nodes where the load is applied.

Given the above background, a simplified brain wall tissue model was developed (Fig. 14) and a dataset comprising simulations performed under various load magnitudes and application orientations was created. In particular, a brain MRI dataset was imported into the 3D Slicer software (v5.6.2) and

[1] https://softreach.eu/

the region of interest, *i.e.*, brain tissue proximal to the upper portion of the lateral ventricles, was segmented (Fig. 14a). Subsequently, the segmented geometry was exported in STL format (Fig. 14b) and a square portion of the brain wall measuring 10 mm × 10 mm × 3 mm (length × width × depth) was isolated and meshed with 1536 hexahedral elements and 2023 node (Fig. 14c). All the outer nodes apart from those of the wall surface were constrained to mimic the resistance from the surrounding brain tissue. The load was applied to 9 nodes at the center, representing the size of the catheter (~Ø1.2 mm) that applies the load in the actual condition (Fig. 14c). A visco-hyperelastic material model for soft tissues was used to model the ventricular wall (corpus callosum) and the material properties and model constants were obtained from (Lyu et al., 2022). The applied loads ranged from 0.5 mN to 3 mN and the application angles ranged from 0° (normal to the surface) to 45°. The duration of the simulation problem was 1s comprising 21 timesteps for each simulation case. Again, all simulations were performed in ANSYS LS-DYNA v11.1. Subsequently, the dataset involved 80 different loading conditions. For these experiments, DeepFEA was trained to predict σ, ε, and R_d under the same experimental setup described in Sub-section 4.1. The results presented in Table 11 indicate that DeepFEA was able to accurately predict all the examined parameters, *i.e.*, σ, ε, and R_d. ($R^2 > 0.87$). Notably, the prediction of ε was more challenging than σ, as in the case of 2D HM dataset (Table 9). The inference time for DeepFEA was 0.4 s on average compared to FEA that was 180s.

Table 11 Predictive performance of DeepFEA trained on the VHM dataset. The metrics were obtained by comparison to the ground truth.

Metrics		DeepFEA
R^2 ↑	σ	0.878
	ε	0.892
	R_d	0.996
NMAE (%) ↓	σ	0.821
	ε	1.683
	R_d	0.694
NRMSE (%) ↓	σ	1.866
	ε	2.705
	R_d	1.038

Snapshots depicting the evolution of effective strain, and stress of a randomly selected simulation case are presented in Fig. 15. As can be observed, nodes and elements closer to the applied load position (Fig. 11c) were affected significantly more by the applied load (Fig. 15); thus, the effective strain and stress was larger compared to the rest of the mesh. Nevertheless, DeepFEA was able to adapt to this challenging scenario providing accurate results.

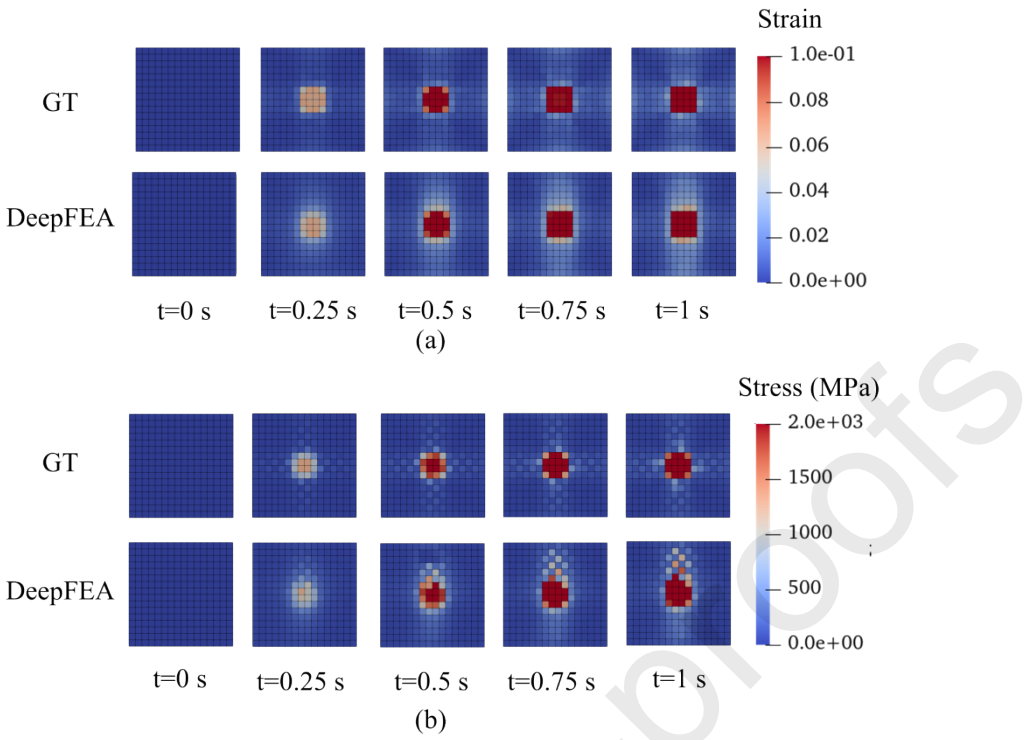


Figure 15 Effective (a) strain and (b) stress for one randomly selected simulation case. GT denotes ground truth.

5 Discussion

In this study, a novel deep learning surrogate model for transient FEA simulations has been proposed. FEA-related studies have focused on surrogate models for steady-state analysis (Deshpande et al., 2022; Du et al., 2022; Farajtabar et al., 2022; Ibragimova et al., 2022; Khanolkar et al., 2021; Krokos et al., 2022; Mendizabal et al., 2020; Phellan et al., 2021). Nevertheless, there have been only few early studies on surrogate models that are aimed for transient FEA simulations and have provided insight into the current progress and limitations (G. Chen, 2021; Frankel et al., 2020; Ghavamian & Simone, 2019; Maurizi et al., 2022; Totounferoush et al., 2021; Wang et al., 2021; X. Zhao et al., 2021). Limitations of these approaches include failure to account for spatiotemporal features, applicability limited to the 2D domain, and dependency on FEA for calculating part of the solution. To address these issues, DeepFEA was designed to predict the solutions of the entire transient FEA simulation given only the initial conditions, the external load, and the boundary conditions.

Mesh deformation is a crucial factor involved in structural mechanics (Deshpande et al., 2022; Frankel et al., 2020; Ibragimova et al., 2022; Liang et al., 2020; Maurizi et al., 2022; Pellicer-Valero et al., 2020). In transient FEA simulations, the deformation of the mesh depends on the predictions of the previous timesteps. Therefore, the core problem addressed by DeepFEA is the efficient prediction of RPs (here, node displacement). Related studies, such as StressNet (Wang et al., 2021), have proposed solutions for predicting such parameters by using a specialized optimization algorithm but are limited to the 2D domain and account for single-output predictions. Other methods, such as the temporal GNN in (Maurizi et al., 2022), can provide multi-dimensional output but do not offer a specific mechanism for handling dynamic input predictions. In contrast, DeepFEA leverages two parallel CNN branches, enabling multi-output predictions for nodes and elements, while also handling the challenging task of dynamic input prediction. DeepFEA provides a framework that is capable of accurately predicting both RPs and NRPs of FEA simulations in the 2D and 3D domains. This is achieved by the implementation of a data-driven approach that incorporates the physical properties of the model in the input tensor (coordinates of the nodes, external load, and boundary conditions) and the utilization of the SSM training scheme. Thus, DeepFEA can be trained for FEA simulations with an arbitrary number of input

and output parameters. In addition, DeepFEA is not limited to a specific problem, and, with proper adjustments, it can be applied to FEA models with various characteristics and properties. These adjustments could be facilitated in future studies by hyperparameter optimization algorithms, such as gravitational search algorithm (Rashedi et al., 2009) and inclined planes system optimization (Mozaffari et al., 2016), enabling automatic tuning of parameters used in the training phase, *e.g.*, in NELO.

The proposed approach of incorporating the coordinates as input features gives an additional merit to DeepFEA. As the structure of the mesh changes due to deformation, the connections between the nodes remain the same, whereas the input channels related to the coordinates of the nodes change to reflect the deformation of the mesh; this enables DeepFEA to solve the regression problem. Thus, this approach can be applied to unstructured meshes by mapping the nodes and elements of the mesh to appropriate tensors similarly to the examined cases of distorted FE meshes due to deformation. As regards the internal load distribution, related studies have either used the loads calculated by FEA as input (Arcones et al., 2022; Totounferoush et al., 2021) or tried to approximate the effect of the external load on each node via elaborate approximation schemes (Pellicer-Valero et al., 2020). On the contrary, DeepFEA relies only on the external load, which demonstrates that it is capable of understanding the complex physical interactions between nodes and encompasses the approximation of the internally applied load in its weights. Moreover, the comparison of different ANNs in Sub-section 4.4 emphasizes the beneficial effect of the ConvLSTM layers in the FExM of DeepFEA. Although the examined networks (CNNs, LSTMs, and GNN-GRU) have been successfully employed in different FEA simulation domains (*e.g.*, structural analysis (Ghavamian & Simone, 2019; Krokos et al., 2022; Maurizi et al., 2022), FSI (Totounferoush et al., 2021)), they are not able to simultaneously predict the deformation, stress, and strain of a mesh that is subjected to an external load over a period of time. This further highlights the need for building surrogate models that can generalize to different domains, such as structural mechanics, fluid dynamics, and FSI, without extensive modifications, *i.e.*, utilizing different ANN architectures for each FEA application. In this way, the evaluation of such models could be standardized, enabling a more efficient validation and comparison process.

DeepFEA was evaluated against three datasets, *i.e.*, 2D LEM, 2D HM, and 3D LEM. These datasets were selected for the performance evaluation of the proposed model in the 2D and 3D domains, as well as to assess the capability of DeepFEA to predict the solutions for models with non-linear material properties (2D HM). An additional merit of these datasets is that they describe FEA simulations with large deformations (maximum effective strain of up to 5.6×10^{-1}). The evaluation results indicate that DeepFEA is capable of accurately predicting the solutions for both the 2D and 3D LEM datasets. The quantitative results provided by DeepFEA (Tables 7, 8 and 9) indicate its capacity to provide consistently accurate predictions during the entire simulation. This is also reflected in the qualitative results (Figs. 10-13). The NRMSE and NMAE for the displacement prediction of each node was estimated to be less than 3%, whereas its predictions regarding σ and ε were below 1%. According to the quantitative results, the 3D scenario is more demanding as regards the prediction of the mesh deformation. This can be attributed to the additional DOFs, *i.e.*, translation and rotation along the z-axis, which make the deformation characteristics of the mesh more complex. Nevertheless, the results obtained for the 3D case were comparable to those for the 2D one (Table 9). By achieving normalized errors less than 3% and $R^2 > 0.8$ for most sequential predictions of up to 200 timesteps, NELO enables DeepFEA to maintain precision in extended sequence predictions without the need for model updating mechanisms.

Comparisons with two relevant state-of-the-art methods further highlighted the advantages of DeepFEA. First of all, both methods were incapable of predicting both node- and element-related output. StressNet, which is tailored for predicting dynamic input, was capable of predicting the output of the 2D datasets and produced more accurate results only for the d_x with $R^2 = \sim0.98$, NMAE < 1%, and NRMSE < 2% in the 2D LEM dataset. Furthermore, StressNet relies on an initial amount of ground truth data during inference, whereas DeepFEA depends only on the initial conditions of the simulations and can also predict node and element-related outputs. Moreover, the temporal GNN produced comparable results only for d_y with $R^2 < 0.8$, and NMAE and NRMSE > 10% for all datasets. Lastly, DeepFEA outperformed the compared methods for all other examined parameters. Based on these

comparisons, DeepFEA combines the advantages of both StressNet, *i.e.*, the capacity to handle dynamic inputs and the temporal GNN, *i.e.*, the capacity to handle multi-dimensional output predictions. Furthermore, the results demonstrate that these features have a tangible impact on performance. In addition, it can be inferred that the use of a dedicated optimization algorithm tailored for transient FEA simulations, such as the proposed NELO, combined with appropriate adjustments to the NEP network (Table 1 and Table 4) positively affect the predictive performance of DeepFEA since it provides accurate results even for cases of FEA simulations of high non-linearity and large deformations, as well as for extended sequence predictions.

Furthermore, the ablation study revealed that DeepFEA is capable of accurately predicting the effective stress and strain of new FEA simulation cases using only 40% of the generated dataset to train the NEP network, while 60% is sufficient for the displacements. Depending on the examined problem and the available computational resources, DeepFEA can be trained on a smaller dataset of FEA simulation cases while maintaining sufficient accuracy. Further experiments demonstrated the ability of DeepFEA to accurately predict the solutions of the entire simulation during inference when trained with randomly sampled sequential timesteps equal or greater than 50% of the simulation duration, *i.e.*, ≥100 timesteps (Fig. 8). In addition, DeepFEA demonstrated robust extrapolation capabilities, maintaining high accuracy when trained on the first 75% of the simulation duration, *i.e.*, 150 timesteps, and only minor performance degradation when trained on the first 50% of the simulation duration, *i.e.*, 100 timesteps (Fig. 9). These attributes could be beneficial in cases of FEA simulations that require significant computational time to be generated. Reducing the computational time translates to reduced use of computational resources, leading to significant cost savings, *e.g.*, energy consumption, hardware needs, resource allocation, *etc.*, compared to traditional FEA methods. Thus, the economic implications of FEA surrogate models might be an interesting topic for future studies.

Apart from its capacity to be applied on both the 2D and 3D domains, DeepFEA can accurately predict the solutions of simulations involving hyperelastic models. Therefore, it can identify the complex characteristics of different models and differentiate between simulations involving linear or non-linear responses. It should be noted that in the context of the HM dataset, the prediction accuracy of DeepFEA was higher with respect to the stresses. On the other hand, the results for the LEM dataset exhibited similar performance both for ε and σ. This may stem from the non-linear characteristics of the material model and the CNN layers in the element branch of DeepFEA, where the same kernels are used to predict the output for the effective stress and strain. During training, the loss function of the element branch generates the average error for both σ and ε. In the case of HM, due to the non-linearity, ε produces more complex outputs than σ, which may cause the weights of the kernels to converge on a solution that generates a larger error for ε. This averaging effect can also be observed in the LEM dataset, where σ and ε converged with the same error. Hence, in future studies, it would be beneficial to examine different loss functions, *e.g.*, incorporating different weights for each variable, that can mitigate this effect. In addition, the datasets examined in this study comprised 3,236 simulation cases with 2,786 different loading conditions (450 for the 2D datasets and 2,336 for the 3D datasets), three different geometries, and three different set of material models, namely linear elastic (Sub-section 4.5.1 and Sub-section 4.6), hyperelastic (Sub-section 4.5.2) and visco-hyperelastic (Sub-section 4.8). This diversity highlights the impact of varying material properties, loading conditions and structure types on the performance of DeepFEA, providing a comprehensive evaluation of its robustness across different FEA scenarios. Nevertheless, a more in-depth analysis exploring the correlation between the averaging effect and material properties such as the Young's modulus and Poisson's ratio could also be conducted in future work. As presented in Sub-section 4.7, the inference time for both the 2D and 3D datasets was significantly lower than the FEA solution time. Therefore, DeepFEA can substantially reduce the computational time needed for studies that examine specific behaviors under different conditions.

The application study presented in Sub-section 4.8 further highlights the potential of DeepFEA to provide rapid and accurate predictions in a clinical setting, where intraoperative and/or preoperative decision-making is crucial. DeepFEA accurately predicted the effective stress, strain, and resultant displacement ($R^2 > 0.87$, NMAE < 1.7% and NRMSE < 2.7%) under various load magnitude and orientation combinations, exhibiting its robustness. The inference time of DeepFEA was significantly

lower (0.4s) compared to that of traditional FEA (180 s), indicating its efficiency. These findings highlight the applicability of the proposed method to such practical applications that involve complex materials, *e.g.*, with visco-hyperelastic properties, as well as the advantages it offers in terms of efficiency. Therefore, future work will focus on extending the capabilities of DeepFEA to more complex and diverse FEA problems by investigating further enhancements that will pave the way for broader clinical applications.

6 Conclusions

This work presented a novel deep learning approach for predicting the solution of transient FEA simulations. DeepFEA is able to recurrently predict the output parameters for all timesteps in a simulation by utilizing a ConvLSTM-based network with two parallel convolutional branches and an NLP-inspired training scheme. To the best of our knowledge, this is the first time that the SSM training scheme has been used for predicting outputs of FEA simulations. The proposed method was evaluated in the domain of structural mechanics via 2D and 3D FEA models, as well as for two different material models (linear elastic and hyperelastic). The results indicated that DeepFEA can predict the evolution of the output parameters with great accuracy in all scenarios, being up to two orders of magnitude faster than FEA. The efficient inference time combined with the accurate predictive performance of DeepFEA in estimating solutions for FEA simulations across multiple timesteps underscores that, after training, it is able to predict the evolution of RPs and NRPs in a simulation without the assistance of FEA. DeepFEA can be trained to predict the solutions of several simulations involving models with varying boundary conditions and external loads, handle distorted meshes (caused by deformation) and adapt to different material models. Moreover, the ablation study showcased the potential of DeepFEA to accurately predict the solutions of FEA simulations when trained with fewer training samples. In addition, DeepFEA does not require the provision of pre-calculated or approximated internal loads. This achievement lays a robust foundation for future research in the field of transient FEA by utilizing similar approaches. Despite its advantages, DeepFEA can be trained to predict the solutions of transient FEA simulations for FEA models with the same mesh size and density. This limitation is due to the fixed dimensions of the weights in the ConvLSTM layers of the FExM. Thus, a different NEP network needs to be trained each time for FEA models with different mesh characteristics. Future studies will address this issue by enhancing the NEP network, enabling it to perform mesh-size independent predictions while minimizing the computational cost. Furthermore, the trade-off between accuracy and dataset generation time should be further investigated. Lastly, future research will focus on exploring the capabilities of DeepFEA in CFD and FSI simulations, as well as in real-life application scenarios, *e.g.*, the evaluation of structural components in the automotive or aerospace industries, or even simulations of human tissues in the context of *in-silico* medicine.

CRediT authorship contribution statement

Georgios Triantafyllou: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing-original draft. **Panagiotis G. Kalozoumis**: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Resources, Software, Supervision, Validation, Visualization, Writing-original draft, Writing- review & editing. **George Dimas**: Conceptualization, Formal analysis, Investigation, Methodology, Resources, Software, Supervision, Validation, Writing-original draft, Writing- review & editing. **Dimitris K. Iakovidis**: Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Writing- review & 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.

Acknowledgements

This work has been funded by the European Union, under grant agreement No 101099145, project SoftReach, (https://softreach.eu/).

Data Availability

The datasets used in this paper can be found at https://doi.org/10.5281/zenodo.10870936.

References

Arcones, D., Meethal, R., Obst, B., & Wüchner, R. (2022). Neural Network-Based Surrogate Models Applied to Fluid-Structure Interaction Problems. *15thWorldCongress Computational Mechanics (WCCM-XV)8thAsianPacificCongress Computational Mechanics (APCOM-VIII)*. https://doi.org/10.23967/wccm-apcom.2022.080

Arzani, A., Wang, J.-X., Sacks, M. S., & Shadden, S. C. (2022). Machine learning for cardiovascular biomechanics modeling: challenges and beyond. *Annals Biomedical Engineering*, *50*(6), 6.

Bender, N. C., Pedersen, H. C., & Andersen, T. O. (2019). Feasibility of deep neural network surrogate models in fluid dynamics. *Modeling,Identification Control*, *40*(2), 2.

Bengio, S., Vinyals, O., Jaitly, N., & Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. *Advances neural information processing systems*, *28*.

Bolandi, H., Sreekumar, G., Li, X., Lajnef, N., & Boddeti, V. N. (2023). Physics informed neural network for dynamic stress prediction. *AppliedIntelligence*, *53*(22), 22.

Burghardt, R., Wächter, M., Masendorf, L., & Esderts, A. (2021). Estimation of elastic–plastic notch strains and stresses using artificial neural networks. *Fatigue&Fracture Engineering Materials & Structures*, *44*(10), 10.

Calzolari, G., & Liu, W. (2021). Deep learning to replace, improve, or aid CFD analysis in built environment applications: A review. *Building Environment*, *206*, 108315.

Chen, G. (2021). Recurrent neural networks (RNNs) learn the constitutive law of viscoelasticity. *ComputationalMechanics*, *67*(3), 3.

Chen, H., Wu, R., Grinspun, E., Zheng, C., & Chen, P. Y. (2023). Implicit neural spatial representations for time-dependent PDEs. *InternationalConference Machine Learning*, 5162–5177.

Chen, Q., Jia, R., & Pang, S. (2021). Deep long short-term memory neural network for accelerated elastoplastic analysis of heterogeneous materials: An integrated data-driven surrogate approach. *CompositeStructures*, *264*, 113688.

Chen, W., Wang, Q., Hesthaven, J. S., & Zhang, C. (2021). Physics-informed machine learning for reduced-order modeling of nonlinear problems. *Journal computational physics*, *446*, 110666.

Chijioke, C., Rodriguez, A., Enriquez, A., Kumar, V., Tandon, V., Terrazas, J., Villanueva, D., & Kotteda, V. K. (2022). FSI of a Cantilever Beam: FVM-FEM and Neural Network Analysis. *FluidsEngineeringDivisionSummerMeeting*, *85833*, V001T03A025.

Cook, R. D., & others. (2007). *Concepts and applications of finite element analysis*. John wiley & sons.

Deshpande, S., Lengiewicz, J., & Bordas, S. P. (2022). Probabilistic deep learning for real-time large deformation simulations. *ComputerMethods Applied MechanicsEngineering*, *398*, 115307.

Du, P., Zhu, X., & Wang, J.-X. (2022). Deep learning-based surrogate model for three-dimensional patient-specific computational fluid dynamics. *Physics Fluids*, *34*(8), 8.

Evans, L. C. (2022). *Partial differential equations* (Vol. 19). American Mathematical Society.

Farajtabar, M., Larimi, M. M., Biglarian, M., Sabour, D., & Miansari, M. (2022). Machine Learning Identification Framework of Hemodynamics of Blood Flow in Patient-Specific Coronary Arteries with Abnormality. *Journal Cardiovascular Translational Research*, 1–16.

Franke, M., & Wagner, M. (2024). Transient surrogate modeling of modally reduced structures with discontinuous loads and damping. *Archive Applied Mechanics*, 1–20.

Frankel, A., Tachida, K., & Jones, R. (2020). Prediction of the evolution of the stress field of polycrystals undergoing elastic-plastic deformation with a hybrid neural network model. *MachineLearningScience Technology*, *1*(3), 3.

Ghavamian, F., & Simone, A. (2019). Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network. *ComputerMethods Applied MechanicsEngineering*, *357*, 112594.

Haghighat, E., Raissi, M., Moure, A., Gomez, H., & Juanes, R. (2021). A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *ComputerMethods Applied MechanicsEngineering*, *379*, 113741.

Haubner, J., Hellan, O., Zeinhofer, M., & Kuchta, M. (2024). Learning mesh motion techniques with application to fluid–structure interaction. *ComputerMethods Applied MechanicsEngineering*, *424*, 116890.

He, J., Kushwaha, S., Park, J., Koric, S., Abueidda, D., & Jasiuk, I. (2024). Sequential deep operator networks (s-deeponet) for predicting full-field solutions under time-dependent loads. *EngineeringApplications Artificial Intelligence*, *127*, 107258.

Hu, L., Zhang, J., Xiang, Y., & Wang, W. (2020). Neural networks-based aerodynamic data modeling: A comprehensive review. *IEEEAccess*, *8*, 90805–90823.

Hughes, T. J. (2012). *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation.

Ibragimova, O., Brahme, A., Muhammad, W., Connolly, D., Lévesque, J., & Inal, K. (2022). A convolutional neural network based crystal plasticity finite element framework to predict localised deformation in metals. *InternationalJournal Plasticity*, *157*, 103374.

Khanolkar, P. M., McComb, C. C., & Basu, S. (2021). Predicting elastic strain fields in defective microstructures using image colorization algorithms. *ComputationalMaterialsScience*, *186*, 110068.

Kim, H., Jeong, I., Cho, H., & Cho, M. (2023). Surrogate model based on data-driven model reduction for inelastic behavior of composite microstructure. *InternationalJournal AeronauticalSpaceSciences*, *24*(3), 3.

Krokos, V., Bui Xuan, V., Bordas, S., Young, P., & Kerfriden, P. (2022). A Bayesian multiscale CNN framework to predict local stress fields in structures with microscale features. *ComputationalMechanics*, *69*(3), 3.

Le, Q. T., & Ooi, C. (2021). Surrogate modeling of fluid dynamics with a multigrid inspired neural network architecture. *MachineLearning Applications*, *6*, 100176.

Leach, L. F., & Henson, R. K. (2007). The use and impact of adjusted R2 effects in published regression research. *MultipleLinearRegressionViewpoints*, *33*(1), 1.

Liang, L., Mao, W., & Sun, W. (2020). A feasibility study of deep learning for predicting hemodynamics of human thoracic aorta. *Journal biomechanics*, *99*, 109544.

Liu, Y.-Z., Ren, S.-F., & Zhao, P.-F. (2022). Application of the deep neural network to predict dynamic responses of stiffened plates subjected to near-field underwater explosion. *OceanEngineering*, *247*, 110537.

Lyu, D., Zhou, R., Lin, C., Prasad, P., & Zhang, L. (2022). Development and validation of a new anisotropic visco-hyperelastic human head finite element model capable of predicting multiple brain injuries. *Frontiers BioengineeringBiotechnology*, *10*, 831595.

Masoumi-Verki, S., Haghighat, F., & Eicker, U. (2022). A review of advances towards efficient reduced-order models (ROM) for predicting urban airflow and pollutant dispersion. *Building Environment*, 108966.

Maurizi, M., Gao, C., & Berto, F. (2022). Predicting stress, strain and deformation fields in materials and structures with graph neural networks. *ScientificReports*, *12*(1), 1.

Mendizabal, A., Márquez-Neila, P., & Cotin, S. (2020). Simulation of hyperelastic materials in real-time using deep learning. *MedicalimageAnalysis*, *59*, 101569.

Mozaffari, M. H., Abdy, H., & Zahiri, S. H. (2016). IPO: an inclined planes system optimization algorithm. *Computing Informatics*, *35*(1), 1.

Pant, P., Doshi, R., Bahl, P., & Barati Farimani, A. (2021). Deep learning for reduced order modelling and efficient temporal evolution of fluid simulations. *Physics Fluids*, *33*(10), 10.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., & others. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances neural information processing systems*, *32*.

Pellicer-Valero, O. J., Rupérez, M. J., Martɪnez-Sanchis, S., & Martɪn-Guerrero, J. D. (2020). Real-time biomechanical modeling of the liver using machine learning models trained on finite element method simulations. *ExpertSystems Applications*, *143*, 113083.

Pfeiffer, M., Riediger, C., Weitz, J., & Speidel, S. (2019). Learning soft tissue behavior of organs for surgical navigation with convolutional neural networks. *Internationaljournal computer assisted radiologySurgery*, *14*(7), 7.

Phellan, R., Hachem, B., Clin, J., Mac-Thiong, J.-M., & Duong, L. (2021). Real-time biomechanics using the finite element method and machine learning: Review and perspective. *MedicalPhysics*, *48*(1), 1.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal Computational physics*, *378*, 686–707.

Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009). GSA: a gravitational search algorithm. *Informationsciences*, *179*(13), 13.

Samadian, D., Muhit, I. B., & Dawood, N. (2024). Application of Data-Driven Surrogate Models in Structural Engineering: A Literature Review. *Archives Computational MethodsEngineering*, 1–50.

Sharma, A., Singh, S., & Ratna, S. (2023). Graph Neural Network Operators: a Review. *MultimediaTools Applications*, 1–24.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., & Woo, W. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Advances neural information processing systems*, *28*.

Ŝolín, P. (2005). *Partial differential equations and the finite element method*. John Wiley & Sons.

Ti, Z., Deng, X. W., & Zhang, M. (2021). Artificial Neural Networks based wake model for power prediction of wind farm. *RenewableEnergy*, *172*, 618–631.

Totounferoush, A., Schumacher, A., & Schulte, M. (2021). Partitioned Deep Learning of Fluid-Structure Interaction. *arXivpreprintarXiv:2105.06785*.

Wang, Y., Oyen, D., Guo, W. (Grace), Mehta, A., Scott, C. B., Panda, N., Fernández-Godino, M. G., Srinivasan, G., & Yue, X. (2021). StressNet - Deep learning to predict stress with fracture propagation in brittle materials. *NpjMaterialsDegradation*, *5*(1), 1. https://doi.org/10.1038/s41529-021-00151-y

Wang, Y., Wang, N., & Ren, Q. (2022). Predicting surface heat flux on complex systems via Conv-LSTM. *CaseStudies Thermal Engineering*, *33*, 101927.

Wriggers, P. (2008). *Nonlinear finite element methods*. Springer Science & Business Media.

Wu, L., Kilingar, N. G., Noels, L., & others. (2020). A recurrent neural network-accelerated multi-scale model for elasto-plastic heterogeneous materials subjected to random cyclic and non-proportional loading paths. *ComputerMethods Applied MechanicsEngineering*, *369*, 113234.

Xiang, L., Lee, C. W., Zikanov, O., Abuhegazy, M., & Poroseva, S. (2023). Reduced Order Modeling of Transport of Infectious Aerosols in Ventilated Rooms. *Available SSRN 4334331*.

Yan, H., Xie, W., Gao, B., Yang, F., & Meng, S. (2023). A spatiotemporal prediction model for rapid prediction of delamination growth in open-hole composite laminates. *CompositesScience Technology*, *235*, 109973.

Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., Deng, M., & Li, H. (2019). T-gcn: A temporal graph convolutional network for traffic prediction. *IEEETransactions Intelligent Transportation Systems*, *21*(9), 9.

Zhao, X., Ru, D., Wang, P., Gan, L., Wu, H., & Zhong, Z. (2021). Fatigue life prediction of a supercritical steam turbine rotor based on neural networks. *EngineeringFailureAnalysis*, *127*, 105435.

Zhao, Y., Dong, S., Jiang, F., & Incecik, A. (2021). Mooring tension prediction based on BP neural network for semi-submersible platform. *OceanEngineering*, *223*, 108714.

Declaration of interests

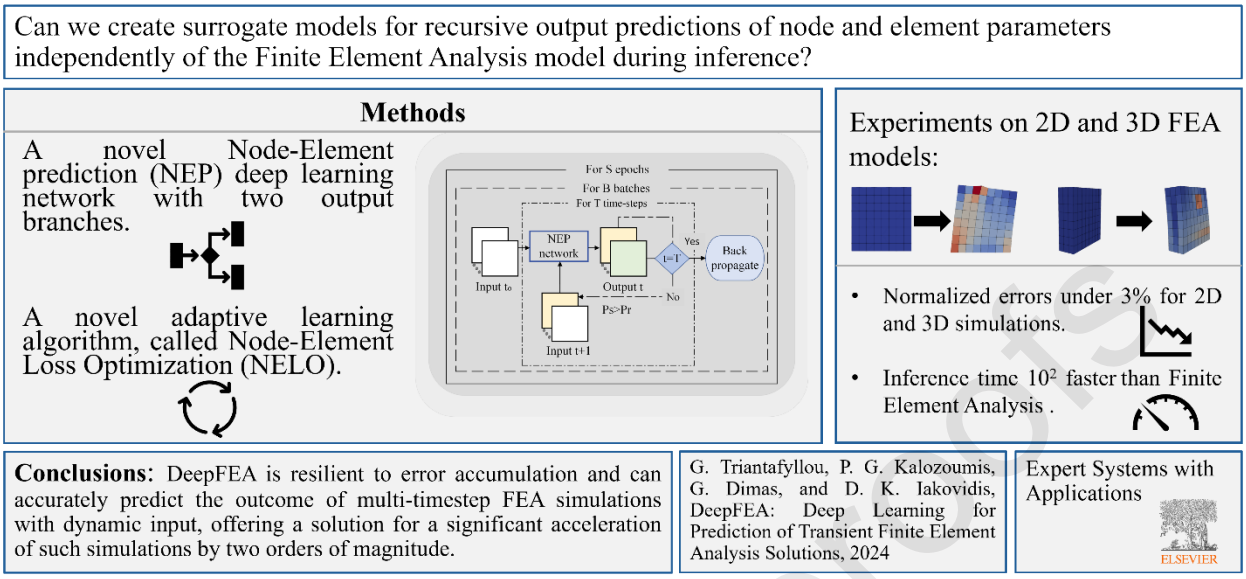
☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

6.1 CRediT authorship contribution statement

Georgios Triantafyllou: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing-original draft. **Panagiotis G. Kalozoumis**: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Resources, Software, Supervision, Validation, Visualization, Writing-original draft, Writing- review & editing. **George Dimas**: Conceptualization, Formal analysis, Investigation, Methodology, Resources, Software, Supervision, Validation, Writing-original draft, Writing- review & editing. **Dimitris K. Iakovidis**: Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Writing- review & editing.

7 Graphical Abstract



Orcid Information

Georgios Triantafyllou : https://orcid.org/0009-0001-2807-9211

Panagiotis G. Kalozoumis : https://orcid.org/0000-0002-0208-301X

George Dimas: https://orcid.org/0000-0002-5770-8988

Dimitris Iakovidis: https://orcid.org/0000-0002-5027-5323