Research paper

Nonlinear analysis of shell structures using image processing and machine learning

M.S. Nashed [a], J. Renno [b,*], M.S. Mohamed [a]

[a] *School of Energy, Geoscience, Infrastructure and Society, Institute for Infrastructure & Environment, Heriot-Watt University, Edinburgh, United Kingdom*
[b] *Department of Mechanical & Industrial Engineering, College of Engineering, Qatar University, Doha, Qatar*

ARTICLE INFO

ABSTRACT

In this paper, we propose a novel approach to solve nonlinear stress analysis problems in shell structures using an image processing technique. In general, such problems in design optimisation or virtual reality applications must be solved repetitively in a short period using direct methods such as nonlinear finite element analysis. Hence, obtaining solutions in real-time using direct methods can quickly become computationally overwhelming. The proposed method in this paper is unique in that it converts the mechanical behaviour of shell structures into images that are then used to train a machine learning algorithm. This is achieved by mapping shell deformations and stresses to a set of images that are used to train a conditional generative adversarial network. The network can then predict the solution of the problem for a varying range of parameters. The proposed approach can be significantly more efficient than training a machine learning algorithm using the raw numerical data. To evaluate the proposed method, two different structures are assessed where the training data is created using nonlinear finite element analysis. Each structure is studied for a varying geometry and a set of material properties. We show that the results of the trained network agree well with the results of the nonlinear finite element analysis. The proposed approach can quickly and accurately predict the mechanical behaviour of the structure using a fraction of the computational cost. All created data and source codes are openly available.

1. Introduction

The finite element method is a useful numerical tool for modelling many engineering applications. The method is especially useful for dealing with complex geometries and/or material heterogeneity. However, it often relies on highly refined mesh grids to provide accurate results. This can be computationally expensive not only for solving problems with many degrees of freedom but also for generating such meshes. The computational needs become even more of an issue when considering finite element models that should produce results in real-time. Running such computations usually involves solving a large system of equations repeatedly several times per second, which can be difficult to achieve even with advanced computational hardware. To deal with this class of applications, a significant amount of computations are performed offline and only necessary computations are updated in real-time [1,2]. This approach can be efficient if the changes in the system can be accurately predicted offline. But accurate predictions are difficult to achieve in many applications related to medical procedures [3,4], interactive virtual environments [1,5], predictive control [6] and

design/process optimisation [7] amongst others. In many such applications, real-time finite element computations must be performed online.

To speed-up online finite element computations, three different approaches are possible [8]: first, it is possible to rely on the constant improvement in the computing hardware as described by Moore's law [9]. This also includes changing the strategy of the hardware utilization [3,5]. The second approach relies on code optimisation and parallel computing [10,11]. The third approach is to develop new modelling algorithms such as mass–spring systems [12,13], model-order reduction [14,15], proper generalized decomposition [16,17], enriched finite element techniques [18] or wave-based finite element methods [19,20]. More recently, machine learning algorithms which are trained using solutions of finite element simulations are also used to solve problems in real-time [21–23].

Solving converged finite element models often provides reliable solutions. However, only models with a moderate number of degrees of freedom can be executed in a real-time manner. Therefore, several attempts were made to alter the way finite element problems are

formulated and solved. In general, the alteration compromises the accuracy of the solution to meet the objective of real-time simulation. For example, ignoring (material or geometric) nonlinearity can significantly reduce the computations and may offer a sufficient level of accuracy based on the application. Huang et al. [24] developed an augmented reality system based on a network of wireless sensors acquiring spatially distributed loads and a finite element model. Real-time computations were achieved by assuming the problem is linear and has a quasi-static behaviour which speeds up the finite element computations to real-time levels. Fiorentino et al. [25] used a similar approach on a cantilever specimen. The results of the finite element simulation were displayed in real-time on the specimen where the boundary conditions were interactively defined/redefined. However, in many applications it is not possible to assume linearity.

Biomedical engineering is a main area where most of the applications cannot be treated as linear. Examples can be found in studies that involve modelling soft tissues such as skin [26,27], internal organs [28, 29], muscles [30,31], and brain tissues [32]. Hence, the finite element analysis (FEA) of this type of applications often cannot be performed in real-time. To overcome this, quasi-nonlinear models were developed by combining a linear-elastic finite element model with a nonlinear one with the nonlinear model being limited to certain parts of the computational domain. Examples of such approach can be found in [14, 33–35]. In general, this approach ignores certain nonlinear effects. Similarly, the approaches that rely on model-order reduction are, in general, not applicable to nonlinear problems [36,37].

Compared to the previous approaches, machine learning algorithms are a strong candidate to overcome the above difficulties. The work on using artificial neural networks (ANN) as an alternative to the FEA has been ongoing since the early 1990s [38]. However, at the early stages ANNs had serious limitations which resulted in severe restrictions on where they can be useful. These restrictions were eased significantly by the recent development of new approaches such as deep learning and physics informed neural networks (PINNs) [39,40]. The popularity of deep learning and PINNs has been on the rise. For example, Karniadakis et al. [41] published an extensive review of physics-informed machine learning approaches. They provided a review of existing capabilities and outlook of this field. Alber et al. [42] provided a comprehensive review of integrating multiscale modelling and machine learning.

To reduce the computational cost of FEA when solving multi-scale problems, machine learning surrogate models were nested in a micromodel, which significantly reduced the computational costs [26]. Other results showed that PINNs can simulate the response of an elastic bar with good accuracy and much faster than standard computational methods [43]. Deep neural networks were also used to evaluate the buckling response of aircraft panels [44] where the neural network estimated the response with 95% accuracy compared to the FEA but using only 1/200 of the central processing unit (CPU) time. Deep learning was also used with higher-order neurons to predict the strength of foamed concrete [45]. Other cases focused on producing more accurate results using ANNs but with reasonable computational cost [46, 47]. More recently, the introduction of PINNs led to numerical models that are driven by data as well as physics and potentially combining the advantages of both approaches. Here, the neural network was used to solve partial differential equations. For example in [48], a Python package for physics-informed deep learning was developed to solve partial differential equations in strong and weak forms. PINNs were also implemented to build surrogate models simulating nonlinear von Mises elastoplasticity in solid materials [49]. Wen et al. [50] combined PINNs with FEA to model complex deformations in multiscale problems. Combining PINNs with FEA is not limited to mechanical problems. For example, PINNs can be used to accurately predict convective heat transfer [51].

Another popular type of neural networks which is used by engineers for solving problems in mechanical engineering is convolutional neural networks (CNNs). This type of networks is composed of convolutional layers, fully connected layers, and pooling layers. The layers are stacked successively with additional activation functions at key points to form a complete CNN [52]. CNNs were initially introduced for image processing in applications related to machine vision [53–55], autonomous vehicles [56,57] and image segmentation [58,59]. More recently, CNNs were also introduced into mechanical applications. The finite element method and CNNs were combined within the same framework to solve optimisation problems for annular shape charge [60]. The network was trained using finite element simulations and the resulting network was then used to predict the optimal parameters of an annular liner. The presented numerical simulations and the experimental results indicated that the CNNs have indeed produced an optimum solution for the annular shape charge problem. Krokos et al. [61] considered multiscale computational mechanics problems where they used an encoder-decoder CNN to generate the stress distribution of the microscale features in their model. They used a Bayesian approach to evaluate the uncertainty of the predictions. Desphande et al. [62] used three benchmark finite element models to generate training datasets. Loads were applied randomly on each model and the displacement results were recorded and used in the training of Bayesian CNNs. The loads were large enough to cause nonlinear deformations. For a more in-depth and comprehensive discussion of CNNs, the interested reader is directed to [63]. Generative adversarial networks (GANs) are a special form of CNNs that are often utilized in the creation of images [64]. The main principle of GANs is that two rival CNNs compete in a way that both networks become better as the training progresses. These two CNNs are called the Generator and Discriminator. The Generator creates images that it aims to pass to the Discriminator as genuine whereas the Discriminator's role is to identify whether the generated images are real or fake. To train the Discriminator, input images are provided as pairs with each pair comprising of a genuine image (from the training dataset) and a generated image (created by the Generator). The Generator is trained to create images that match the ground-truth whereas the Discriminator is trained to differentiate between the ground-truth and generated images. The performance of both networks improves as training progresses [64]. In computational fluid dynamics, a GAN was used to develop a new sub-filter modelling approach in the context of large-eddy simulations [65]. The network performed well with decaying turbulence calculation and in complex combustion scenarios which otherwise would have been very challenging for conventional numerical methods.

In this paper, we propose a new machine learning approach for solving nonlinear computational mechanics in shell structures using a conditional Generative Adversarial Network (cGAN) as an image processing technique. A cGAN is trained to map input images to output images and learn the loss function to train the cGAN. Thus, defining a loss function is not required when using cGANs which minimizes human input in the process of building the cGAN and makes the method more general allowing its application to a variety of image datasets (e.g., transforming hand-drawn sketches into photos, etc.) [64]. Further information about the cGAN is provided in Section 2.2.

To train the machine learning algorithm, a set of data is created using nonlinear FEA models. The results of the finite element model along with the material and geometric properties are embedded into images that are then used to train the cGAN. The red-green-blue (RGB) images that are created are not contour plots of the solution of the FEA simulation. Instead, every RGB image includes the solution of the FEA simulation (displacement or stress) in the red channel whereas geometric and material properties are included in the green and blue channels of the image. The transformation of the data into images makes the problem suitable for emulating FEA by cGANs. The proposed approach has a significant computational advantage in treating the mechanical behaviour as an image processing problem. Trying to recognize the visual patterns in the finite element results and then mimicking them is more efficient compared to the process of dealing with a significant amount of raw data that is typically produced in such simulations. Three cGANs are designed where a dedicated network is

used to identify deflections by reading forces, stresses by reading deflections and stresses by reading forces. Thus, the contribution of this paper lies in transforming the solution of a nonlinear FEA problem into an image dataset that can be used to train a cGAN which is then used to emulate FEA. Geometric and material properties are embedded in the image dataset. The resulting cGAN can produce near real-time results for nonlinear FEA problems.

The remainder of this paper is organized as follows. Machine learning and ANNs will be briefly reviewed in Section 2. Section 3 lays out the proposed approach and how the cGAN is built, trained, and validated. Two examples are then presented to assess the performance of the proposed method in Section 4 while conclusions are drawn in Section 5.

2. Machine learning and artificial neural networks

Artificial neural networks (ANNs) are a class of computational tools inspired by the biological nervous system [66]. The main processing unit is called the neuron or perceptron. The neurons are organized in sets of layers. The network usually starts with an input layer and ends with an output layer. The specific layout to which the neurons and layers are interconnected is known as the "connectivity pattern" or "architecture" of the network. The network is trained to learn a certain data pattern by going through training cycles (or epochs). During each epoch, each neuron receives input signals from other units yielding an output. Throughout the training process, the weights (w) and biases (b) of the neurons are affected by the flow of data through the network layers. Then, the values carried by the neurons are aggregated by various 'activation functions' resulting in the 'activation' value (a) of that neuron. The neurons' activation values are then transmitted as inputs to the next connected layer. The network accuracy is calculated by a backpropagation algorithm during the training process which leads to the network convergence toward a solution. This process makes the ANN capable of learning linear and nonlinear relations, hence allowing it to solve many problems instead of following a predefined set of rules.

2.1. Convolutional neural networks

Convolutional neural networks (CNNs) are a specific type of ANNs that emerged from the study of the human brain visual cortex for image recognition purposes. CNNs showed a great ability to solve difficult image-based tasks in pattern recognition, classification and machine vision in general [54,67] as well as in traffic prediction and autonomous vehicle applications [57,68]. A CNN is a collection of convolutional layers, fully connected layers, and pooling layers. Convolutional layers form the most important block in a CNN. These layers are not fully connected layers because each neuron is only connected to a few other neurons located within a small rectangular area in the previous layer. This architecture allows the network to focus on small low-level features in the first hidden layer, then assemble them into larger higher-level features in the next hidden layer and so on. The pooling layers are like the convolutional layers in that they are not fully connected. Each neuron in a pooling layer is connected to the outputs of a limited number of neurons in the previous layer. However, a pooling neuron just aggregates the inputs using an aggregation function such as the max or mean. This process allows the network to reduce memory usage and computations. Finally, the fully connected layers in conjunction with the activation functions produce scores to be used for classification.

2.2. Generative adversarial network

The generative adversarial network (GAN) is a specific type of CNN which was initially designed for image creation and enhancement. It was first proposed in 2014 by Goodfellow et al. [69]. Although the idea of GAN showed great potential almost instantly, it took some time to overcome some of the difficulties in its training. The core principle of

GAN relies on the competition between two rival networks; a generator and a discriminator; hence their "adversarial" nature. The generator is tasked with creating an image which it aims to pass off as genuine to the discriminator. The discriminator in turn is tasked with identifying whether the images it receives from the generator are in fact "real" or "fake". The discriminator always works with two images during the training phase. One image comes from the generator and the other is a genuine image from the training dataset. After assessing the level to which it believes the generator image is real or fake via loss functions, the results are backpropagated for each network, i.e., for the generator and the discriminator, and the cycle is repeated. This competitive loop between the generator and the discriminator improves both networks, but primarily makes the generator capable of producing near-ground-truth images. A subtype of GANs is the conditional generative adversarial network (cGAN). This network was developed by Isola et al. [64] with their seminal "Pix2Pix" model. In principle, the cGAN and the GAN are based on the same fundamentals. However, instead of generating an image from random input noise and discerning whether it is passable as genuine or not as in the GAN, the cGAN enforces the conditional generation and discerning of the images using auxiliary information (such as class labels or data). Isola et al. [64] demonstrated that the Pix2Pix method can be utilized on a broad range of datasets for a variety of tasks such as architectural labels transformed into building images and maps transformed to aerial photos and colorization of black and white images. Since the image generation is conditional the convergence of the cGAN is faster than the convergence of GAN. Moreover, unlike conventional CNNs, there is no need to define a loss function when using cGANs. The Pix2Pix method learns the loss function without human input thereby providing a "universal" loss function. This makes cGAN a more powerful tool and a potential candidate to emulate FEA.

3. Proposed method

We propose to build a machine learning approach based on the cGAN that can successfully predict the mechanical behaviour of a structure for a variation of loads and material properties. The approach consists of multiple stages which are summarized in the swimlane diagram of Fig. 1. Each stage is explained in detail in the following sections.

3.1. Data creation

In this section, two case studies will be presented. In both cases, a bilinear material model is used, and multiple vertical loads are randomly applied to the structure which experiences nonlinear deformations. The first case study considers the vertical deformation of a square plate whereas the second case study considers the vertical displacement of a curved shell.

3.1.1. Plate

In this case study, we focus on the nonlinear response of a square plate under varying localized pressure loads applied at multiple locations where the magnitudes of the loads and their locations are randomly varied. The four edges of the plate are fully fixed. The choice of this structure is deliberate where a large area with numerous input-output coordinates can be rapidly produced and since a closed form solution for this problem does not exist. The data is generated using a finite element model with SHELL181 elements in ANSYS [70]. A uniform mesh of 256×256 elements is used to model the plate. A schematic of the plate and an example of the applied loads is shown in Fig. 2.

Six different values of the modulus of elasticity (60, 90, 120, 150, 180 and 210 GPa) and eight values of the plate thickness (20, 40, 60, 80, 100, 150, 200 and 300 mm) are considered. This results in a total of 48 possible physical realizations of the plate. Each realization starts with simulating the response to one (randomly located) load, then two (randomly located) and so on until five (randomly located) loads are
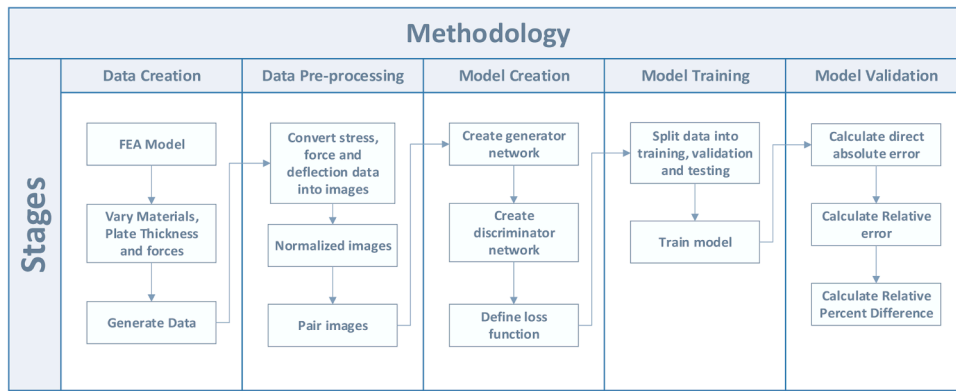
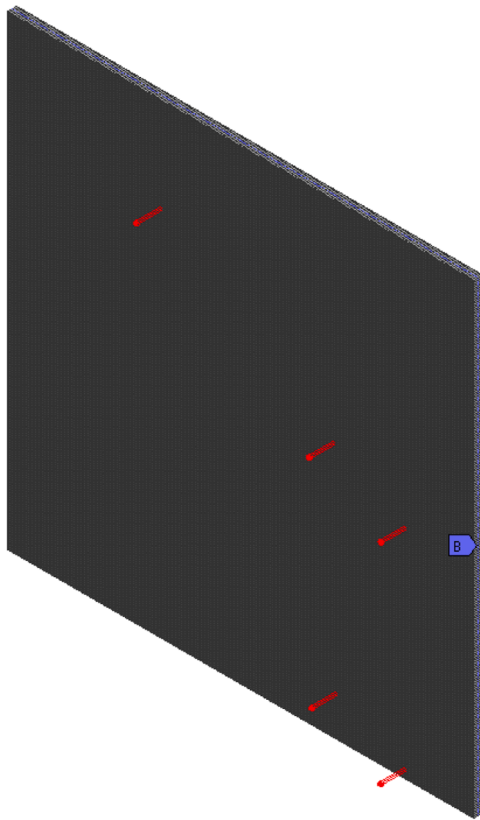**Fig. 1.** Swimlane diagram summarizing the methodology.

Fig. 2. Schematic of a rectangular plate with five (randomly located) loads. The four edges of the plate (annotated by B in the schematic) are fully fixed.

acting on the plate. Since the plate is fully fixed, the loads are applied within 5% of the edges of the plate.

The value of the load that would cause nonlinear deformations in the plate depends on the modulus of elasticity and the thickness of the plate. For example, Table 1 shows such loads for eight realizations of the plate (with one value for the modulus of elasticity). Similar tables were generated for the remaining 40 realizations. The value of the transverse force F that is displayed in Table 1 were obtained with guidance from [71] and was then validated using FEA to make sure that this load will induce nonlinear deformations in the plate. For each realization in Table 1, ten more values for the vertical load were calculated. Thus, for each realization of the plate, five loads of those shown in Table 1 are randomly selected, sequentially applied to the load at random locations and the simulation is conducted. Thus, the choice of the load F ensures that the plate will undergo nonlinear deformations in each of the five simulations.

For each realization, the five simulations are repeated 25 times as "trial cycles" to generate randomness and ensure a wide statistical spread in the dataset. Hence, each realization with one modulus of elasticity and plate thickness will generate 125 data samples. With 48 physical realizations of the plate, the total number of simulations comes up to $125 \times 48 = 6,000$. The plate deflections as well as the first principal stress are evaluated for all the considered simulations and the results are stored in the dataset needed for training and validating the cGAN. The generation of this dataset is computationally intensive but can be performed in parallel on multiple machines. Fig. 3 below shows a sample of both stress and vertical displacement results from one of the simulated finite element models.

3.1.2. Curved shell

In this case study, the nonlinear response of a curved shell to randomly located vertical loads is considered. Fig. 4 below shows a schematic of the curved shell. The shell is fixed at the edge where $x = 0$ and at the edge where $x = 1$ m; the other two edges are free. The width (along the y-axis) is 1 m, and the radius of the shell is 1 m. The shell has the same material properties as the plate. Six values of the modulus of elasticity (60, 90, 120, 150, 180 and 210 GPa) and eight values of the plate thickness (20, 40, 60, 80, 100, 150, 200 and 300 mm) are considered. The combinations of the modulus of elasticity and thickness provide 48 possible physical realizations. For each realization, loads are

Table 1
Possible value of the load that would cause the plate whose modulus of elasticity is 60 GPa to experience nonlinear deformations in the traverse direction.

Thickness [mm]	F [kN]	1.01F	1.02F	1.03F	1.04F	1.05F	1.06F	1.07F	1.08F	1.09F	1.10F
20.0	13.2	13.3	13.5	13.6	13.7	13.9	14.0	14.1	14.3	14.4	14.5
40.0	52.0	52.5	53.0	53.6	54.1	54.6	55.1	55.6	56.2	56.7	57.2
60.0	130.0	131.3	132.6	133.9	135.2	136.5	137.8	139.1	140.4	141.7	143.0
80.0	245.0	247.5	249.9	252.4	254.8	257.3	259.7	262.2	264.6	267.1	269.5
100.0	380.0	383.8	387.6	391.4	395.2	399.0	402.8	406.6	410.4	414.2	418.0
150.0	730.0	737.3	744.6	751.9	759.2	766.5	773.8	781.1	788.4	795.7	803.0
200.0	1300.0	1313.0	1326.1	1339.4	1352.8	1366.3	1380.0	1393.8	1407.7	1421.8	1430.0
300.0	2900.0	2929.0	2958.0	2987.0	3016.0	3045.0	3074.0	3103.0	3132.0	3161.0	3190.0

(a) Displacement in the vertical direction in mm.
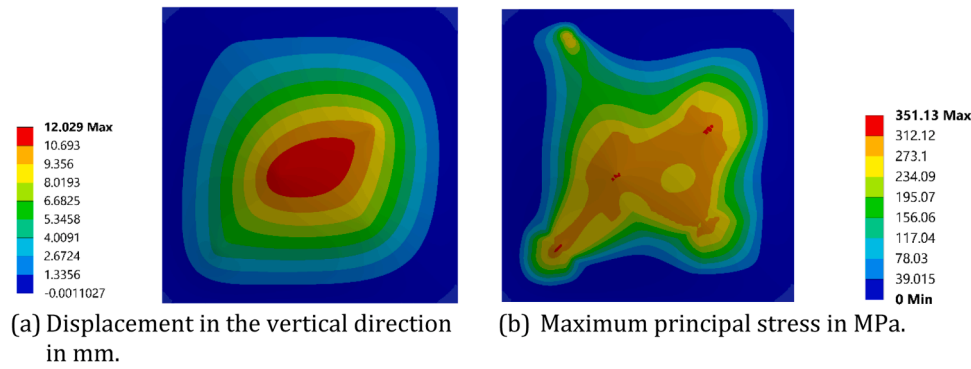

(b) Maximum principal stress in MPa.

Fig. 3. A sample of thin plate analysis results in response to five randomly located loads.
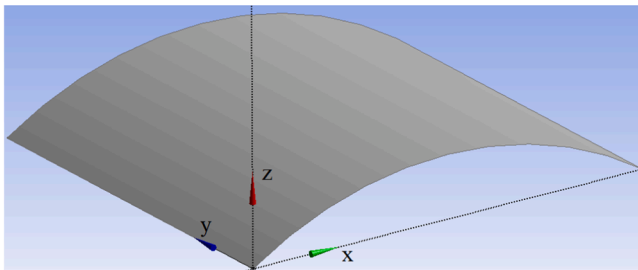

Fig. 4. A schematic of the curved shell considered in this case study.

added one at a time staring with one (randomly located) load until five (randomly located) loads are applied. The loads are all acting in the negative z-direction. The load values were obtained in similar fashion as laid out in Section 3.1.1. For each realization, the load with the lowest value would cause nonlinear deformation in the shell. Ten loads were defined like Table 1 and the loads that are applied sequentially are selected randomly from these ten loads. All the loads that were used in this case study are provided as part of the data associated with this paper (which is openly available). Each simulation is repeated 25 times to generate randomness and ensure that the dataset has a wide statistical spread. Like in the first case study, the dataset will be formed from the results of 6000 simulations. Vertical dispalcement and the first principal stress are saved after each simulation to be later used in the training and validation of the cGAN.

3.2. Data pre-processing

To train the cGANs, the dataset was organized as paired images. One image as an input and the other as the target. Rasterized images can be used in any of the following popular formats: Bitmap (bmp), Portable Network Graphics (png) or Joint Photographic Experts Group (jpeg). The images are composed of a matrix of pixels each with a minimum of three channels. Those channels represent red, green, and blue colours, i. e., the RGB format. The granularity of a pixel in each channel is administered by the bit-depth which can be 8 bits for 256, 16 bits for 65,536 or 24 bits for 16,777,216 shades of each colour. It should be noted that usually for photography purposes, 24 bits are used as the human eye do not easily distinguish between colours beyond this level of shadings. However, it is possible to create higher bit-depth for industrial uses. Obviously, a higher bit-depth results in a larger image file and subsequently increases the size of the dataset. After careful consideration, we decided to use images with depth of 8 bits as this provided sufficiently accurate results for a significant reduction in the CPU time which was otherwise used to train the cGAN. Furthermore, the image format may cause compression artefacts. Although this might not be visible to the human eye, such artefacts can damage the data or even corrupt it. A comparison of high-quality image formats is summarized in

Table 2

Comparison of common image file formats recreated from [72].

Parameter	GIF89a	JPEG	TIFF	PNG
Maximum Colour Depth	8-bit	24-bit	48-bit	48-bit
Number of Colours	256 colours	16 million	281 trillion	281 trillion
Compression Technique	Lossless	Lossy	Lossy	Lossless
Gamma correction	No	No	Yes	Yes
Patent Issues	Yes	No	Yes	No

Table 2. In this work we use png file format as it is a lossless data compression format.

To maximize the amount of information passed to the cGAN, the following parameters: elasticity modulus, plate thickness, load location, load magnitude, nodal deflections, and nodal stresses, have been transferred to the images. The red channel in each image is assigned to the magnitude (applied force, stress, or deflection), green to the modulus of elasticity and blue to the plate thickness. This process results in three images for each sample of results. Fig. 5 shows a sample of paired images decomposed into the three channels red, green, and blue. The target image is obtained by transforming the results of the FEA simulation and the material and geometric properties as explained above.

3.2.1. Data normalization

Converting the decimal results of the finite element simulation into an image requires scaling the data to fit within the colour depth range. Therefore, each input variable is normalized to the range [0,1]. Then, the normalized values are multiplied by the bit-depth of each channel, i. e., 255 (since the bit-depth is 8 bits and the channel colour range is [0, 255]). Clearly, the cGAN cannot distinguish the amplitude difference between different input variables. The resultant is a three-dimensional colour-space of potential input values mapped to pixel colour as illustrated with an example of the two-dimensional image files in Fig. 6 where a sample of paired image shows the deflection as an input and the corresponding stresses as an output.

3.3. Model creation

The cGAN which is implemented in this paper follows the working principles and the architecture of the "Pix2Pix" network developed by Isola et al. [64]. Like the GAN, the cGAN trains the generator to produce realistic samples for each class of data in the training dataset. The discriminator on the other hand learns to distinguish fake samples from real ones. However, unlike GAN, the discriminator in the cGAN does not learn to identify the class of the image. It learns only to accept real, matching samples while rejecting samples that are mismatched. Moreover, for the generator to fool the discriminator, noise is used to synthesize fake samples. Next, we discuss the main components of the cGAN.
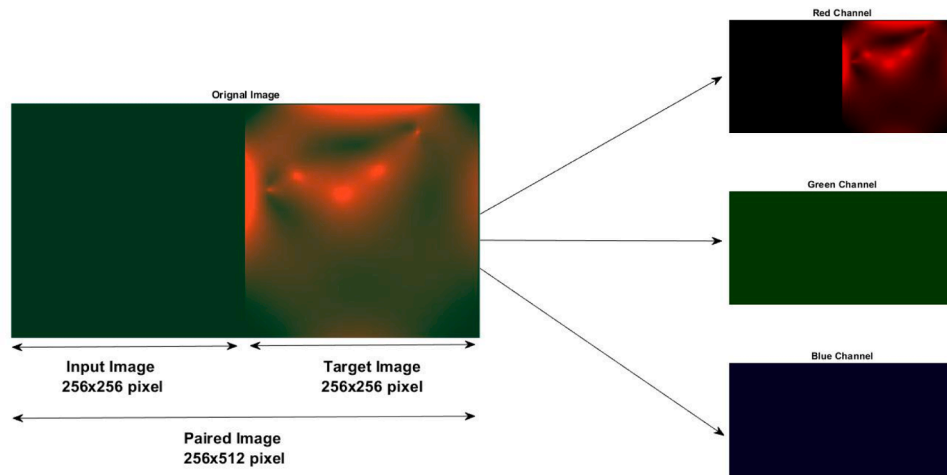
Fig. 5. Colour channel decomposition for a single sample of 256 × 512 png paired image dataset reflecting the colour channel mapping.
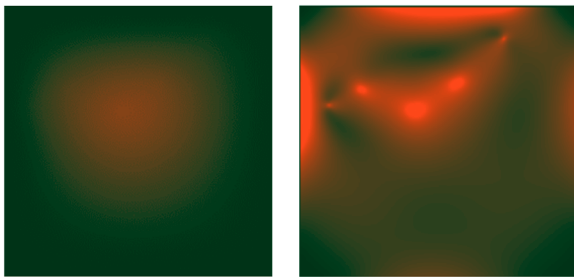

Fig. 6. Example of the processed paired image dataset: deflection as an input (left) and the corresponding stress as an output (right).

3.3.1. Generator network

The objective of the generator in a cGAN is to produce samples that capture the characteristics of the training dataset and look indistinguishable from the training data. The generator learns the patterns in the dataset to distinguish the dataset content. Then instead of recognizing the patterns, the generator learns to create the dataset from scratch; this takes the form of an auto encoder/decoder whereby the encoder portion is tasked with mapping the input space to another intermediate space (sometimes called a "latent space"). Thereafter and by contrast, the decoder has the complementary function of mapping from the latent space to another target space. The Pix2Pix implementation used in this research has additional bridging connections between the layers of the encoder and layers of the decoder to form a "U-Net". This "cross-linking" provides the ability to pass information across the network whilst avoiding the central bottleneck region inherent in the conventional encoder/decoder networks and functionally imposes structural conditions between layers. Fig. 7 shows a schematic of the generator network.

3.3.2. Discriminator network

The cGAN discriminator network is a CNN which Isola et al. [64] refer to as a "patchGAN" classifier. With patchGan, the discriminator divides each image into overlapping "patches". Then, the discriminator evaluates if each patch is real or fake. The output of the discriminator is a matrix with predicted probability for each patch. This process allows the network to measure the goodness of the discriminator at distinguishing images based on their pattern rather than their content.

3.3.3. Loss functions and gradients

The objective function of the cGAN can be stated as follows

$$\mathscr{L}_{cGAN}(\boldsymbol{G}, \boldsymbol{D}) = E_{x,y}[\boldsymbol{logD}(\boldsymbol{x}, \boldsymbol{y})] + E_{x,z}[\boldsymbol{log}(1 - \boldsymbol{D}(\boldsymbol{x}, \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{z}))] \tag{1}$$

where the generator (G) is trained to map the real image x and a random noise vector z to an output image y. On the other hand, the discriminator (D) is trained to classify between the "real" (x) and "fake" (y) images. The generator tries to minimize the objective function while the discriminator (which acts as the adversarial network) tries to maximize the same objective, i.e.,
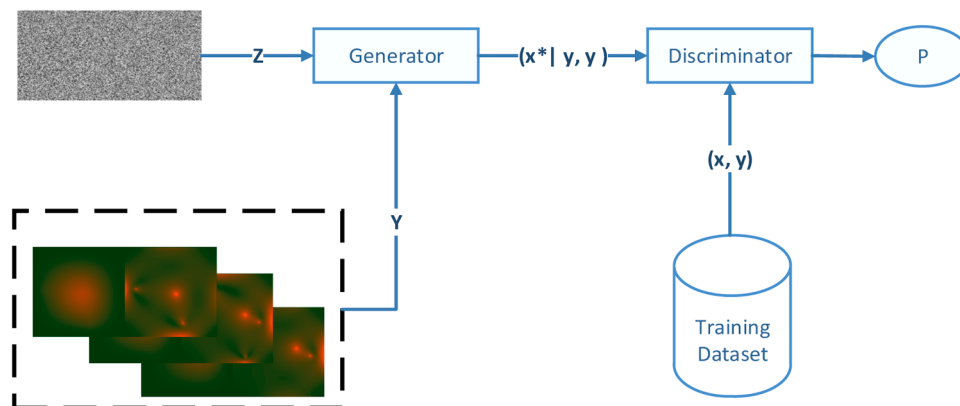

Fig. 7. cGAN generator uses a random noise vector z and a label y (one of them possible labels) as inputs and produces a fake example ($x^*|y$) that strives to be both realistic looking and a convincing match for the label y.

$$G^* = \underset{G}{argmin}\underset{D}{max} \mathcal{L}_{cGAN}(G, D) \tag{2}$$

Additionally, the cGAN includes the mean absolute error loss, L_1, imposed on the generator to be near the ground truth output (in an L_1 sense) in addition to the conventional requirement of convincing the discriminator that the output is real. This L_1 requirement can be stated as

$$\mathcal{L}_{L1}(G) = E_{x,y,z}[\parallel y - G(x, z) \parallel_1] \tag{3}$$

The final objective of the cGAN can then be summarized as

$$G^* = \underset{G}{argmin}\underset{D}{max} \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G) \tag{4}$$

In practice, calculating the losses and utilizing them to change the network weights and biases is done sequentially as laid out in [73].

3.4. Model training

The results of the finite element simulation were organized into three datasets: loads, deflections, and stresses. Since the stresses and the deflections are generated by the applied loads, the force dataset is paired with both the deflections and the stresses datasets to create two experiments. In the first experiment, we train the cGAN to identify the structure's deflections by reading the forces, i.e., a force-to-deflection (FtD) mapping. In the second experiment, we train the cGAN to identify the structure's stresses by reading the forces, i.e., a force-to-stresses mapping (FtS). Furthermore, we also created a third pair by coupling the deflections with the stresses to create a deflection-to-stress (DtS) mapping. In this work, the cGAN are implemented in MATLAB [74]. Two thirds of the dataset was used for training while the remaining third was equally split between testing and validation sets. Hence, 4000 images were used for training, 1000 images were used for testing and 1000 images were used for validation. Random jitter and random flip functions were used to introduce additional randomness and discourage over-fitting [64]. The losses and gradients were determined and applied to both the generator and discriminator networks using the ADAM solver and backpropagation function [73].

3.5. Model validation

GAN in general (and cGAN in particular) are difficult to train for several reasons. For example, a very good discriminator may result in a poor generator with a vanishing gradient problem. In this case the discriminator does not provide sufficient feedback to the generator to improve its outputs. Another problem happens when the generator produces the same output repeatedly leading the discriminator to always reject the output, which is known as the mode collapse problem. Consequently, the discriminator gets stuck in a local minimum and does not find the optimal hyperparameters of the cGAN. Because the accuracy of prediction is paramount, we validate each generated result against the ground-truth (obtained from the FEA simulations) by several metrics. We use the direct absolute error (DAE) to evaluate the absolute value of the difference between the target and generated results; this is given by

$$DAE = \left| y_{target} - y_{generated} \right| \tag{5}$$

where y_{target} is the ground-truth obtained through the FEA and $y_{generated}$ is the prediction generated by the cGAN. We also use the relative error (RE) which is the ratio of the DAE magnitude to the target value magnitude, and is defined by

$$RE = \frac{\left| y_{target} - y_{generated} \right|}{y_{target}} = \frac{DAE}{y_{target}} \tag{6}$$

The third metric we use is the relative percent difference (RPD) which estimates the variation between the target and the generated results

$$RPD = \frac{1}{2} \frac{\left| y_{target} - y_{generated} \right|}{(y_{target} + y_{generated})} = \frac{1}{2} \frac{DAE}{(y_{target} + y_{generated})} \tag{7}$$

The latter is an important metric as there might be regions wherein the target and the generated results have zero values (e.g., at fixed edges) where the DAE and the RE are undefined. All the above metrics are calculated for the overall images (which results in 2D plots). We also calculate the averages of these metrics over the entire result and at the locations of the maximum input values. Furthermore, the errors are calculated and tracked across every saved epoch of the model to gauge the improvements in the training of the cGAN, hence, to assess the changes.

4. Results and discussion

This section discusses the numerical results of the three primary experiments, namely, FtS, FtD and DtS mapping for the two case studies that were described in Section 3.1. The cGAN architecture explained in Section 3.3 is used to obtain all the following results.

4.1. Training progress

The training for each experiment is run for 200 epochs with a batch size of 32 samples each so that all the 4000 training samples are organized into 125 batches resulting into a total of 25,000 iterations to complete the training. Fig. 8 shows snippets of the training progress for the FtS experiment for the rectangular plate case study. The figure includes the results of the generator, the discriminator and the cGAN. The snippets show that the score of the discriminator rises whilst conversely, that of the generator decreases starting from around the 150th to the 550th iterations. This is an indication that the discriminator can assess the generated result as a "fake" up to this point. However, around the 5,900th iteration the score for both the generator and the discriminator jumps between 0 and 1. The losses for both networks also increase in the same range. These observations indicate that the discriminator is having difficulty in distinguishing the generator images as fake and that the generator is producing believable results. From here onward, both networks advance in steps with one another, and the learning changes are reflected in the fine details of the snippets. The eventual alignment of the generator losses with that of the discriminator indicates the improvement in fooling the discriminator, hence, the improved prediction accuracy. Similar behaviour is observed in the FtD and DtS experiments of the rectangular plate and in all the experiments of the curved shell.

The training is conducted for 200 epochs with the model being saved at a checkpoint every four epochs. Hence, an array of 51 checkpoints is produced for each of the samples. Fig. 9 shows the output of a randomly selected sample from the validation dataset at each of the saved checkpoints. Fig. 9 shows the gradual improvement of the predictions as the cGAN's training progresses. Again, similar patterns are found when training for the FtD and DtS mappings of the plate case and for all the experiments of the curved shell.

4.2. Validation results

For validating the FtS case, 2000 paired images were used. Each pair of images consisted of the force image as an input and the stress image as a target. A randomly selected input image from the validation dataset is used to compare against the ground-truth using Eq. (5) through to Eq. (7). The results in Fig. 10 show the prediction of the generator at the first checkpoint (4th epoch) and at the last checkpoint (200th epoch). The absolute error plot shows a significant reduction in the error from around 50% at the first checkpoint to less than 10% at the last checkpoint. Furthermore, the maximum RE throughout the plate is at the first checkpoint, while the maximum RE at the last checkpoint is near the fixed edges of the plate where the solution is almost zero. This justifies
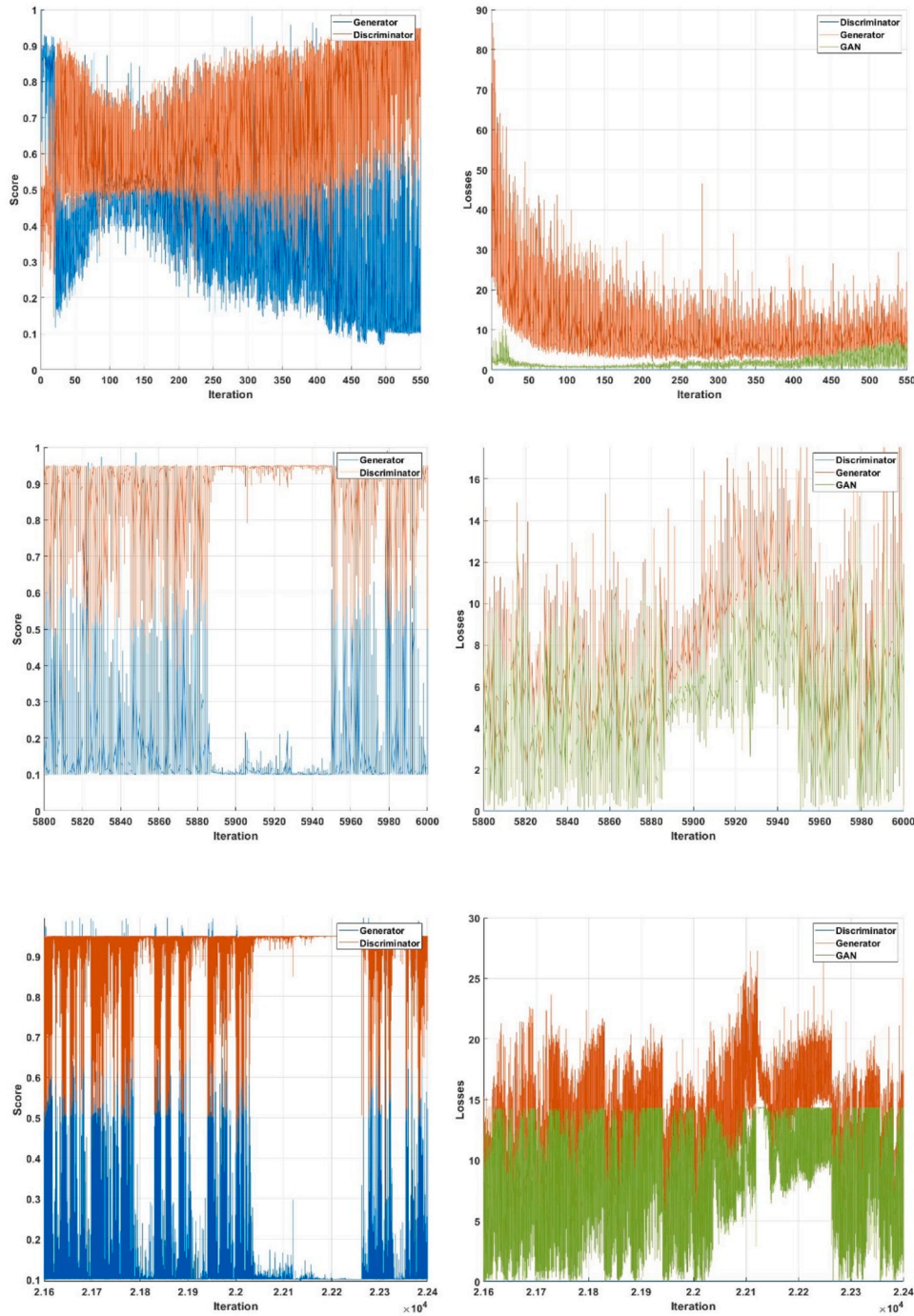
Fig. 8. FtS Model training progress snippets of the scores, losses, and random validation output for: the first 550 iterations (top), between 5800 and 6000 iterations (middle) and between 21,600 and 22,400 iterations (bottom).

the relatively high RE whereas the DAE is much smaller at the fixed edges.

Next, we illustrate in Fig. 11 through Fig. 14 the DAE and RPD errors for the full validation dataset (2000 samples), where both the maximum and the mean value of the DAE and RPD errors are shown. The aim here is to show the errors for each sample in the validation dataset, the error in the maximum stress and the mean value of the error for all the pixels. Fig. 11 shows the kernel density distribution of the DAE error at the location of the maximum stress and the mean value of the DAE error over all training epochs and for all the validations samples. It is possible to see that the best performance is in general not at the end of the training stage (i.e., not at the 200th epoch). The figure also shows that

the mean value of both distributions (DAE target and DAE mean) is below 5% for all training epochs. Fig. 12 presents the cascade for the DAE mean over all epochs and is consistent with Fig. 11.

Figs. 13 and 14 present similar results to those presented in Figs. 11 and 12 but for RPD errors. These figures are consistent with the previous ones. However, in this case the mean value of both distributions, i.e., RPD at the maximum stress and RPD mean are below 20% for all training epochs as shown in the figures.

It should be noted that the choice of 200 epochs is made to ensure that the network converges. However, this choice does not mean the network will find the best solution at the 200th epoch. The network may find an optimal solution before going through the 200 epochs and then
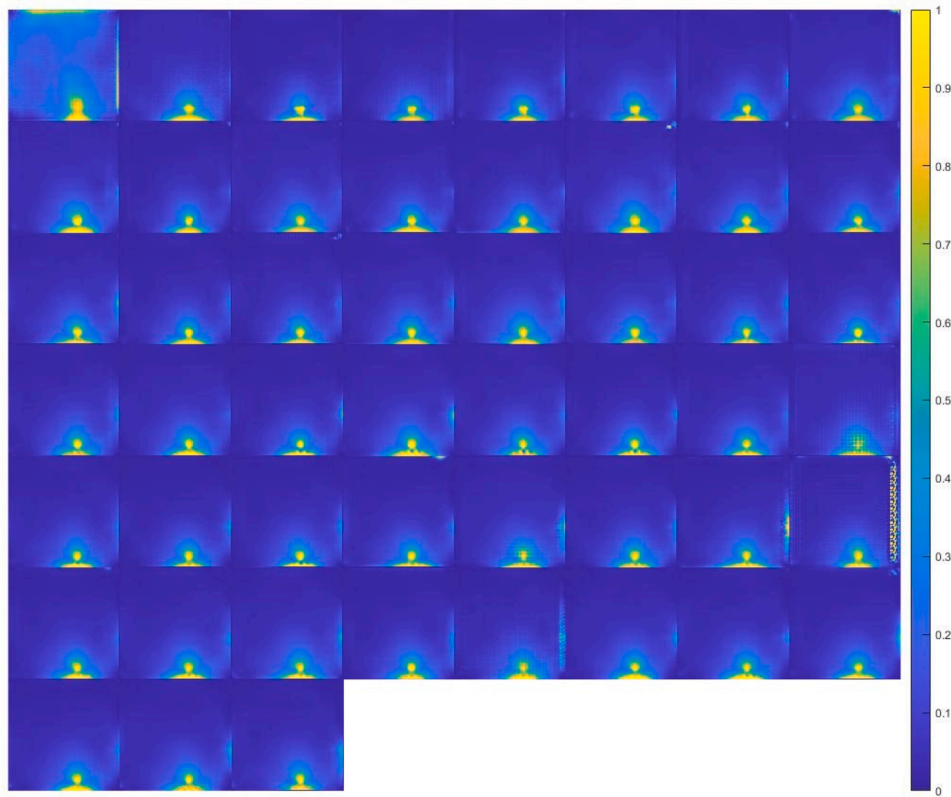
Fig. 9. Sequence of generated predictions for FtS Mapping of the plate: results at 51 checkpoints where finer details and smoother gradients emerge with continued training running from left to right and top to bottom.
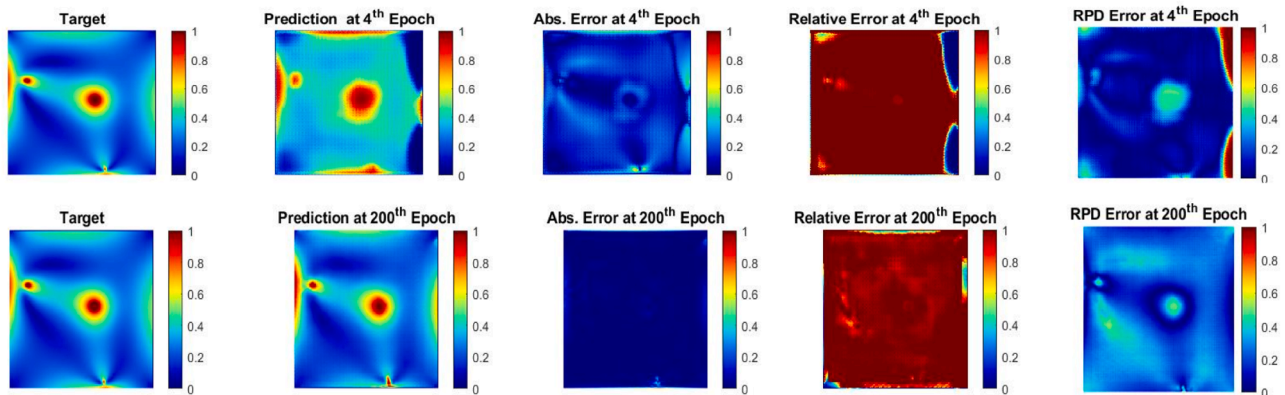

Fig. 10. FtS validation results of the error metrics at the first checkpoint (top) and the last checkpoint (bottom).

start over-fitting the data afterwards. Indeed, Fig. 13, shows that the network reaches its best solution after around 136 epochs of training. Thereafter, the network starts over fitting the data which leads to poorer performance at later epochs. To deal with such cases an early stopping technique [75] was adopted. The technique is a regularizing machine learning approach that allows to save and restore a network after it reaches an optimal solution.

Fig. 15 shows that the mean value of the DAE of the maximum stress error distribution is 3% while for RPD is 5% when the cGAN is at its best performance.

Similarly, to train and validate the FtD and DtS networks 6000 paired images are used for each. The input and the target are changed accordingly. Both DAE and RPD errors for all validation data are calculated following the same procedure for FtS mapping. The target in this case study is the maximum deflection for FtD and the maximum

stress for DtS. The FtD network achieved its best performance at the 112th epoch. The error evaluation at this epoch is presented in Fig. 16. The mean value of the error distributions for DAE target and DAE mean are below 1%. The RPD results of the mean and the target errors are below 5%. The cGAN for the DtS mapping achieved its optimal performance at the 160th epoch. Fig. 17 shows the error evaluation at this epoch where the value of DAE target error distribution is 2% and for RPD target is 4%.

To further showcase the capability of the developed approach, two specific cases are presented. Here, the FtS network is used to predict the stress distribution given the plate's material, thickness and the location of the forces. In Case I, the plate's modulus of elasticity is 60 GPa its thickness is 20 mm. The loads are applied at five random locations as shown in Table 3. In Case II, the plate's modulus is 210 GPa and its thickness is 20 mm. The loads are applied at three random locations as
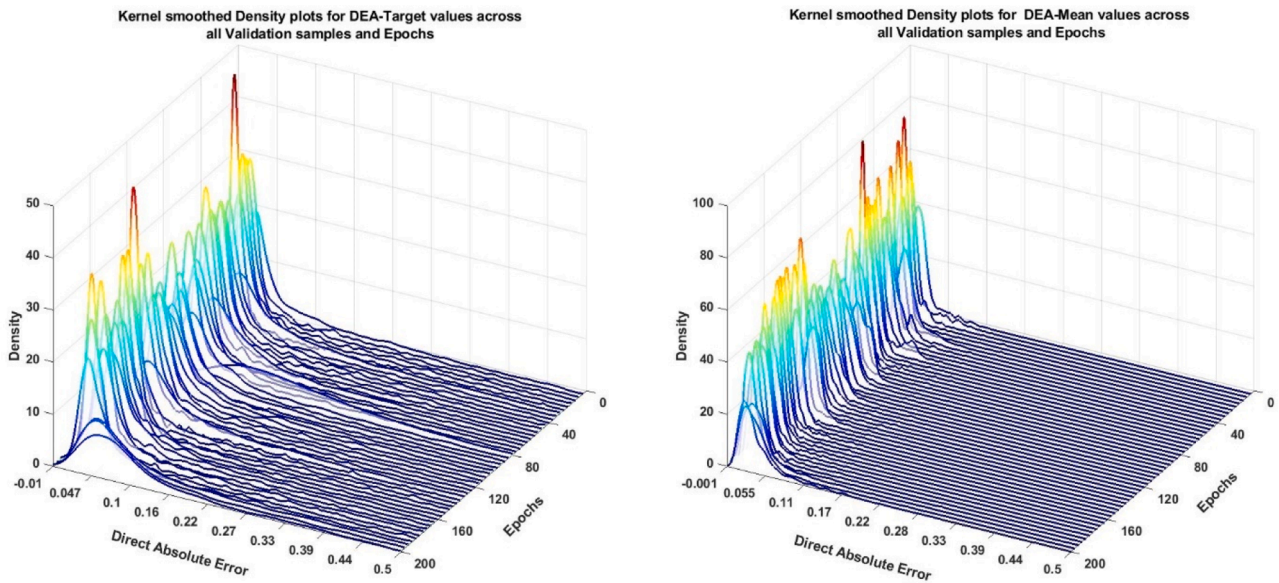
Fig. 11. FtS DAE results: Kernel density for error in the maximum stress (left) and kernel density for mean value of the error (right) for all samples and all epochs.
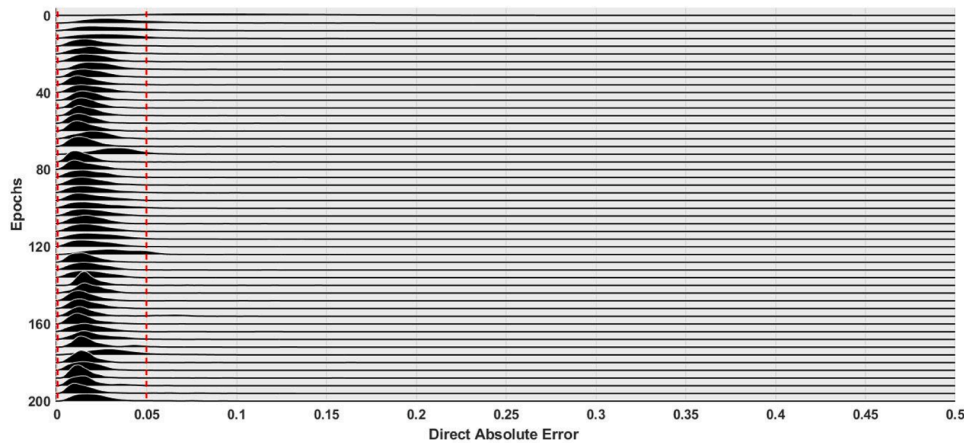

Fig. 12. FtS cascade plot of DAE distribution for all samples and all epochs.

shown in Table 3. In each case, the plate's modulus of elasticity, thickness and location and magnitude of the forces were provided to the trained FtS network. The network outputs an image. Using a modest Lenovo® ThinkPad T460S with Dual Core i7–6600 U @ 2.6 GHz, the generation of the image took 32 ms for Case I and 27 ms for Case II. The resulting image is scaled back to obtain the stress distribution; this process took less than 7 ms in both cases. Figs. 18 and 19 present the target (true) and the predicted stress distribution for Case I and Case II, respectively. In both cases, the predicted stress distribution and target stress distribution are in good agreement. In both cases, the RE (defined in Eq. (6)) is below 3% which is in agreement with the results presented in Fig. 15.

4.3. Validation results of the curved shell

The same methodology to validate the cGAN of the rectangular plate is used with the curved shell herein. Fig. 20 illustrates random ground-truth sample with prediction of the generator at the first checkpoint (4th epoch) and at the last checkpoint (200th epoch). The cGAN for the curved shell shows a similar behaviour to the plate's cGAN where a huge reduction in the DAE occurs through the training process between the first and last checkpoints.

For validating the FtS case, 2000 paired images were used. Like the training in the rectangular plate case study, each pair of images consisted of the force image as an input and the stress image as the target. A randomly selected input image from the validation dataset is used to compare against the ground-truth using Eq. (5) through to Eq. (7). Fig. 20 shows a significant reduction in the absolute error from around 50% at the first checkpoint to less than 10% at the last checkpoint. Furthermore, the maximum error is at the peak force at the first checkpoint, while the maximum error at the last checkpoint is near the two fixed edges of the shell. The curved shell shows very low absolute error levels which justifies the higher level of the relative error.

The cGAN of the curved shell showed its best performance after 96 epochs of training. The error evaluation at this epoch is presented in Fig. 21. The mean value of the error distributions for DAE target and DAE mean are below 1%. The RPD results of the mean and the target error is below 5%.

5. Conclusions

The finite element analysis (FEA) is often used to model problems with complex geometries and homogeneous or heterogenous material properties. However, for this class of problems the method can be
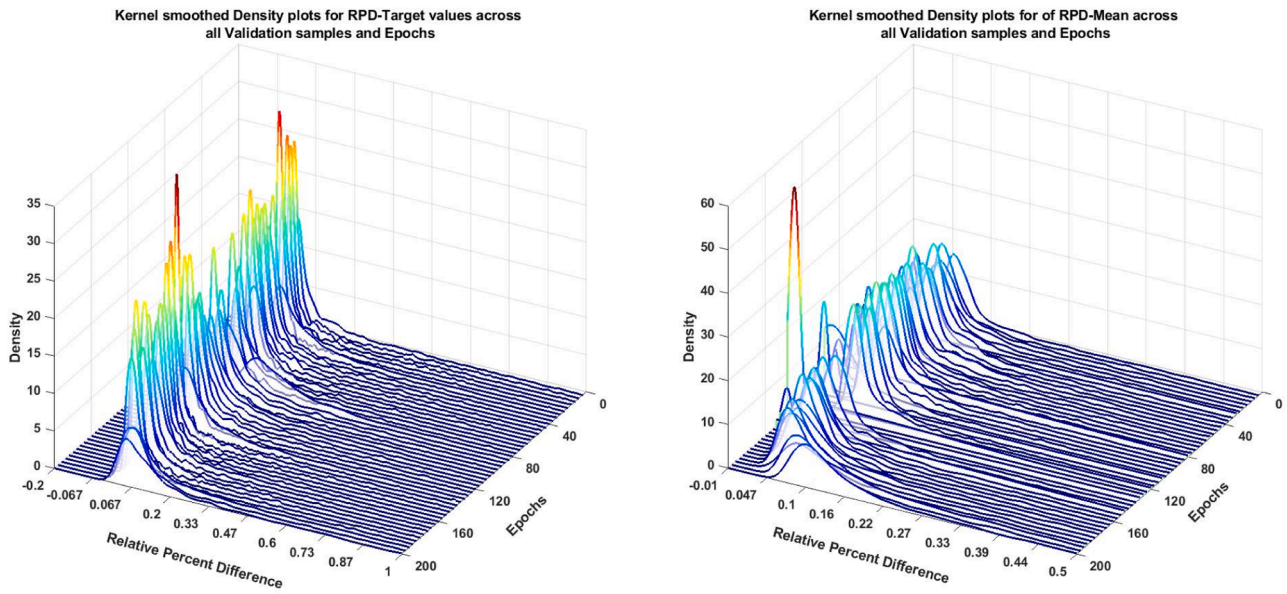
Fig. 13. FtS RPD results: Kernel density for error in the maximum stress (left) and kernel density for mean value of the error (right) for all samples and all epochs.
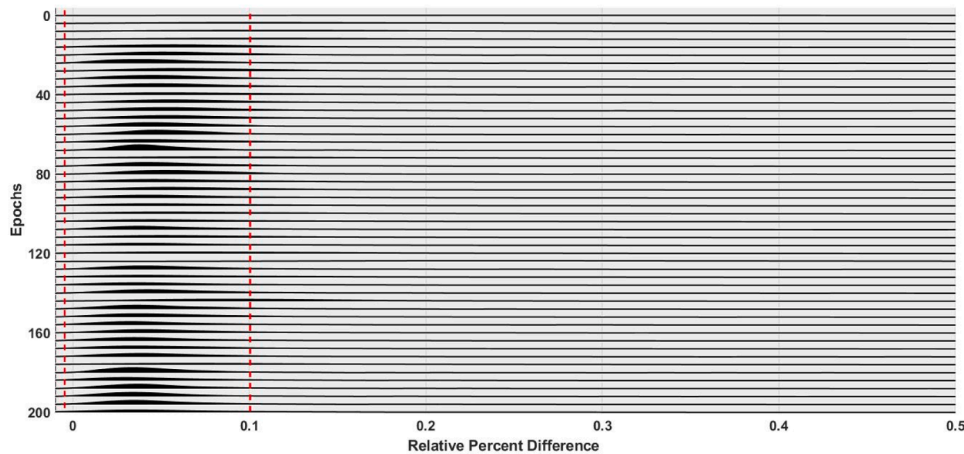

Fig. 14. FtS cascade plot of RPD distribution for all samples and all epochs.

computationally demanding, which is a major barrier for real-time applications such as virtual reality and digital twin models. It is also a significant barrier to work on optimisation problems even when using high performance computing. To reduce the computational demand of FEA problems, several approaches have been developed. Amongst these approaches, machine learning has emerged as a very capable candidate with potential to enable FEA in real-time. The development in deep and convolutional neural networks and combining them with physical models can both accelerate the analysis and retain the method's accuracy. In this paper, we propose a new approach to evaluate the stresses and the displacements in a nonlinear shell structure using image processing and machine learning techniques. The proposed approach significantly reduces the computational costs by treating the mechanical behaviour of the problem as an image. The performance of the proposed approach is assessed through two case studies. First, the nonlinear behaviour of a fully fixed rectangular plate with a varying geometry and a set of material properties is considered under several randomly placed loads. FEA is used to generate a dataset of solutions for hundreds of realizations of the plate. The results of the simulations are converted into contour plots of the stress or the displacement. The full dataset includes the results of 6000 simulations that are transformed into image pairs

with the green and blue channels being assigned to the modulus of elasticity and the plate thickness, respectively. The red channel is assigned to the force and the displacement where half of the image is assigned to the force and the other half is assigned to the displacement to create an image pair. A conditional generative adversarial network (cGAN) is implemented and trained to emulate FEA. Numerical experiments are conducted to recover the stresses or the deflections using the proposed approach. The network is trained to map the force to the displacement, the displacement to the stress and the force to the stress. The network is trained for 200 epochs and the results are saved every 4 epochs in this experiment. Using early stopping with error metrics, the best model was retrieved for each experiment. For both the stress and the deflection, the relative error remains below 5% of the FEA ground-truth.

In the second case study, a curved shell is used to demonstrate the potential of the proposed method to work with alternative geometries. The curved shell is a sector of a cylindrical shell with the two straight edges being fixed and the two curved edges being free. The curved surface has the same variation in the thickness and modulus of elasticity as in the first case study. The loads are also applied randomly, and this resulted in 6000 simulation cases. Only the force to stress scenario is
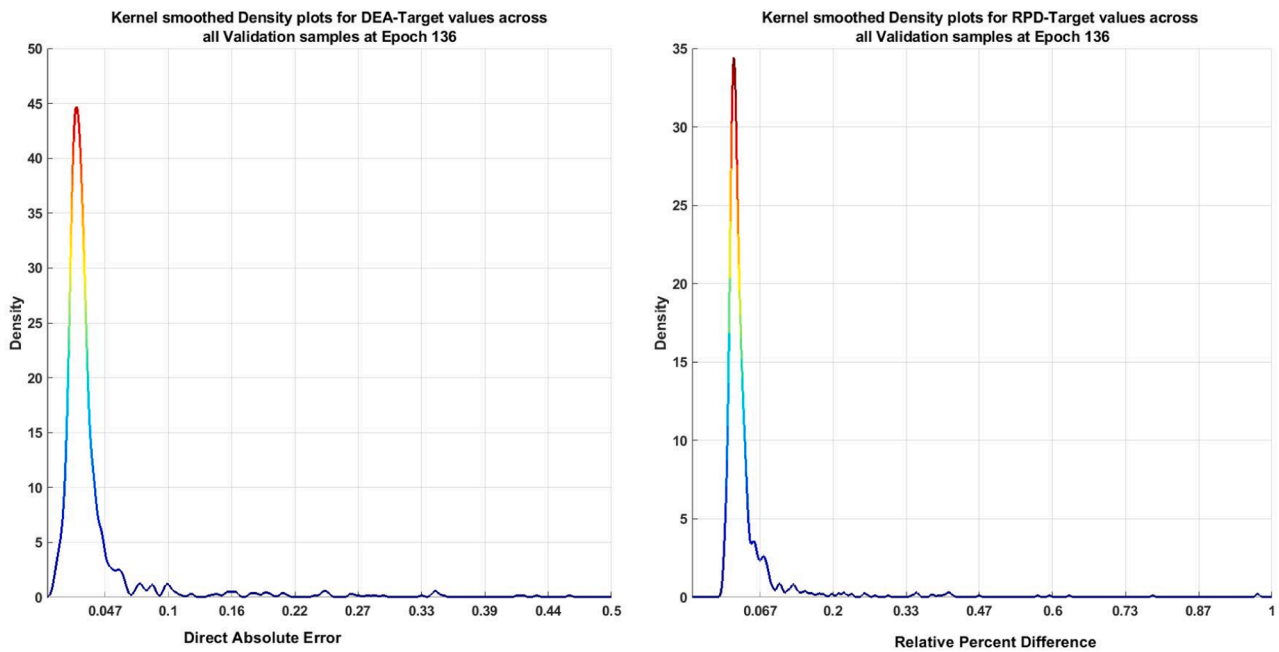
Fig. 15. FtS maximum stress error distribution for RPD and DAE at Epoch 136: Kernel Density for DAE error (Left) and Kernel Density for RPD error (right) for all samples at Epoch 136.
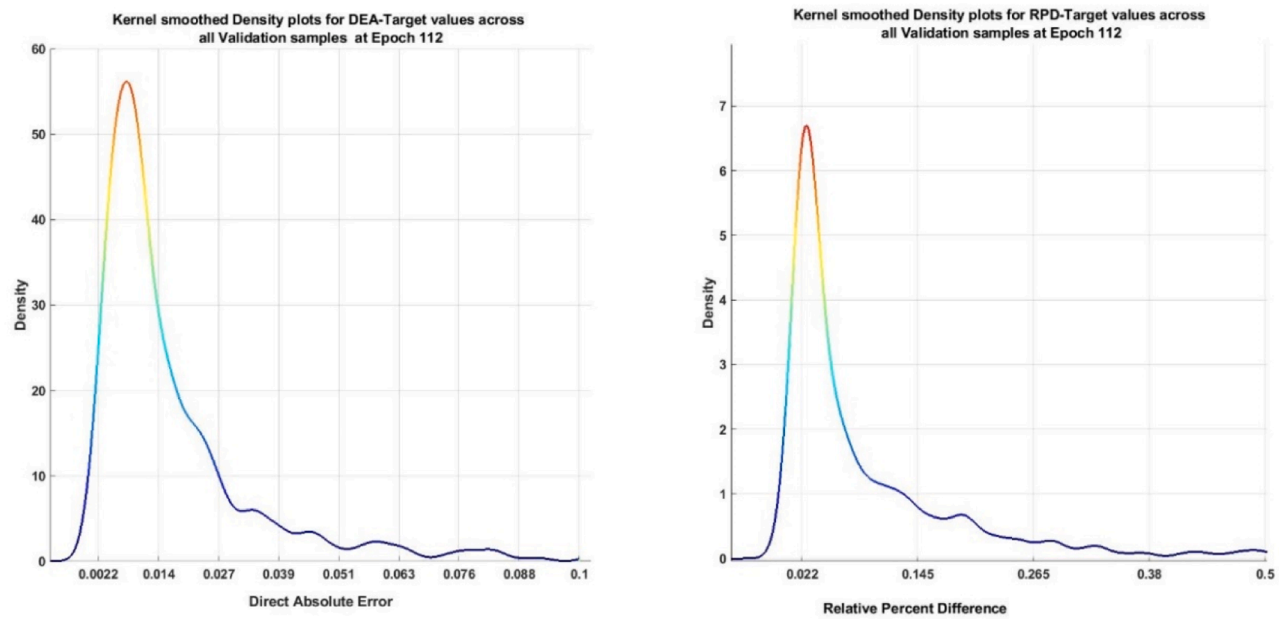

Fig. 16. FtD maximum stress error distribution for RPD and DAE: Kernel Density for DAE error (Left) and Kernel Density for RPD error (right) for all samples at Epoch112.

considered in this case study. To deal with the three-dimensional geometry, a projection into two-dimensions is introduced to generate the training images for the cGAN. In this step, the FEA results are projected onto the horizontal plane to generate the stress and the force images. Again here, the final performance of the resulting cGAN shows a relative error below 5% of the FEA ground-truth.

The two case studies presented and the three numerical experiments that were conducted show that the proposed method can provide a potential alternative for conventional FEA applications that require repetitive solutions of a set of problems. The results also demonstrate the considerable potential of the proposed method in recovering the mechanical behaviour of shell structures where it is possible to reproduce

the solution of the problem at a fraction of the computational cost required by traditional FEA.

Disclosure

CRediT authorship contribution statement

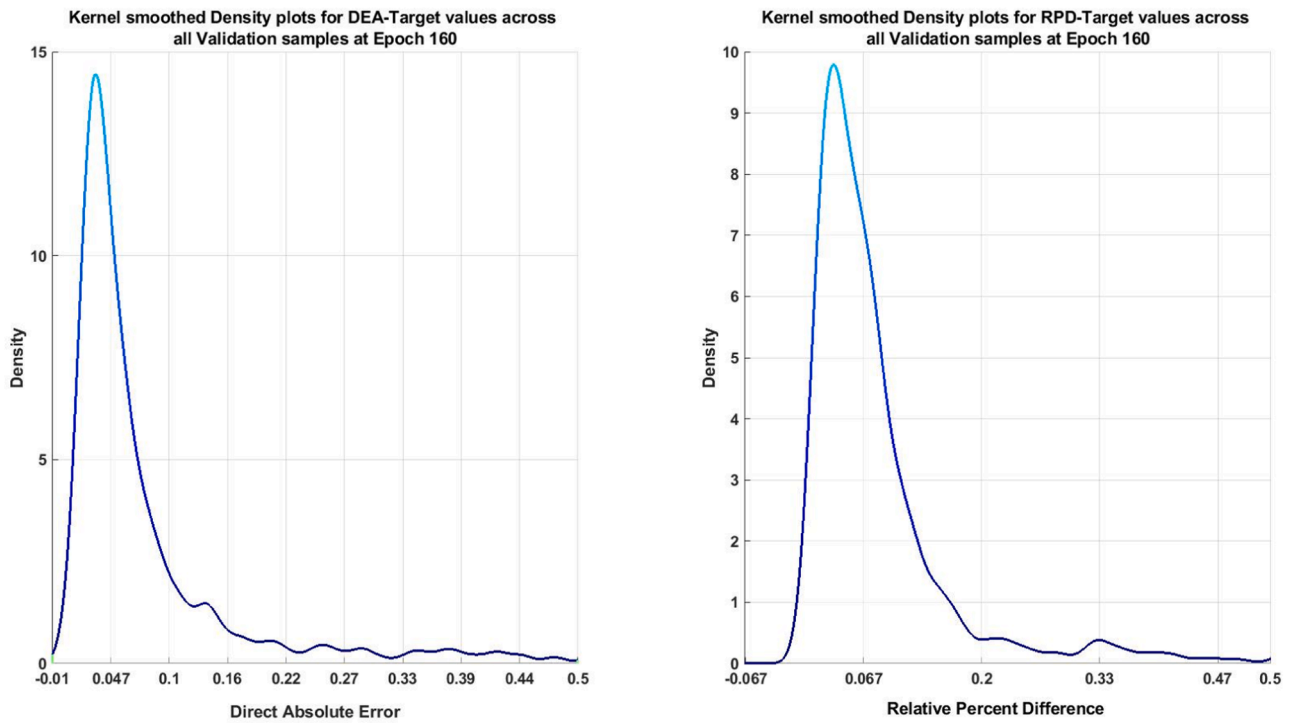**M.S. Nashed:** Conceptualization, Methodology, Investigation,

Fig. 17. DtS maximum stress error distribution for RPD and DAE: Kernel Density for DAE error (left) and Kernel Density for RPD error (right) for all samples at Epoch160.

Table 3
Two realizations of the plate with.

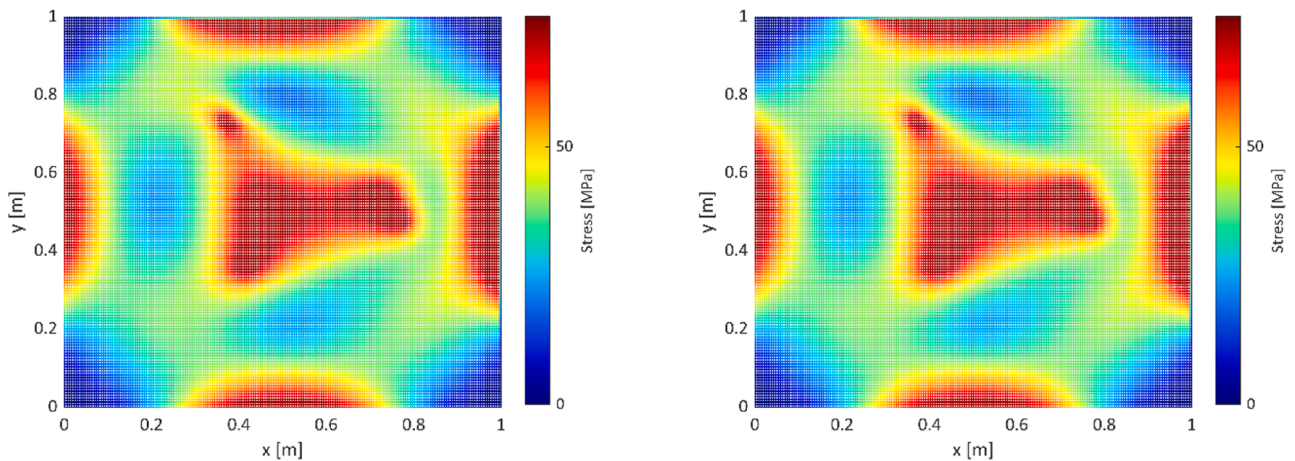| Case I	Modulus of Elasticity: 60 GPa, Thickness of 20 mm				
Force Location	(0.42,0.37)	(0.42,0.37)	(0.42,0.37)	(0.42,0.37)	(0.42,0.37)
Force Magnitude	13.6 kN	13.6 kN	13.6 kN	13.6 kN	13.6 kN
Case I	Modulus of Elasticity: 210 GPa, Thickness: 20 mm				
Force Location	(0.72,0.43)	(0.72,0.43)	(0.72,0.43)		
Force Magnitude	39.2 kN	39.2 kN	39.2 kN		

Fig. 18. Stress distribution in the rectangular plate with five applied loads: (left) target/true stress distribution obtained by FEA and (right) predicted stress distribution.

Writing – original draft, Software, Visualization. **J. Renno:** Project administration, Funding acquisition, Conceptualization, Supervision, Software, Writing – review & editing, Supervision, Visualization, Data curation. **M.S. Mohamed:** Conceptualization, Methodology, Writing – review & editing, Project administration, Funding acquisition, Supervision.

Declaration of Competing Interest

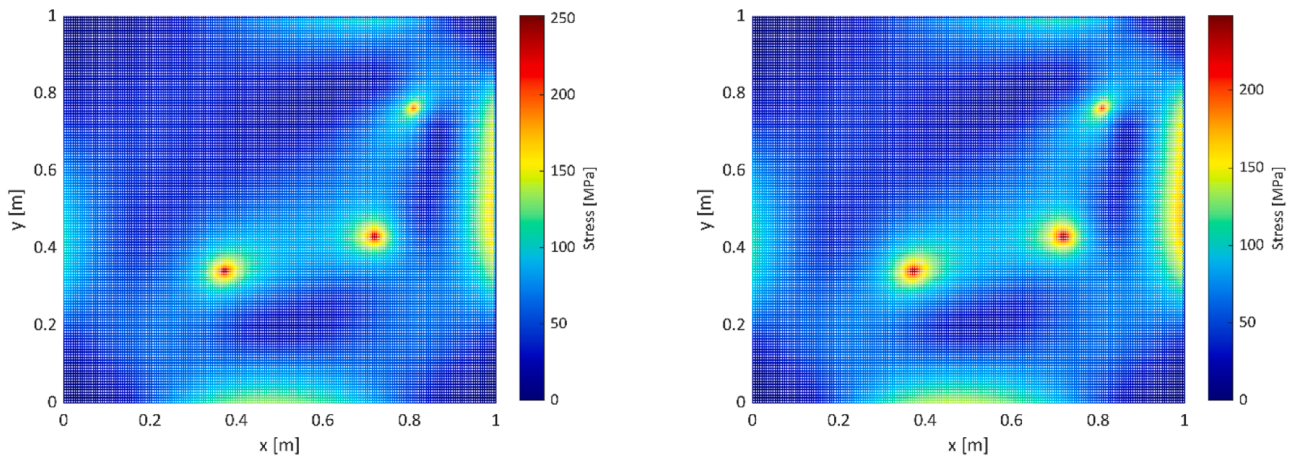The authors declare no potential conflicts of interest with response to

Fig. 19. Stress distribution in the rectangular plate with five applied loads: (left) target/true stress distribution obtained by FEA and (right) predicted stress distribution.
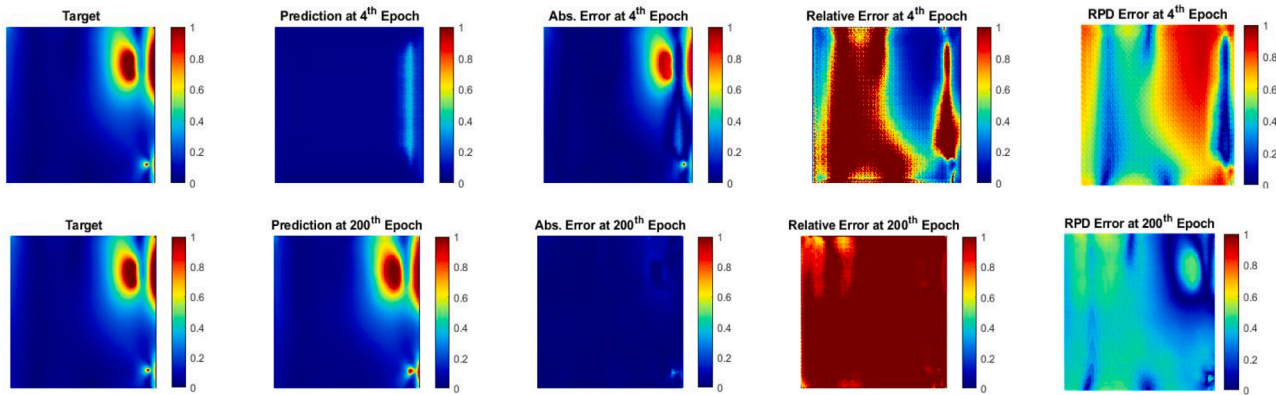

Fig. 20. FtS validation results of the error metrics at the first checkpoint (top) and the last checkpoint (bottom).
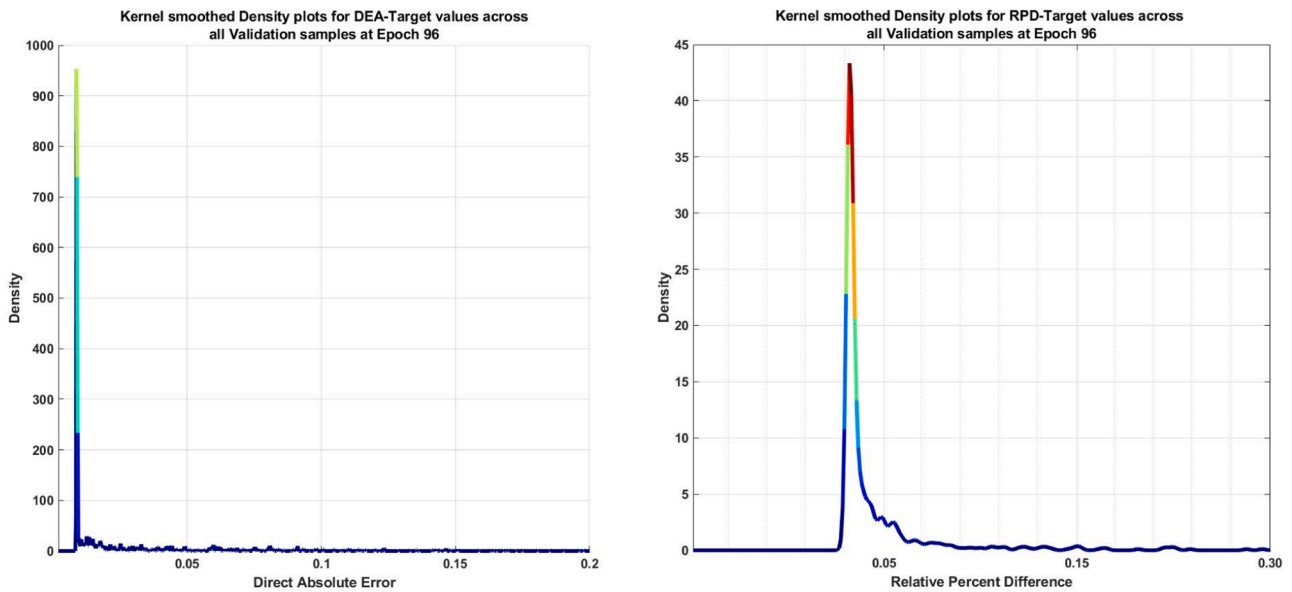

Fig. 21. FtS maximum stress error distribution for RPD and DAE at Epoch 96: Kernel Density for DAE error (Left) and Kernel Density for RPD error (right) for all samples at the 96th epoch.

the research, authorship, and/or publication of this article.

Data availability

The data is available at an open respository.

Acknowledgements

References

[1] Liangyin X, Yunpeng L, Sheng Z, Biaosong C. Efficient visualization strategies for large-scale finite element models. J Comput Inf Sci Eng 2018;18(1):011007.

[2] Lee HH. Finite element simulations with ANSYS workbench 2020. SDC Publications; 2020.

[3] Driscoll M. The impact of the finite element method on medical device design. Springer; 2019.

[4] Koslan MFS, Zaidi AMA, Othman MZ, Abdullah S, Thanakodi S. The effect of mesh sizing toward deformation result in computational dynamic simulation for blast loading application. Mod Appl Sci 2013;7:23.

[5] Lee EJ, El-Tawil S. FEMvrml: an interactive virtual environment for visualization of finite element simulation results. Adv Eng Softw 2008;39:737–42.

[6] Kakosimos P, Beniakar M, Sarigiannidis AG, Kladas AG. Model predictive control employing finite-element methods for aerospace actuators. Mater Sci Forum 2016: 202–6. Trans Tech Publ.

[7] Ali N, Behdinan K, Fawaz Z. Applicability and viability of a GA based finite element analysis architecture for structural design optimization. Comput Struct 2003;81: 2259–71.

[8] Marinkovic D, Zehn M. Survey of finite element method-based real-time simulations. Appl Sci 2019;9:2775.

[9] Georgescu S, Chow P, Okuda H. GPU acceleration for FEM-based structural analysis. Arch Computa Methods Eng 2013;20:111–21.

[10] Yan J, Zhang Q, Xu Q, Fan Z, Li H, Sun W, et al. Deep learning driven real time topology optimisation based on initial stress learning. Adv Eng Inform 2022;51: 101472.

[11] Yoshida T, Okuzono T, Sakagami K. A parallel dissipation-free and dispersion-optimized explicit time-domain fem for large-scale room acoustics simulation. Buildings 2022;12:105.

[12] Wang Y, Guo S, Gao B. Vascular elasticity determined mass-spring model for virtual reality simulators. Inte J Mechatron Autom 2015;5.

[13] White RE, Macdonald JHG, Alexander NA. A nonlinear frequency-dependent spring-mass model for estimating loading caused by rhythmic human jumping. Eng Struct 2021;241:112229.

[14] Ferhatoglu E, Cigeroglu E, Özgüven HN. A novel modal superposition method with response dependent nonlinear modes for periodic vibration analysis of large MDOF nonlinear systems. Mech Syst Signal Process 2020;135:106388.

[15] Favoretto B, de Hillerin CA, Bettinotti O, Oancea V, Barbarulo A. Reduced order modeling via PGD for highly transient thermal evolutions in additive manufacturing. Comput Methods Appl Mech Eng 2019;349:405–30.

[16] Nasri MA, Robert C, Ammar A, El Arem S, Morel F. Proper Generalized Decomposition (PGD) for the numerical simulation of polycrystalline aggregates under cyclic loading. C R Méc 2018;346:132–51.

[17] Barbarulo A, Ladevèze P, Riou H, Kovalevsky L. Proper Generalized Decomposition applied to linear acoustic: a new tool for broad band calculation. J Sound Vib 2014; 333:2422–31.

[18] Diwan GC, Mohamed MS, Seaid M, Trevelyan J, Laghrouche O. Mixed enrichment for the finite element method in heterogeneous media. Int J Numer Methods Eng 2015;101:54–78.

[19] Renno J, Manconi E, Mace B. A finite element method for modelling waves in laminated structures. Adv Struct Eng 2013;16:61–75.

[20] Van Hal B, Desmet W, Vandepitte D. Hybrid finite element - wave-based method for steady-state interior structural-acoustic problems. Comput Struct 2005;83: 167–80.

[21] Nikitin I, Nikitina L, Frolov P, Goebbels G, Göbel M, Klimenko S, et al. Real-time simulation of elastic objects in virtual environments using finite element method and precomputed Green's functions. In: Proceedings of the workshop on virtual environments. Eurographics Association; 2002. p. 47–52.

[22] Logg A, Lundholm C, Nordaas M. Finite element simulation of physical systems in augmented reality. Adv Eng Softw 2020;149:102902.

[23] Huang JM, Ong SK, Nee AYC. Real-time finite element structural analysis in augmented reality. Adv Eng Softw 2015;87:43–56.

[24] Huang J, Ong SK, Nee AY. Real-time finite element structural analysis in augmented reality. Adv Eng Softw 2015;87:43–56.

[25] Fiorentino M, Monno G, Uva A. Interactive "touch and see" FEM simulation using augmented reality. Int J Eng Educ 2009;25:1124–8.

[26] Guan Q, Du X, Shao Y, Lin L, Chen S. Three-dimensional simulation of scalp soft tissue expansion using finite element method. Comput Math Methods Med 2014; 2014. https://doi.org/10.1155/2014/360981.

[27] Li X, Li C, Xue Z, Tian X. Investigation of transient thermo-mechanical responses on the triple-layered skin tissue with temperature dependent blood perfusion rate. Int J Therm Sci 2019;139:339–49.

[28] Cheung JTM, Zhang M. A 3-dimensional finite element model of the human foot and ankle for insole design. Arch Phys Med Rehabil 2005;86:353–8.

[29] Panda SK, Buist ML. A finite element approach for gastrointestinal tissue mechanics. Int J Numer Method Biomed Eng 2019;35:e3269.

[30] Heidlauf T, Klotz T, Rode C, Siebert T, Röhrle O. Force enhancement and stability of finite element muscle models. PAMM 2016;16:85–6.

[31] Heidlauf T, Klotz T, Rode C, Siebert T, Röhrle O. A continuum-mechanical skeletal muscle model including actin-titin interaction predicts stable contractions on the descending limb of the force-length relation. PLoS Comput Biol 2017;13:e1005773.

[32] Ferrant M, Nabavi A, Macq B, Jolesz FA, Kikinis R, Warfield SK. Registration of 3-D intraoperative MR images of the brain using a finite-element biomechanical model. IEEE Trans Med Imaging 2001;20:1384–97.

[33] Picinbono G, Lombardo JC, Delingette H, Ayache N. Improving realism of a surgery simulator: linear anisotropic elasticity, complex interactions and force extrapolation. J Vis Comput Animat 2002;13:147–67.

[34] Wu W, Sun J, Heng PA. A hybrid condensed finite element model for interactive 3D soft tissue cutting. Medicine meets virtual reality. 11. IOS Press; 2003. p. 401–3.

[35] Joannin C, Thouverez F, Chouvion B. Reduced-order modelling using nonlinear modes and triple nonlinear modal synthesis. Comput Struct 2018;203:18–33.

[36] Chinesta F, Ladeveze P, Cueto E. A short review on model order reduction based on proper generalized decomposition. Arch Comput Methods Eng 2011;18:395–404.

[37] Tharwat A. Principal component analysis-a tutorial. Int J Appl Pattern Recognit 2016;3:197–240.

[38] Yeh YC, Kuo YH, Hsu DS. Building an expert system for debugging FEM input data with artificial neural networks. Expert Syst Appl 1992;5:59–70.

[39] Bihlo A, Popovych RO. Physics-informed neural networks for the shallow-water equations on the sphere. J Comput Phys 2022;456:111024.

[40] Penwarden M, Zhe S, Narayan A, Kirby RM. Multifidelity modeling for physics-informed neural networks (PINNs). J Comput Phys 2022;451:110844.

[41] Karniadakis GE, Kevrekidis IG, Lu L, Perdikaris P, Wang S, Yang L. Physics-informed machine learning. Nat Rev Phys 2021;3:422–40.

[42] Alber M, Buganza Tepole A, Cannon WR, De S, Dura-Bernal S, Garikipati K, et al. Integrating machine learning and multiscale modeling-perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences. npj Digit Med 2019;2:115.

[43] Jokar M, Semperlotti F. Finite element network analysis: a machine learning based computational framework for the simulation of physical systems. Comput Struct 2021;247:106484.

[44] Singh K, Kapania RK. Accelerated optimization of curvilinearly stiffened panels using deep learning. Thin-Walled Struct 2021;161:107418.

[45] Nguyen T, Kashani A, Ngo T, Bordas S. Deep neural network with high-order neuron for the prediction of foamed concrete strength. Comput Aided Civ Infrastruct Eng 2019;34:316–32.

[46] Viana FA, Nascimento RG, Dourado A, Yucesan YA. Estimating model inadequacy in ordinary differential equations with physics-informed neural networks. Comput Struct 2021;245:106458.

[47] Saha S, Gan Z, Cheng L, Gao J, Kafka OL, Xie X, et al. Hierarchical Deep Learning Neural Network (HiDeNN): an artificial intelligence (AI) framework for computational science and engineering. Comput Methods Appl Mech Eng 2021; 373:113452.

[48] Haghighat E, Juanes R. Sciann: a keras/tensorflow wrapper for scientific computations and physics-informed deep learning using artificial neural networks. Comput Methods Appl Mech Eng 2021;373:113552.

[49] Haghighat E, Raissi M, Moure A, Gomez H, Juanes R. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. Comput Methods Appl Mech Eng 2021;379:113741.

[50] Wen J, Zou Q, Wei Y. Physics-driven machine learning model on temperature and time-dependent deformation in lithium metal and its finite element implementation. J Mech Phys Solids 2021;153:104481.

[51] Zobeiry N, Humfeld KD. A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications. Eng Appl Artif Intell 2021;101:104232.

[52] Albawi S, Mohammed TA, Al-Zawi S. Understanding of a convolutional neural network. In: Proceedings of the international conference on engineering and technology (ICET). Ieee; 2017. p. 1–6.

[53] Puttagunta M, Ravi S. Medical image analysis based on deep learning approach. Multimed Tools Appl 2021:1–34.

[54] Hameurlaine M., Moussaoui A., Safa B. Deep learning for medical image analysis, 2019.

[55] Tajdari M, Pawar A, Li H, Tajdari F, Maqsood A, Cleary E, et al. Image-based modelling for adolescent idiopathic scoliosis: mechanistic machine learning analysis and prediction. Comput Methods Appl Mech Eng 2021;374:113590.

[56] Fujiyoshi H, Hirakawa T, Yamashita T. Deep learning-based image recognition for autonomous driving. IATSS Res 2019;43:244–52.

[57] Ranjan N, Bhandari S, Zhao HP, Kim H, Khan P. City-wide traffic congestion prediction based on CNN, LSTM and transpose CNN. IEEE Access 2020;8: 81606–20.

[58] Chen L, Bentley P, Mori K, Misawa K, Fujiwara M, Rueckert D. DRINet for medical image segmentation. IEEE Trans Med Imaging 2018;37:2453–62.

[59] Mortazi A, Bagci U. Automatically designing CNN architectures for medical image segmentation. In: Proceedings of the international workshop on machine learning in medical imaging. Springer; 2018. p. 98–106.

[60] Xu W, Wang C, Yuan J. Impact performance of an annular shaped charge designed by convolutional neural networks. Thin-Walled Struct 2021;160:107241.

[61] Krokos V, Bui Xuan V, Bordas SPA, Young P, Kerfriden P. A Bayesian multiscale CNN framework to predict local stress fields in structures with microscale features. Comput Mech 2022;69:733–66.

[62] Deshpande S, Lengiewicz J, Bordas SPA. Probabilistic deep learning for real-time large deformation simulations. Comput Methods Appl Mech Eng 2022;398:115307.

[63] Dumoulin V., Visin F. A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:160307285. 2016.

[64] Isola P, Zhu JY, Zhou T, Efros AA. Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 1125–34.

[65] Bode M, Gauding M, Lian Z, Denker D, Davidovic M, Kleinheinz K, et al. Using physics-informed enhanced super-resolution generative adversarial networks for subfilter modeling in turbulent reactive flows. Proc Combust Inst 2021;38: 2617–25.

[66] Haykin SS. Neural networks: a comprehensive foundation. Prentice Hall; 1999. Upper Saddle River, N.J.

[67] Shen D, Wu G, Suk HI. Deep learning in medical image analysis. Annu Rev Biomed Eng 2017;19:221–48.

[68] Ranjan N, Bhandari S, Khan P, Hong YS, Kim H. Large-scale road network congestion pattern analysis and prediction using deep convolutional autoencoder. Sustainability 2021;13:5108.

[69] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. Adv Neural Inf Process Syst 2014;27.

[70] Gabriel JD, John AS. ANSYS engineering analysis system user's manual. Houston, PA.: Swanson Analysis Systems; 1985. 1985.

[71] Richard GB, Ali MS. Roark's formulas for stress and strain. 8th ed. New York: McGraw-Hill Education; 2020. 9th ed. /Edition.

[72] Wiggins RH, Davidson HC, Harnsberger HR, Lauman JR, Goede PA. Image file formats: past, present, and future. Radiographics 2001;21:789–98.

[73] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature 1986;323:533–6.

[74] MATLAB. version (R2020a). Natick, Massachusetts: The MathWorks Inc; 2010.

[75] Géron A. Hands-on machine learning with scikit-learn, keras, and tensorflow: concepts, tools, and techniques to build intelligent systems. O'Reilly Media; 2019.