applied sciences

MDPI

Article

Efficient Deep Learning for Gradient-Enhanced Stress Dependent Damage Model

Xiaoying Zhuang [1,2], L. C. Nguyen [3], Hung Nguyen-Xuan [4], Naif Alajlan [5] and Timon Rabczuk [5,*]

1 Division of Computational Mechanics, Ton Duc Thang University, Ho Chi Minh City, Vietnam
2 Faculty of Civil Engineering, Ton Duc Thang University, Ho Chi Minh City, Vietnam; xiaoying.zhuang@tdtu.edu.vn
3 Institute for Continuum Mechanics, Leibniz Universität Hannover, Appelstr. 11, 30167 Hannover, Germany; zhuang@ikm.uni-hannover.de
4 CIRTech Institute, Ho Chi Minh City University of Technology (HUTECH), Ho Chi Minh City, Vietnam; ngx.hung@hutech.edu.vn
5 Department of Computer Engineering College of Computer and Information Sciences King Saud University, Riyadh 11543, Saudi Arabia; najlan@KSU.EDU.SA
* Correspondence: timon.rabczuk@uni-weimar.de

Received: 21 January 2020; Accepted: 11 March 2020; Published: 8 April 2020

Abstract: This manuscript introduces a computational approach to micro-damage problems using deep learning for the prediction of loading deflection curves. The location of applied forces, dimensions of the specimen and material parameters are used as inputs of the process. The micro-damage is modelled with a gradient-enhanced damage model which ensures the well-posedness of the boundary value and yields mesh-independent results in computational methods such as FEM. We employ the Adam optimizer and Rectified linear unit activation function for training processes and research into the deep neural network architecture. The performance of our approach is demonstrated through some numerical examples including the three-point bending specimen, shear bending on L-shaped specimen and different failure mechanisms.

Keywords: deep neural network; deep learning; gradient enhanced damage; stress-level dependent damage model

1. Introduction

Neural networks (NN) have been used for numerous applications in different areas including computational mechanics. Initially, single-layer or shallow neural networks (SNN) consisted only of one input and one output layer. Later on, additional hidden layers were added to the network architecture resulting in so-called deep neural networks (DNN). Table 1 gives a brief summary of different network architectures. Neural networks or more specifically deep neural networks suffer from three major difficulties, i.e., (a) vanishing gradients, (b) over-fitting and (c) computational loading. However, significant advances such as deep belief networks (DBN) [1], rectified linear unit (ReLU) activation functions [2], drop-out algorithms [3] or back-propagation algorithms and associated tools have contributed to the popularity of DNN. The vanishing gradient problem for instance has been significantly alleviated thanks to the RELU activation function and cross entropy-driven learning techniques. Nonetheless, certain issues such as over-fitting still remains a challenge in deep neural networks. Common techniques to address such problems include regularization techniques.

We recall that the crucial idea behind Artificial Neural Network (ANN) is that many neurons can be joined together by connecting weights to conduct complex computations. The structure is often

demonstrated as a graph or a map whose nodes are the neurons and each (directed) edge in the graph connects the output to the input of the associated neurons. Deep learning methods, representative of learning methods with multiple processing layers in the hidden layers, consist of linear and non-linear transformations [4,5]. Some simplifications of the problems stem from their need for a usually very strong CPU and a noticeably long time to detect and analyse slow convergence behaviour. For complex problems, the solution could be non-existent. A newer method or a faster algorithm has yet to be found in recent decades.

Table 1. The branches of the neural network depending on the layer architecture.

Single-Layer Neural Networks		**Input Layer-Output Layer**
Multilayer-Layer Neural Networks	Shallow Neural Networks	Input Layer-Hidden Layer -Output Layer
Neural Networks	Deep Neural Networks	Input Layer-Hidden Layers -Output Layers

While machine learning (ML) approaches have been extensively and successfully used in numerous areas, its application in "modeling and simulation" is still in its infancy. For instance, in medicine, ML has been employed in diagnostics where it outperforms the diagnosis of established physicians [6]. The authors of [7] have applied successfully deep learning to cellular imaging. Park et al. [8] focused on the problems of cellular imaging in regulatory genomics. Goh et al. [9] took advantage of deep learning in computational chemistry. Deep learning techniques were also applied in applications such as bio-informatics, or the public health sector [10]. On the other hand, most approaches in engineering, or more specifically computational mechanics, have been used in data-driven contexts though there are numerous other applications such as the direct solution of partial differential equations [11], which have the potential to drastically accelerate the design to analysis time and the way modeling and simulation is performed. In the data-driven context, neural networks have commonly been used in the context of constitutive models [12,13] as an alternative to traditional constitutive models. The models in [14–16] presented interesting approaches to capture the response of anisotropic materials. New training algorithms for specific constitutive laws have been presented in [17,18]. However, setting up the network architecture for such engineering problems still remains a major challenge and is often determined by trial and error. The authors of [19] exploited Deep Learning to optimize the fine-scale structure of composites. Multi-fields problems were tackled for instance in [20,21]. Recently, Lee et al. [22] has applied deep learning algorithms to structural analysis. In this manuscript, we present a novel methodology to predict the load-deflection curve by deep learning. Passing through the three-point bending as an illustrative example, we suggest some possible architectures of the deep neural networks based on the Adam optimizer. Such findings can open a new branch of research that may prove beneficial to the fourth industrial revolution, where deep learning algorithms play a major role in big data analysis of structural engineering.

2. Continuum Damage Theory

2.1. Constitutive Equations of Isotropic Damage Models

Let us assume small strain theory in the context of a scalar/isotropic damage model. The relation between the Cauchy stress tensor $\boldsymbol{\sigma}$ and the linear strain tensor $\boldsymbol{\varepsilon}$ is given by

$$\boldsymbol{\sigma} = (1-\omega)\mathbb{C} : \boldsymbol{\varepsilon} = \mathbb{C}^{eff} : \boldsymbol{\varepsilon} \tag{1}$$

where ω denotes a monotonically increasing scalar damage variable, $\mathbb{C}$ the fourth-order elasticity tensor and $\mathbb{C}^{eff} = (1-\omega)\mathbb{C}$ refers to the so-called 'effective' elasticity tensor. A value of $\omega = 1$ indicate a completely damaged material while the material is intact for a value o $\omega = 0$. The evolution of the

damage variable is commonly governed by a scalar state variable κ, i.e., $\omega = \omega(\kappa)$. The Kuhn–Tucker conditions, which finally lead to a convex optimization problem, ensures that the product of the loading function f and the rate of the state variable is equal to zero:

$$\begin{cases} f & \leq 0 \\ \dot{\kappa} & \geq 0 \\ f\dot{\kappa} & = 0 \end{cases} \tag{2}$$

ε_{eq} indicating the effective strain, which is a projection of a multi-axial strain state onto a single scalar value. We consider a strain-based formulation, where the loading function is expressed in terms of the effective strain (instead of effective stresses), which facilitates the implementation in the context of displacement based finite element analysis:

$$f = f(\epsilon_{eq}, \kappa) = \epsilon_{eq} - \kappa = 0 \tag{3}$$

We test two different approaches to compute ε_{eq}. The first one has been presented by Mazars [23] for quasi-brittle materials and is given by

$$\varepsilon_{eq} = \sqrt{\langle \boldsymbol{\varepsilon} \rangle : \langle \boldsymbol{\varepsilon} \rangle} = \sqrt{\langle \varepsilon_1 \rangle^2 + \langle \varepsilon_2 \rangle^2} \tag{4}$$

where $\langle \cdot \rangle$ indicates a Macaulay bracket while ε_1 and ε_2—for two-dimensional problems—denote the principal strains. For this approach, we employ an exponential evolution law for the damage variable as suggested by Peerlings et al. [24]:

$$\omega(\kappa) = 1 - \frac{\kappa_0}{\kappa}\left[1 - \alpha + \alpha e^{-\beta(\kappa - \kappa_0)}\right] \tag{5}$$

where κ_0 stands for an initial damage threshold while the material parameters α and β needs to be determined through experiments. We also employ an expression for the equivalent strain as suggested by de Vree et al. [25] and Peerlings et al. [26]:

$$\varepsilon_{eq} = \frac{k-1}{2k(1-\nu)} I_1 + \frac{1}{2k}\sqrt{\frac{(k-1)^2}{(1-2\nu)^2} I_1^2 + \frac{12k}{(1+\nu)^2} J_2} \tag{6}$$

where I_1 indicates the first invariant of the linear strain tensor and J_2 refers to the second invariant of the deviatoric part of the strain tensor.

$$\begin{cases} I_1 & = \varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33} \\ J_2 & = \dfrac{1}{2}\varepsilon_{ij}\varepsilon_{ij} - \dfrac{1}{6} I_1^2 \end{cases} \tag{7}$$

The parameter k in Equation (6) determines the ratio of the compressive and tensile strength.

2.2. Gradient-Enhanced Damage Models

It is well known that local damage models lead to ill-posed boundary value problems and associated numerical difficulties such as mesh-dependent results in computational modeling. Non-local damage models as proposed [26–30] restore the well-posedness of the boundary value problem by introducing an intrinsic length scale which smear the crack over a certain width. In such models, the loading function is therefore expressed in terms of non-local equivalent strain $\bar{\varepsilon}_{eq}(\boldsymbol{x})$ which in turn depend on the local effective strain and a weighting function $g(\boldsymbol{\xi})$ governing the domain of non-locality:

$$\bar{\varepsilon}_{eq}(\boldsymbol{x}) = \frac{1}{V}\int_V g(\boldsymbol{\xi})\, \varepsilon_{eq}(\boldsymbol{x} + \boldsymbol{\xi})\, dV \tag{8}$$

An alternative to such strongly nonlocal models are weakly nonlocal approaches [26,28,31] which are based on Taylor series expansions to approximate the effective strain. Substituting these into the Equation (8) yields a different expression for the non-local equivalent strain. However, it is well known that such formulations require C^1—continuity for second-order gradient models (C^2—continuity for fourth-order gradient enhancements), which complicates their implementation in Lagrange polynomial based finite element analysis. Implicit formulations overcome these difficulties. Differentiating twice and neglecting higher-order gradient terms of the local equivalent strain, the implicit second-order gradient model reads:

$$\varepsilon_{eq} = \bar{\varepsilon}_{eq} - \frac{1}{2} l_c^2 \nabla^2 \bar{\varepsilon}_{eq}(\boldsymbol{x}) \tag{9}$$

where the parameter l_c indicates the intrinsic length scale, which can be regarded either as purely numerical regularization parameter or material parameter which needs to be determined through experiments or other theoretical considerations. For concrete materials, Bažant et al. related l_c to the maximum aggregate size [32]. Furthermore, the following von Neumann boundary conditions need to be satisfied [28,33]:

$$\nabla \bar{\varepsilon}_{eq} \cdot \boldsymbol{n} = 0 \tag{10}$$

It can be shown that the gradient enhanced damage model, Equation (9), can be expressed in terms of the principal directions and an anisotropic weight function:

$$\varepsilon_{eq}(x_1, x_2) = \bar{\varepsilon}_{eq}(x_1, x_2) - c_1 \frac{\partial^2 \bar{\varepsilon}_{eq}(x_1, x_2)}{\partial {x_1}^2} - c_2 \frac{\partial^2 \bar{\varepsilon}_{eq}(x_1, x_2)}{\partial {x_2}^2} \tag{11}$$

where c_1, c_2 are the weighting factors. More details about the derivation of above described gradient-enhanced damage model and its implementation can be found for instance in [26].

3. Training Data

3.1. Optimizers

In machine learning (ML) approaches, the weights and biases of the network (see Figure 1) are obtained through minimizing an objective function. Commonly, gradient descent methods are employed in ML. The update of the gradient descent has the form $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k)$, η being the step size, which is also referred to as *learning rate*. Gradient descent methods are based on so-called *batches* employed to calculate the gradient in a single iteration. Commonly, the batch is the entire data set. In addition, a batch can be enormous. A very large batch may cause even a single iteration to take a very long time in computation. By choosing examples from a data set at random, it could estimate (albeit noisily) a big average from a much smaller one. *Stochastic gradient descent* (SGD) takes this idea to the extreme—it uses solely one example (a batch size of 1) per iteration. Every one computation for all data points, it is called one *epoch*. The term "stochastic" implies that the one example comprising each batch is chosen arbitrarily. The mini-batch stochastic gradient descent method (mini-batch SGD) is a compromise between the full-batch iteration and SGD. A mini-batch is usually between 10 and 1000 examples, chosen at random. Mini-batch SGD reduces the amount of noise in SGD but is still more efficient than full-batch. It is well known that the steepest gradient descent faces difficulties in areas where surface curves exhibit different gradients in different dimensions, which frequently occurs around local optima. To reduce the risk of getting stuck in local optima, we take advantage of the *Momentum Method*, which shows analogies to the equations governing the movement of particles in a viscous medium:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k - \boldsymbol{v}_k \\ \boldsymbol{v}_k &= \gamma \boldsymbol{v}_{k-1} + \eta \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k) \end{aligned} \tag{12}$$

where the parameter $\gamma \approx 0.9$ governs the updating of the iterations within the stochastic gradient descent method (SGDM). This approach is commonly employed with the back propagation algorithm,

which will be explained later. However, the momentum method is based on the situation where a ball rolling downhill blindly follows the slope. A smarter ball would slow down before the slope goes up again, which is the essence of the *Nesterov Accelerated Gradient* (NAG) approach [34]. Therefore, the changing of momentum is computed, which is simply the sum of the momentum vector and the gradient vector at the current step. The changing of the Nesterov momentum is then the sum of the momentum vector and the gradient vector at the approximation of the next step:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k - \boldsymbol{v}_k \\ \boldsymbol{v}_k &= \gamma \boldsymbol{v}_{k-1} + \eta \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}_k - \gamma \boldsymbol{v}_{k-1}\right) \end{aligned} \tag{13}$$

Gradients of complex functions as used in DNN tend to either vanish or explode. These vanishing/exploding problems become more pronounced with increasing complexity of the function. These issues can be alleviated by an adapted learning rate method, named RMSProp, which was suggested by Geoff Hinton in Lecture 6e of his Coursera Class. The idea is based on a moving average of the squared gradients, which normalizes the gradient. This approach proves effective to balance the step size by decreasing the step for large gradient to avoid explosion while increasing the step for small gradients, which evenutally alleviates the vanishment problem:

$$\begin{aligned} r_k &= (1-\beta) \cdot \left[\nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}_k\right)\right]^2 + \beta r_{k-1} \\ \boldsymbol{v}_{k+1} &= \frac{\eta}{\sqrt{r_k}} \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}_k\right) \\ \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k - \boldsymbol{v}_{k+1} \end{aligned} \tag{14}$$

where the so-called *squared gradient decay factor* β ranges from 0 to 1; we employ the suggested value of $\beta \approx 0.9$. Another method, that calculates learning rates adaptively for each parameter is the Adaptive Moment Estimation (Adam) [35]. Adam keeps an exponentially decaying average of past gradients $\boldsymbol{v}_k$ as well. However, in contrast to the momentum method, Adam can be seen as a heavy ball with friction, that prefers flat minima in the error surface. The past decaying averages and squared gradients $\boldsymbol{v}_k$ and r_k can be obtained by

$$\begin{aligned} \boldsymbol{v}_k &= \beta_1 \boldsymbol{v}_{k-1} + (1-\beta_1) \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}_k\right) \\ r_k &= \beta_2 r_{k-1} + (1-\beta_2) \cdot \left[\nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}_k\right)\right]^2 \end{aligned} \tag{15}$$

$\boldsymbol{v}_k$ and r_k being estimates of the first and second moment, which are the mean and the uncentered variance of the gradients, respectively. To avoid the bias towards zero vectors for the initialized vectors $\boldsymbol{v}_k$ and r_k, we employ corrected first and second moment estimates as suggested by the authors of Adam:

$$\begin{aligned} \boldsymbol{v}_k &= \frac{\boldsymbol{v}_k}{1-\beta_1^t} \\ r_k &= \frac{r_k}{1-\beta_2^t} \end{aligned} \tag{16}$$

Subsequently, we use the Adam update rule given by

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \frac{\eta}{\sqrt{r_k} + \epsilon} \boldsymbol{v}_k \tag{17}$$

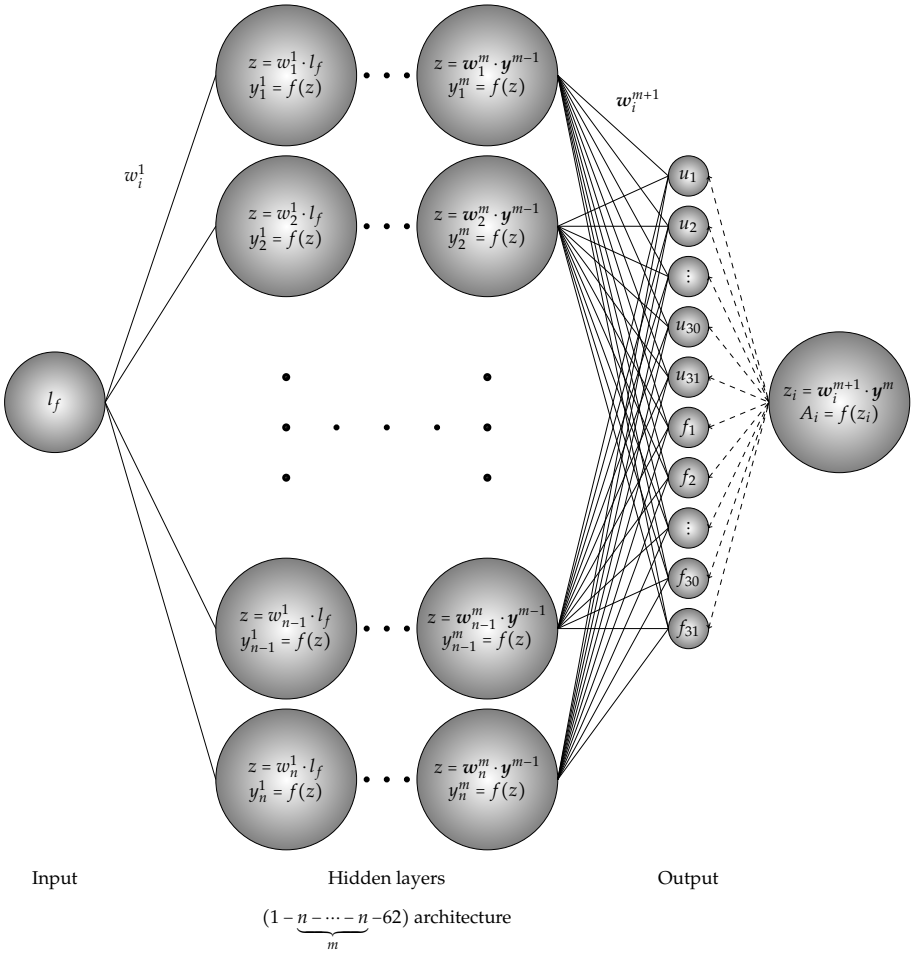


Figure 1. Activation function role in deep neural network.

3.2. Activation Functions

Let us consider a three-point bending beam as illustrated in Figure 2. We assume plane stress conditions and a beam thickness of $h = 0.05$ m. The material parameters are: Young's modulus $E = 20$ (GPa), Poisson's ratio $\nu = 0.2$, softening parameters $\alpha = 0.99$, $\beta = 500$, $\kappa_0 = 10^{-4}$. We employ the formulation for the equivalent strain as in Equation (4). The input of the process are locations l_f where the force f is applied which will be created by 120 different positions from $\frac{L}{2} - \Delta_p$ to $\frac{L}{2} + \Delta_p$ where $L = 2$ (m) denotes the length of the specimen and $2\Delta_p$ indicates the span of the various forces. For each input, the vector of output is formed by $[u_1, \cdots, u_{31}, f_1, \cdots, f_{31}]^T$ which (u_i, f_i) $\forall i = 1, \cdots, 31$ are points of loading deflection curve. Therefore, the matrix of output is a 62×120 matrix. The model of deep learning aims to minimize the loss between data and simulation values. The objective function is a loss function that comes from an approximate function f called *activation function* and the training data from simulation g:

$$loss(l_f) = \left\| g(l_f) - f(l_f) \right\| \tag{18}$$

where $g : \mathbb{R} \longrightarrow \mathbb{R}^{62}$ which is defined by $g(l_f) = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{f} \end{bmatrix} = [u_1, \cdots, u_{31}, f_1, \cdots, f_{31}]^T$ based on the gradient-enhanced damage models. The roles of activation functions are are illustrated in Figure 1 and the list of different types of mostly used activation functions is shown in Table 2.

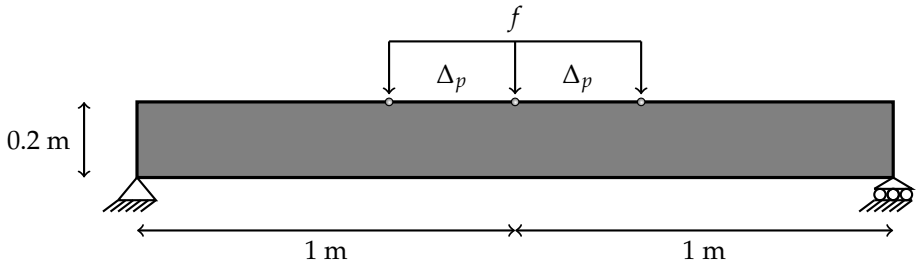


Figure 2. The various forces with respect to the location in three-point bending specimen.

Table 2. Activation functions.

Name	Equation	Derivative
Identity $(-\infty, +\infty)$	$f(z) = z$	$f'(z) = 1$
Logistic $(0, 1)$	$f(z) = \frac{1}{1+e^{-z}}$	$f'(z) = f(z)\,[1 - f(z)]$
SoftPlus $(0, +\infty)$	$f(z) = \ln(1 + e^{z})$	$f'(z) = \frac{1}{1+e^{-z}}$
TanH $(-1, 1)$	$f(z) = Tanh(z)$	$f'(z) = 1 - f(z)^2$
Rectified Linear Unit (ReLU) $(0, +\infty)$	$f(z) = \begin{cases} 0, & z < 0 \\ z, & z \geq 0 \end{cases}$	$f'(z) = \begin{cases} 0, & z < 0 \\ 1, & z \geq 0 \end{cases}$

3.3. Back-Propagation Algorithm

Though the back-propagation algorithm can be traced back to the 1970s, its popularity grew with the seminal paper of Rumelhart et al. [36]. It has become the 'backbone' of many efficient machine learning tools such as Pytorch or Tensorflow. The back-propagation algorithm basically minimizes the error function in weight space by the method of gradient descent. Mathematically, the input and output are in matrix form $(\boldsymbol{X}, \boldsymbol{Y})$ where $X \in \mathbb{R}^{d \times n}$ and $Y \in \mathbb{R}^{k \times n}$, d designate the number of input parameters, k is number of output parameters and n is number of training data. After calculating the output from the input of a mini-batch $\boldsymbol{X}$, the activation $\boldsymbol{y}^l$ at each hidden layer are saved where $l = 1, \cdots, m$ is the order of hidden layers. At the output layer, the derivative of the loss function with respect to z are calculated $\boldsymbol{e}^m = \frac{\partial J}{\partial \boldsymbol{z}^m}$ where $J = \frac{1}{n} \sum_{i=1}^{n} \left\| \overline{\boldsymbol{y}}_i - \boldsymbol{y}_i \right\|_2^2$. Hence, the gradient can be computed $\frac{\partial J}{\partial \boldsymbol{w}^m} = \boldsymbol{y}^{m-1} \boldsymbol{e}^m$. For $l = m - 1, \cdots, 1$, the derivatives $\boldsymbol{e}^l = \left(\boldsymbol{w}^{l+1} \boldsymbol{e}^{l+1} \right) \otimes f'(\boldsymbol{z}^l)$ are determined where the operator $\otimes$ means element wise product. Finally, the gradient is updated by $\frac{\partial J}{\partial \boldsymbol{w}^l} = \boldsymbol{y}^l \boldsymbol{e}^l$ which apparently requires the continuity and differentiability of the error function. Applicatively in the three-point bending specimen, the training data will be collected by the gradient-enhanced damage models and the training process based on the $(1 - 100 - 100 - 100 - 62)$ architecture in Figure 3. Because of the symmetric of the specimen, the loading deflection curve are quite similar in the left from $\frac{L}{2} - \Delta_p$ to $\frac{L}{2}$ and in the right from $\frac{L}{2}$ to $\frac{L}{2} + \Delta_p$ with regards to the middle point. Hence, it is a good idea to divide the data into two parts, the left part and the right part with respect to the middle point. Thus, the training is separated into two processes with 61 data for each case.

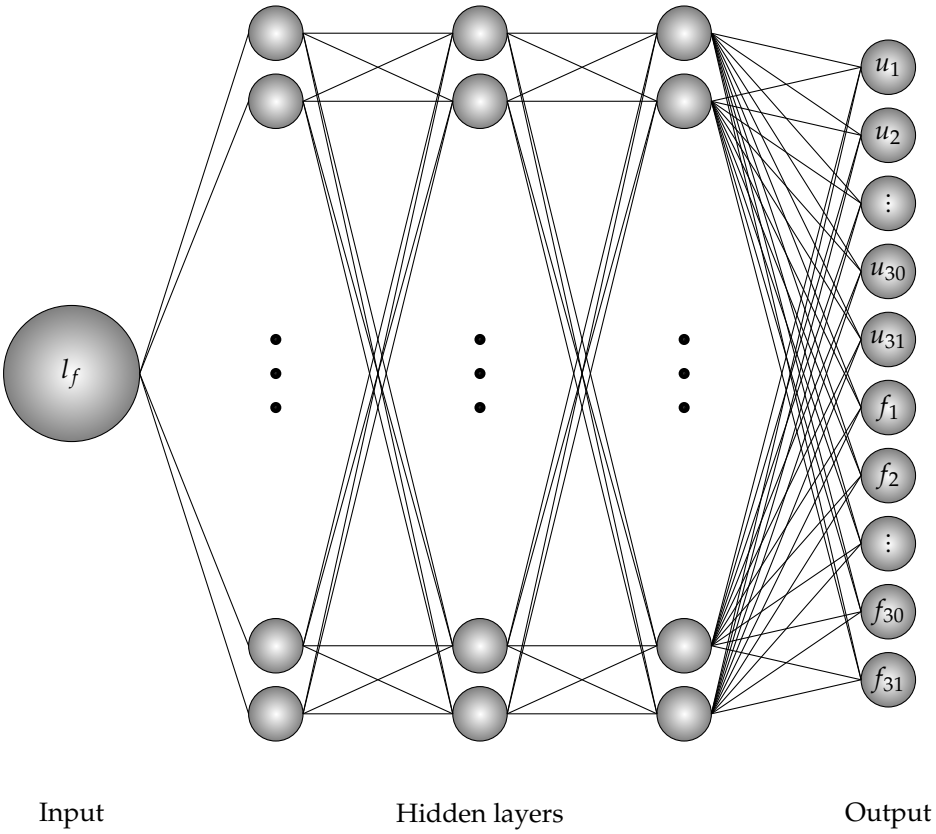


Figure 3. Deep neural network architecture for various location forces problem.

3.4. Scaled Layer

For the material problem which will be discussed in Section 5, the input data were built by different parameters. Due to the wide range of these, some data can be as large as Young's modulus or softening parameter β whereas other ones can be as small as the critical value of equivalent strain. This presents some limitations to the sensitive input and the training. To overcome these issues, a *scaled layer* based on a convex combination is introduced to reduce the overload input data and increase the tiny ones. Let s be a bijection from the range $[\min, \max]$ of a parameter to the interval $[0, 1]$ which is defined by $s(x) = \dfrac{x - min}{l}$ where min, max are the boundaries and l is the length of the parameter region. The role of the layer was applicable to both training and predicting process. The scale layer can be added at the input layer, the output layer or both of them.

4. Results and Testing

Consider back to the location force problem which was trained by Adam optimizer where Rectified Linear Unit (ReLU) activation function with back propagation algorithm was employed. The learning rate of the left data is $\eta_l = 3 \times 10^{-3}$ and of the right data is $\eta_r = 4 \times 10^{-3}$. For every period of a number of epochs (which is called *the learning rate drop period*), the learning rate is dropped base on a factor called *the learning rate drop factor* to let the jump steps are smaller after the period of iteration. This technique assists the convergent optimization. In this computation, the learning rate drop period is 100 epochs and the learning rate drop factor is 95%. The squared gradient decay factor is 0.99. The gradient decay factor is 0.95. The size of the mini-batch is 100 and the total of epochs is $10,000$. The training process for both left and right data are illustrated in Figure 4. The neural network architecture for this problem is $(1 - 100 - 100 - 100 - 62)$. Ten data were chosen randomly for testing. The loss values is

computed by root mean square error (RMSE) and by mini-batch loss (Loss) for comparison in Table 3. The testing of loading deflection curves by deep learning after training and by data for both left and right is illustrated in Figure 5.

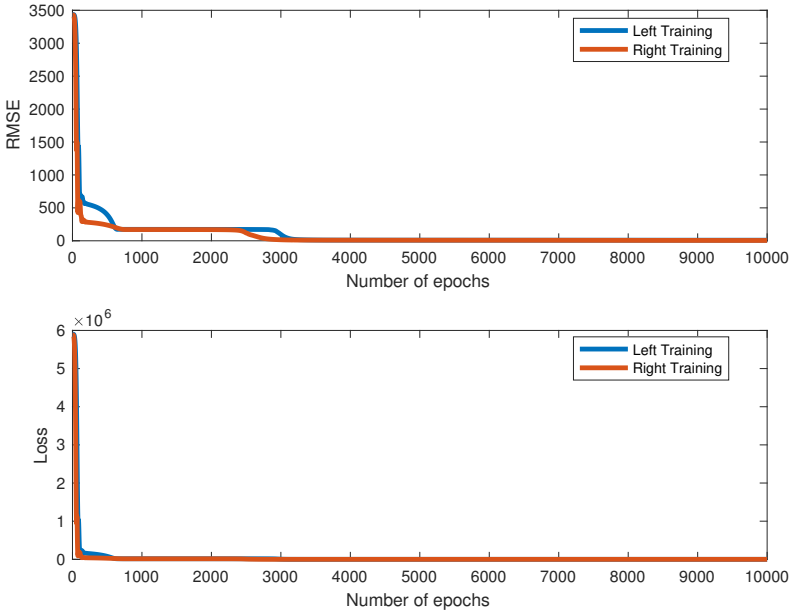
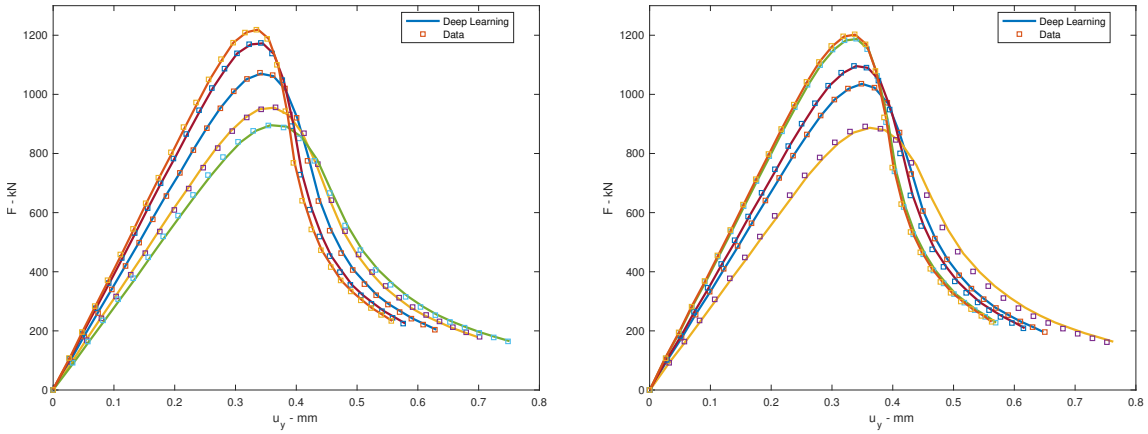


Figure 4. The training process for location-force data in three-point bending specimen.

Table 3. Loss values for both left and right data in the location-force problem.

Data	RMSE	Mini-Batch Loss
Left	9.03	40.8
Right	5.03	12.7



(**a**) For the left data (**b**) For the right data

Figure 5. The testing of loading deflection curves for location-force problem.

5. Numerical Examples

5.1. Dimensional Problem

We reconsider the three-point bending specimen that was subjected to an external force f whose various geometry and boundary condition of the specimen are shown in Figure 6. The thickness

h of the beam is 0.05 m and plane stress conditions are assumed. The input of the process are dimensional vectors $[H, L]$ where $H \in [0.1, 0.4]$ is the height and $L \in [1.0, 2.3636]$ is the length of the specimen. The data will be created by 156 different dimensional vectors where the heights H are linearly generated 12 times while the lengths L are created 13 times. The number of training data is 150 and six data for testing. In this example, to avoid the sensitivity of the data where the displacements $u \in [0.1, 0.8]$ and the forces can exceed 2500, the training process will be separated into two parts. One for displacement and one for forces whose architectures are illustrated in Figure 7. For each input, the vector of output is represented by $[u_1, \cdots, u_{31}]^T$ for displacements and $[f_1, \cdots, f_{31}]^T$ for forces which (u_i, f_i) $\forall i = 1, \cdots, 31$ are points of loading deflection curve. Therefore, the matrix of output is a 31×150 matrix. The training data will be collected by the gradient-enhanced damage models and the training process based on $(2 - \underbrace{100 - \cdots - 100}_{7} - 31)$ architecture.

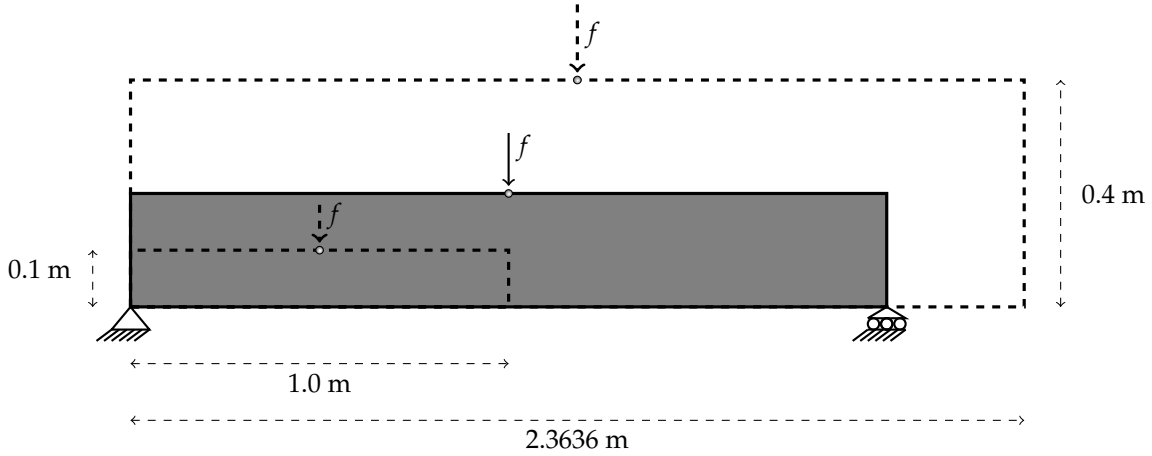
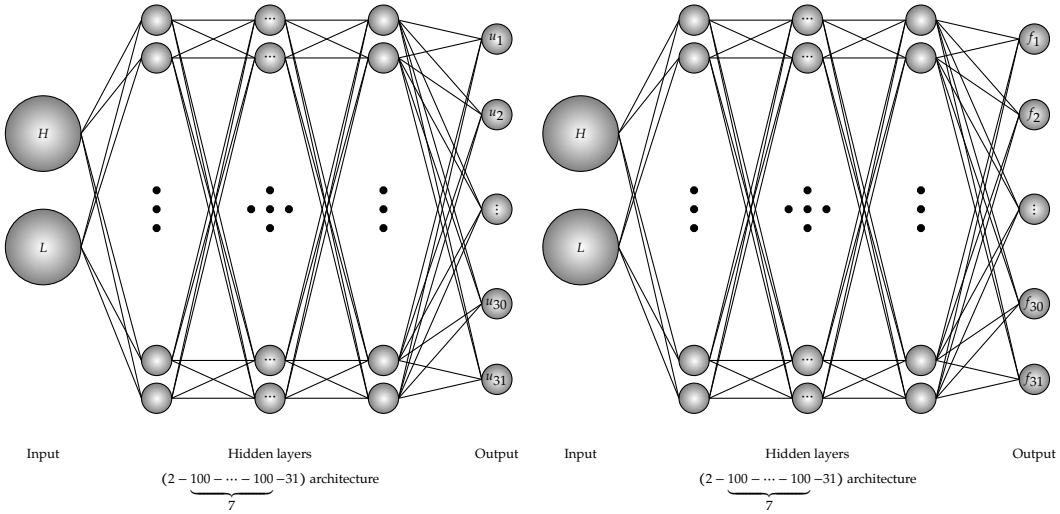


Figure 6. The various dimension for three-point bending specimen.



(**a**) Architecture for displacements (**b**) Architecture for forces

Figure 7. Deep neural network architecture for dimensional problem of the three-point bending specimen.

The problem was trained by Adam optimizer by the employment of Rectified Linear Unit (ReLU) activation function with back propagation algorithm. The learning rate of the displacement data is $\eta_u = 10^{-3}$ and of the force data is $\eta_f = 2 \cdot 10^{-3}$. The learning rate drop period is 100 epochs. The learning

rate drop factor is 99% for displacement training process and is 99.9% for force training process. The squared gradient decay factor is 0.99. The gradient decay factor is 0.95. The size of mini-batch is 100 and the total of epochs is 50,000 for force - training and is 5000 for displacement-training. The training processes are illustrated in Figure 8. Six data were chosen randomly for testing. The loss values are computed by root mean square error (RMSE) and by mini-batch loss (Loss) for comparison in Table 4. The testing of loading deflection curves by deep learning after train and by data are illustrated in Figure 9.

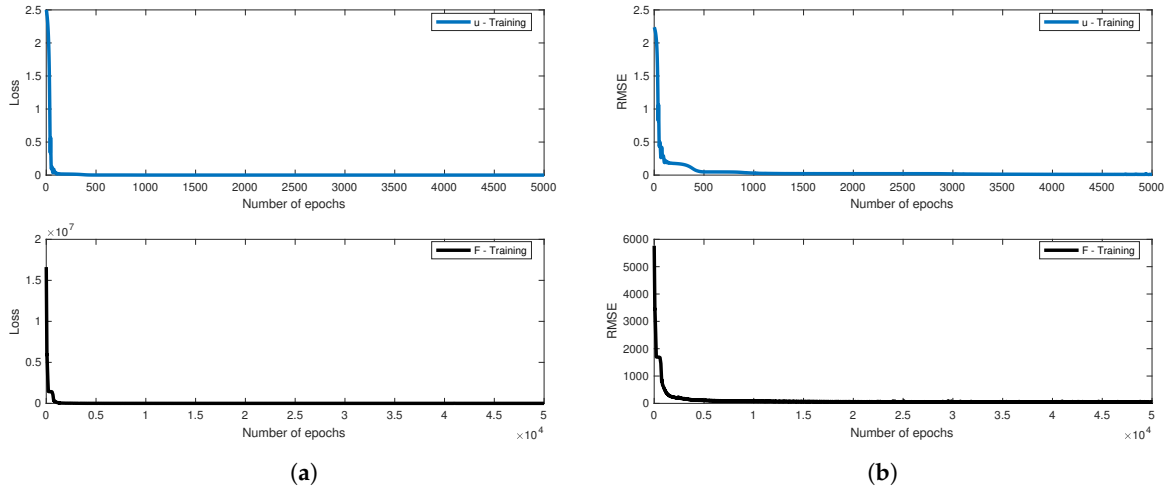


Figure 8. The training process for dimensional problem in the three-point bending specimen. (**a**) Mini-batch loss for dimensional training process; (**b**) RMSE for dimensional training process.

Table 4. Loss values for both displacement and forces training in dimensional problem.

Data	RMSE	Mini-Batch Loss
Displacements	0.0106	5.6221×10^{-5}
Forces	44.6711	997.7515

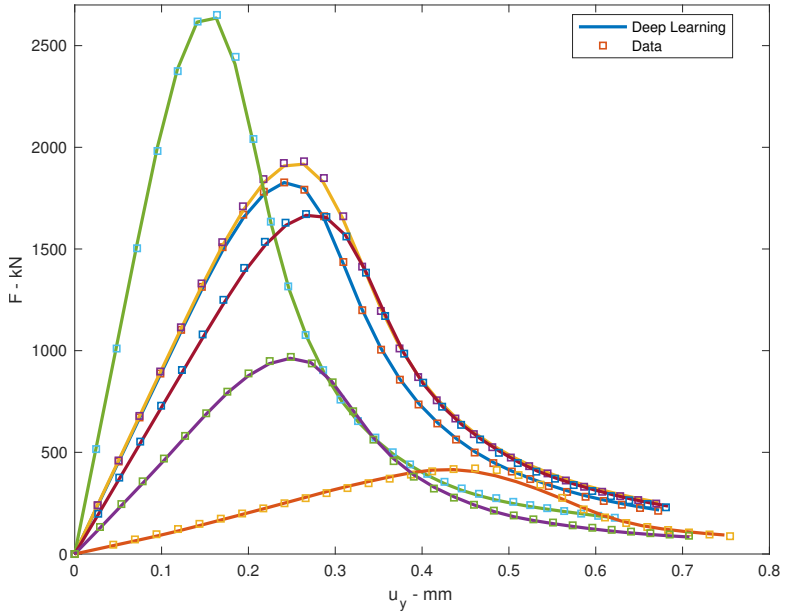


Figure 9. The testing of loading deflection curves for dimensional problem.

5.2. Material Parameter Problem

The three-point bending specimen is subjected to an external force f whose geometry and boundary condition of the specimen are shown in Figure 2. In this problem, the force is placed at the middle of the specimen. The thickness h of the beam is 0.05 m and plane stress conditions are assumed. The input of the process is material parameter vector $[E, \nu, \kappa, le, \alpha, \beta]$ where E is the Young's modulus, ν is the Poisson's ratio, κ is the critical value of equivalent strain, le is the characteristic length and α, β are softening parameters. The data is created by 4096 different parameter vectors where the parameters are taken in intervals based on the experienced average values in Table 5. In this example, to avoid the sensitivity of the data where the displacements $u \in [0.1, 0.8]$ and the forces can exceed 1400, and the training process is separated into two parts. One for displacements and one for forces whose architectures are similarly and illustrated in Figure 10. For each input, the vector of output is represented by $[u_1, \cdots, u_{31}]^T$ for displacements, $[f_1, \cdots, f_{31}]^T$ for forces, and (u_i, f_i) $\forall i = 1, \cdots, 31$ for points of the loading deflection curve. Therefore, the matrix of output is a 31×4096 matrix. The training data are collected by the gradient-enhanced damage models where the training process based on $(2 - \underbrace{100 - \cdots - 100}_{4} - 31)$ architecture for both displacement and force - training.

Table 5. The experienced average value and the range of material parameters.

Parameters	Average	Range
E	30×10^9	$\left[25 \times 10^9, 35 \times 10^9\right]$
ν	0.2	$[0.17, 0.23]$
κ	1.1×10^{-4}	$\left[8 \times 10^{-5}, 1.4 \times 10^{-4}\right]$
le	10^{-2}	$[0.005, 0.015]$
α	0.99	$[0.96, 0.99999]$
β	1800	$[1600, 2000]$

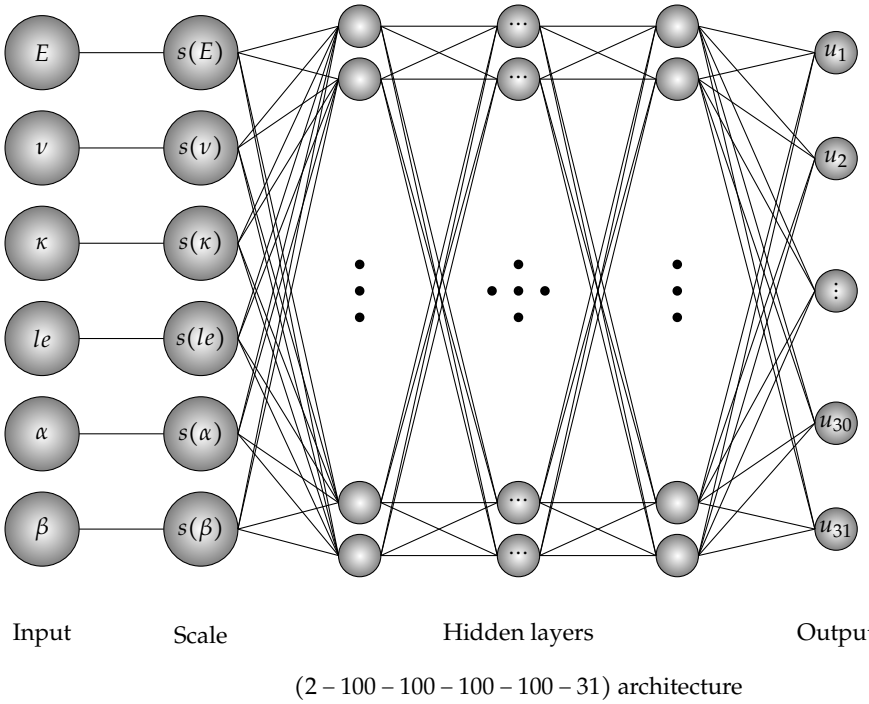


Figure 10. Deep neural network architecture for material parameter problem of the three-point bending specimen.

The problem was trained by Adam optimizer by the employment of Rectified Linear Unit (ReLU) activation function with back propagation algorithm. The learning rate of the displacement data is $\eta_u = 10^{-4}$ and of the force data is $\eta_f = 3 \times 10^{-4}$. The learning rate drop period is 100 epochs. The learning rate drop factor is 99%. The squared gradient decay factor is 0.99. The gradient decay factor is 0.95. The size of the mini-batch is 100 and the total of epochs is 5000. The training processes are illustrated in Figure 11. Six data were chosen randomly for testing. The loss values will be computed by root mean square error (RMSE) and by mini-batch loss (Loss) for comparison in Table 6. The testing of loading deflection curves by deep learning after training and by data are illustrated in Figure 12.

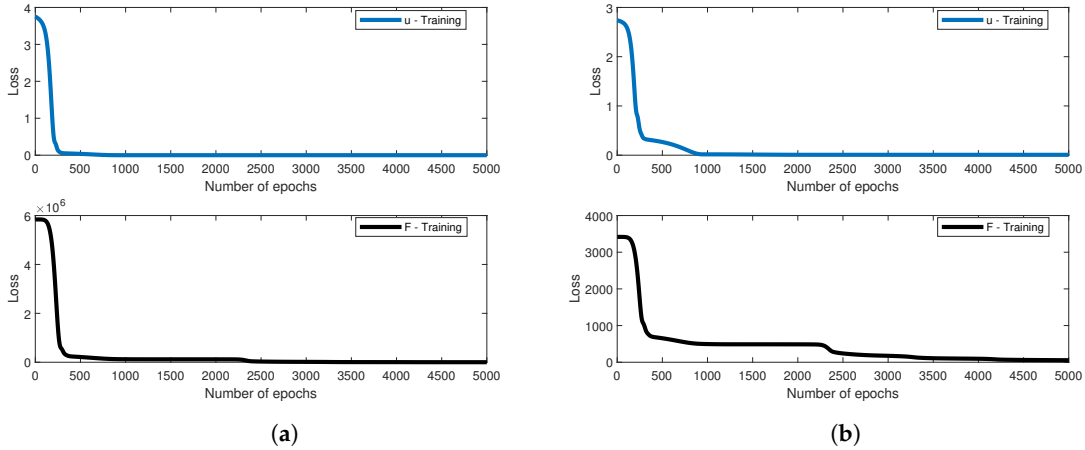


Figure 11. The training process for material problem in the three-point bending specimen. (**a**) Mini-batch loss for material training process; (**b**) RMSE for material training process.

Table 6. Loss values for both displacement and forces training in material problem

Data	RMSE	Mini-Batch Loss
Displacements	7.386×10^{-3}	2.728×10^{-5}
Forces	54.996	1.512×10^{3}

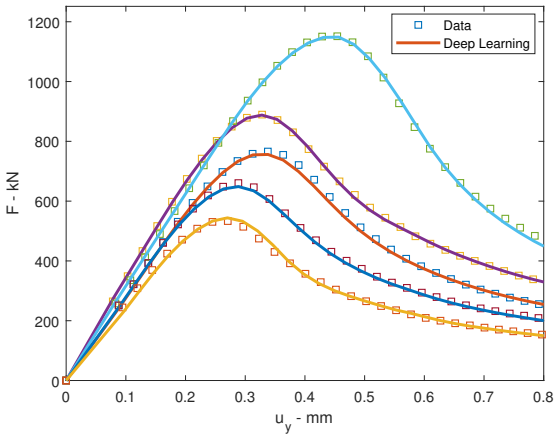


Figure 12. The testing of loading deflection curves for material problem.

5.3. L-Shape Specimen

The last example is an L-shaped specimen subjected to a distributed load as shown in Figure 13, which is also a classical benchmark problem used for instance to demonstrate the performance of Isogeometric Analysis (IGA) [37]. The thickness of the specimen is 20 cm and plane stress conditions are assumed. The material parameters are: Young's modulus E = 10 (GPa) and Poisson's ratio ν = 0.2.

The effective strain in Equation (6) and the damage law in Equation (5) are used with parameters $\kappa_0 = 4 \times 10^{-4}$, $\alpha = 0.98$ and $\beta = 80$. The non-local length scale is taken as $l_c = 5\sqrt{2} \approx 7.07$ (mm). The input of the training process are locations l_f on which the force f is applied. These locations are created by 123 different positions from 0 to Δ_p where Δ_p is the span of the various forces. In this example, to avoid the sensitivity of the data where the displacements $u \in [0, 3]$ and the forces can be more than 15000, the training process will be separated into two parts. One for displacements and one for forces whose architectures are illustrated in Figure 14. For each input, the vector of output is formed by $[u_1, \cdots, u_{36}]^T$ for displacements and $[f_1, \cdots, f_{36}]^T$ for forces which (u_i, f_i) $\forall i = 1, \cdots, 36$ are points of loading deflection curve. Therefore, the matrix of output is a 36×5000 matrix. The training data are collected by the gradient-enhanced damage models where the training process based on $(2 - \underbrace{100 - \cdots - 100}_{4} - 36)$ architecture for displacement-training and $(2 - \underbrace{100 - \cdots - 100}_{4} - 36)$ architecture for force-training.

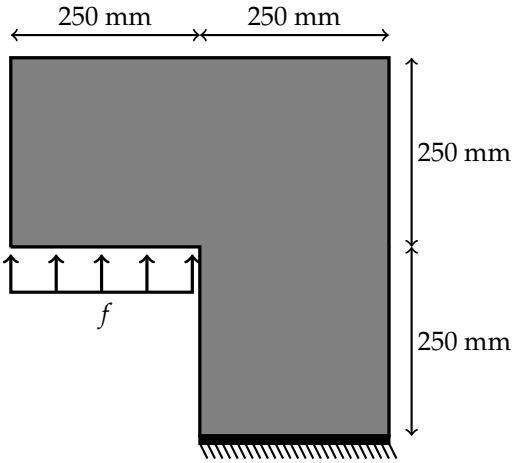


Figure 13. The various forces with respect to the location in L-shape specimen.

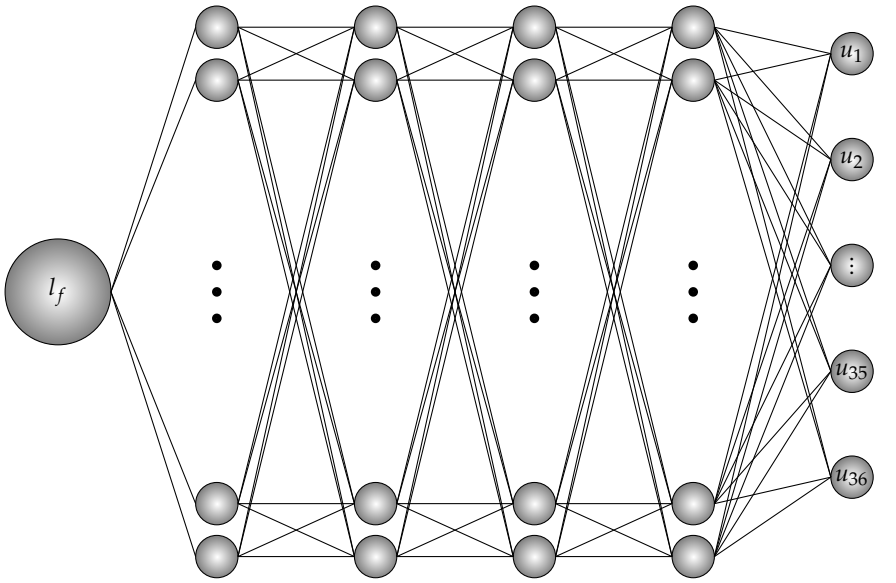
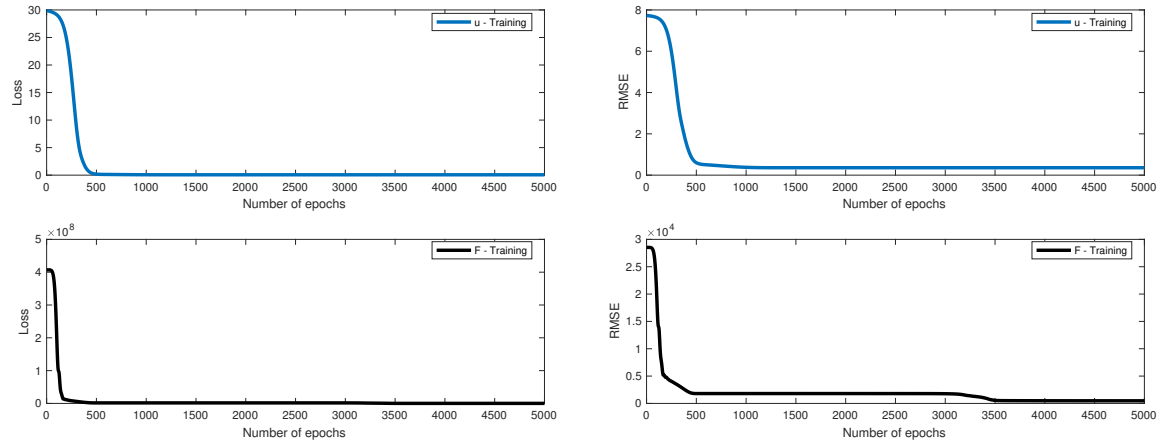


Figure 14. Deep neural network architecture for force locations problem of the L-shape specimen.

The problem was trained by Adam optimizer by the employment of Rectified Linear Unit (ReLU) activation function with back propagation algorithm. The learning rate of the displacement data is $\eta_u = 10^{-4}$ and of the force data is $\eta_f = 2 \times 10^{-3}$. The learning rate drop period is 100 epochs. The learning rate drop factor is 99%. The squared gradient decay factor is 0.99. The gradient decay factor is 0.95. The size of mini-batch is 100 and the total of epochs is 5000. The training processes are illustrated in Figure 15. Five randomly data were chosen for testing. The loss values are computed by root mean square error (RMSE) and by mini-batch loss (Loss) for comparison in Table 7. The testing of loading deflection curves by deep learning after training and by data are illustrated in Figure 16.



(**a**) (**b**)

Figure 15. The training process for shear force problem in the L-hape specimen. (**a**) Mini-batch loss for L-shape shear force training process; (**b**) RMSE for L-hape shear force training process.

Table 7. Loss values for both displacement and forces training in L-shape problem.

Data	RMSE	Mini-Batch Loss
Displacements	0.36	6.4×10^{-2}
Forces	479.18	114809.0

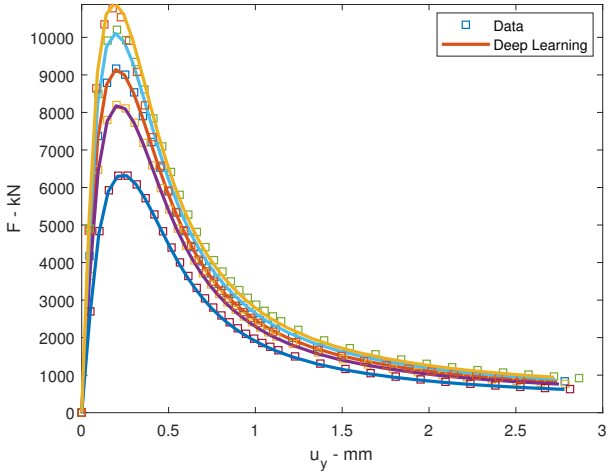


Figure 16. The testing of loading deflection curves for L-hape problem.

6. Conclusions

This paper introduced a deep learning technique in the context of artificial neural networks for predicting loading deflection curves of structures under gradient-enhanced damage responses. The neural network has been trained based on finite element simulations. We conducted our approach in the form of numerical experiments, using various shapes of specimen as well as inputs. The predictions based on material changes are much more challenging and the number of input parameters will trigger big data in the training phase. The Adam optimizer and Rectified Linear Unit activation function produced the best result for training. The major contribution of this study is to integrate deep learning into computational mechanics. The key feature is how to choose training parameters and deep neural network architecture. Our research is potentially competent for application to a wide range of complex practical engineering problems. In the next step, we intend to develop and apply sufficient transfer learning algorithms, which is important for computational efficiency.

Author Contributions: Conceptualization, X.Z., L.C.N., H.N.-X., N.A., and T.R.; methodology, X.Z., L.C.N., H.N.-X., N.A., and T.R.; software, L.C.N.; validation, L.C.N.; formal analysis, L.C.N.; investigation, X.Z., L.C.N., H.N.-X., N.A., and T.R.; writing, X.Z., L.C.N., H.N.-X., N.A., and T.R.; visualization, L.C.N.; and supervision, X.Z., H.N.-X., N.A., and T.R. All authors have read and agreed to the published version of the manuscript.

Funding: The authors extend their appreciation to the Distinguished Scientist Fellowship Program (DSFP) at King Saud University for funding this work.

Acknowledgments: The authors extend their appreciation to the Distinguished Scientist Fellowship Program (DSFP) at King Saud University for funding this work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hinton, G.E.; Osindero, S.; Teh, Y.W. A fast learning algorithm for deep belief nets. *Neural Comput.* **2006**, *18*, 1527–1554. [CrossRef] [PubMed]
2. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010; pp. 807–814.
3. Hinton, G.E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R.R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv* **2012**, arXiv:1207.0580.
4. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436. [CrossRef]
5. Adeli, H. Neural networks in civil engineering: 1989–2000. *Comput.-Aided Civ. Infrastruct. Eng.* **2001**, *16*, 126–142. [CrossRef]
6. Deo, R.C. Machine learning in medicine. *Circulation* **2015**, *132*, 1920–1930. [CrossRef] [PubMed]
7. Angermueller, C.; Pärnamaa, T.; Parts, L.; Stegle, O. Deep learning for computational biology. *Mol. Syst. Biol.* **2016**, *12*, 878. [CrossRef]
8. Park, Y.; Kellis, M. Deep learning for regulatory genomics. *Nat. Biotechnol.* **2015**, *33*, 825. [CrossRef]
9. Goh, G.B.; Siegel, C.; Vishnu, A.; Hodas, N.O.; Baker, N. Chemception: A deep neural network with minimal chemistry knowledge matches the performance of expert-developed qsar/qspr models. *arXiv* **2017**, arXiv:1706.06689.
10. Ravì, D.; Wong, C.; Deligianni, F.; Berthelot, M.; Andreu-Perez, J.; Lo, B.; Yang, G.Z. Deep learning for health informatics. *IEEE J. Biomed. Health Inf.* **2017**, *21*, 4–21. [CrossRef]
11. Samaniego, E.; Anitescu, C.; Goswami, S.; Nguyen-Thanh, V.M.; Guo, H.; Hamdia, K.; Zhuang, X.; Rabczuk, T. An energy approach to the solution of partial differential equations in computational mechanicsvia machine learning: Concepts, implementation and applications. *Comput. Methods Appl. Mech. Eng.* **2020**, *362*, 112790. [CrossRef]
12. Furukawa, T.; Yagawa, G. Implicit constitutive modelling for viscoplasticity using neural networks. *Int. J. Numer. Methods Eng.* **1998**, *43*, 195–219. [CrossRef]
13. Lefik, M.; Boso, D.; Schrefler, B. Artificial neural networks in numerical modelling of composites. *Comput. Methods Appl. Mech. Eng.* **2009**, *198*, 1785–1804. [CrossRef]
14. Rafiee, R.; Habibagahi, M.R. Evaluating mechanical performance of GFRP pipes subjected to transverse loading. *Thin-Walled Struct.* **2018**, *131*, 347–359. [CrossRef]

15. Rafiee, R.; Torabi, M.A. Stochastic prediction of burst pressure in composite pressure vessels. *Compos. Struct.* **2018**, *185*, 573–583. [CrossRef]
16. Rafiee, R.; Torabi, M.A.; Maleki, S. Investigating structural failure of a filament-wound composite tube subjected to internal pressure: Experimental and theoretical evaluation. *Polym. Test.* **2018**, *67*, 322–330. [CrossRef]
17. Ghaboussi, J.; Pecknold, D.A.; Zhang, M.; Haj-Ali, R.M. Autoprogressive training of neural network constitutive models. *Int. J. Numer. Methods Eng.* **1998**, *42*, 105–126. [CrossRef]
18. Oeser, M.; Freitag, S. Modeling of materials with fading memory using neural networks. *Int. J. Numer. Methods Eng.* **2009**, *78*, 843–862. [CrossRef]
19. Ootao, Y.; Kawamura, R.; Tanigawa, Y.; Imamura, R. Optimization of material composition of nonhomogeneous hollow sphere for thermal stress relaxation making use of neural network. *Comput. Methods Appl. Mech. Eng.* **1999**, *180*, 185–201. [CrossRef]
20. Gawin, D.; Lefik, M.; Schrefler, B. ANN approach to sorption hysteresis within a coupled hygro-thermo-mechanical FE analysis. *Int. J. Numer. Methods Eng.* **2001**, *50*, 299–323. [CrossRef]
21. Yun, G.J.; Ghaboussi, J.; Elnashai, A.S. Self-learning simulation method for inverse nonlinear modeling of cyclic behavior of connections. *Comput. Methods Appl. Mech. Eng.* **2008**, *197*, 2836–2857. [CrossRef]
22. Lee, S.; Ha, J.; Zokhirova, M.; Moon, H.; Lee, J. Background Information of Deep Learning for Structural Engineering. *Arch. Comput. Methods Eng.* **2018**, *25*, 121–129. [CrossRef]
23. Mazars, J.; Pijaudier-Cabot, G. Continuum damage theory—Application to concrete. *J. Eng. Mech.* **1989**, *115*, 345–365. [CrossRef]
24. Peerlings, R.; De Borst, R.; Brekelmans, W.; Geers, M. Gradient-enhanced damage modelling of concrete fracture. *Mech. Cohesive-frictional Mater.: Int. J. Exp., Model. Comput. Mater. Struct.* **1998**, *3*, 323–342. [CrossRef]
25. De Vree, J.; Brekelmans, W.; Van Gils, M. Comparison of nonlocal approaches in continuum damage mechanics. *Comput. Struct.* **1995**, *55*, 581–588. [CrossRef]
26. Peerlings, R.H.J.; De Borst, R.; Brekelmans, W.A.M.; De Vree, J. Gradient enhanced damage for quasi-brittle materials. *Int. J. Numer. Methods Eng.* **1996**, *39*, 3391–3403. [CrossRef]
27. Bazant, Z.P.; Pijaudier-Cabot, G. Nonlocal continuum damage, localization instability and convergence. *J. Appl. Mech.* **1988**, *55*, 287–293. [CrossRef]
28. Lasry, D.; Belytschko, T. Localization limiters in transient problems. *Int. J. Solids Struct.* **1988**, *24*, 581–597. [CrossRef]
29. De Borst, R.; Mühlhaus, H.B. Gradient-dependent plasticity: Formulation and algorithmic aspects. *Int. J. Numer. Methods Eng.* **1992**, *35*, 521–539. [CrossRef]
30. Schreyer, H.L.; Chen, Z. One-dimensional softening with localization. *J. Appl. Mech.* **1986**, *53*, 791–797. [CrossRef]
31. Triantafyllidis, N.; Aifantis, E.C. A gradient approach to localization of deformation. I. Hyperelastic materials. *J. Elast.* **1986**, *16*, 225–237. [CrossRef]
32. Bažant, Z.P.; Pijaudier-Cabot, G. Measurement of characteristic length of nonlocal continuum. *J. Eng. Mech.* **1989**, *115*, 755–767. [CrossRef]
33. MuḶhlhaus, H.; Aifantis, E. A variational principle for gradient principle. *Int. J. Solids Struct.* **1991**, *28*, 845–857. [CrossRef]
34. Nesterov, Y. Gradient methods for minimizing composite objective function. *Math. Program.* **2013**, *140*, 125–161. [CrossRef]
35. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
36. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533. [CrossRef]
37. Kuhl, K. Numerical Models for Cohesive Frictional Materials. Ph.D. Thesis, Stuttgart University, Stuttgart, Germany, 2000.