HHS Public Access

Author manuscript

Comput Mech. Author manuscript; available in PMC 2024 July 01.

Published in final edited form as:

Comput Mech. 2023 July ; 72(1): 173–194. doi:10.1007/s00466-023-02293-z.

HiDeNN-FEM: A seamless machine learning approach to nonlinear finite element analysis

Yingjian Liu[1], **Chanwook Park**[2], **Ye Lu**[2], **Satyajit Mojumder**[3], **Wing Kam Liu**[2], **Dong Qian**[1,*]

[1]Department of Mechanical Engineering, The University of Texas at Dallas, Richardson, TX 75080, USA

[2]Department of Mechanical Engineering, Northwestern University, 2145 Sheridan Rd., Evanston IL 60208-3111, USA

[3]Theoretical and Applied Mechanics, Northwestern University, 2145 Sheridan Rd, Evanston, IL 60208-3111, USA

Abstract

The hierarchical deep-learning neural network (HiDeNN) (Zhang et al, Computational Mechanics, 67:207–230) provides a systematic approach to constructing numerical approximations that can be incorporated into a wide variety of Partial differential equations (PDE) and/or Ordinary differential equations (ODE) solvers. This paper presents a framework of the nonlinear finite element based on HiDeNN approximation (nonlinear HiDeNN-FEM). This is enabled by three basic building blocks employing structured deep neural networks: 1) A partial derivative operator block that performs the differentiation of the shape functions with respect to the element coordinates, 2) An r-adaptivity block that improves the local and global convergence properties and 3) A materials derivative block that evaluates the material derivatives of the shape function. While these building blocks can be applied to any element, specific implementations are presented in 1D and 2D to illustrate the application of the deep learning neural network. Two-step optimization schemes are further developed to allow for the capabilities of r-adaptivity and easy integration with any existing FE solver. Numerical examples of 2D and 3D demonstrate that the proposed nonlinear HiDeNN-FEM with r-adaptivity provides much higher accuracy than regular FEM. It also significantly reduces element distortion and suppresses the hourglass mode.

Keywords

Hierarchical deep neural network; nonlinear finite element method; r-adaptivity; shape function; data-driven

1 Introduction

As a major branch of artificial intelligence, Machine Learning (ML) involves the development of algorithms to “learn” based on given information (typically known as training data) and then make predictions by applying the “knowledge” that is learned.

*Corresponding author, dong.qian@utdallas.edu.

Such type of knowledge can be cast in different forms, e.g., using neural networks (NN). NN mimics the biological structure of a neuron network and consists of combinations of neurons as the fundamental unit that interacts with the other neurons through information passage. Each neuron processes the information passed through by first assigning the weight to each connection, summing up the weighted inputs, and adding a bias. It will then process this input through a so-called activation function and pass the processed result to the other neurons that connect to it. Deep Neural Network (DNN) [1] is a special case of NN in which neurons are arranged in multiple layers: there is an input layer that feeds the training data, an output layer that provides the prediction, and in between, there are multiple internal layers that process the data using combinations of weights, biases, and activation functions.

DNN has found its applications in areas such as image analysis [2–5], language processing [6], medical assistance [7–10], strategic decision-making [11], and material design [12, 13] and demonstrated its outstanding data processing capabilities. There is a continuing interest in applying DNN to solve ordinary and partial differential equations (ODE/PDE) [14–18] that govern broad engineering and science applications. The main motivations are two folds: first, DNN is capable of establishing the so-called universal approximation [19], which can be employed to build nonlinear approximations with arbitrary orders of resolution. These are generally difficult to construct using single-scale approaches such as the finite element method (FEM). Second, DNN employs back and forward propagation to “learn” and “predict”. Advanced algorithms and hardware based on parallel computing architectures are widely available to accelerate these processes and are continuously being improved. The existing approaches can be generally divided into two categories based on whether they are purely data-driven or not. In a purely data-driven approach, data collected from experiments or simulations are fed to the DNN to capture the nonlinear mapping between the input and output [20–23]. In most cases, large amounts of data are required for accomplishing good prediction accuracy. This is a limiting factor due to the cost or time it takes to generate the data. On the other hand, the method may also suffer from overfitting if the data provided does not fully represent the whole spectrum of features. Although regularization methods have been established to alleviate the overfitting issue to some extent, there are no general approaches available. In addition, general training algorithms [24–26] such as those based on the gradient descent approach are not always robust due to the lack of insight into the physics of the problem.

Given the difficulties in directly applying a purely data-driven approach, there has been a continuing interest in integrating DNN with mechanistic principles that govern the application. One of the successful examples is the physics-informed neural network (PINN) in which loss functions based on evaluating the residual errors associated with the governing differential equations are introduced to accelerate the convergence of data training [27]. For general science and engineering applications, the application of DNN remains a great challenge for problems that are featured by: 1) High computational cost of purely physics-based model; 2) Lack of insight on the physics governing the application, and 3) High volume of data that is beyond the reach of the existing data processing capability. In light of these challenges, Zhang et al have recently proposed the Hierarchical Deep Learning Neural Network (HiDeNN) framework[28]. Unlike most of the existing approaches, HiDeNN establishes a hierarchical and structured framework to directly integrate the neural network

structure with the numerical approximation. Three elementary building blocks that perform the operations of linear transformation, multiplication, and inversion are introduced to generate the DNN representations of the commonly used interpolation functions, such as those based on FEM, Lagrangian polynomial, spline functions, reproducing kernel meshfree shape functions, NURBS, and Isogeometric analysis (IGA).

One of the key features of HiDeNN is that the weights and biases of DNN are functions of the nodal positions. As such, training of the HiDeNN leads to optimized nodal positions. This is also known as r-adaptivity in the context of finite element-based interpolations (HiDeNN-FEM). When combined with a physics-based loss function such as the potential energy of the system, the robustness of HiDeNN-FEM has been illustrated for 1D and 2D linear elasticity problems [29]. It was also shown that HiDeNN-FEM can be further enhanced with reduced-order modeling using proper generalized decomposition (PGD), leading to HiDeNN-PGD [30, 31]. In this work, we present the application of HiDeNN-FEM to nonlinear problems in solid mechanics. For these types of problems, Lagrangian meshes are commonly used in which nonlinearity arises due to large deformation and/or material nonlinearity. The shape functions are typically constructed in the parent (element) configuration and expressed in terms of the parent (element) coordinates. Correspondingly, three building blocks employing HiDeNN are introduced: 1) A partial derivative operator block that performs the differentiation of the shape functions with respect to the element coordinate, 2) An r-adaptivity block that improves the local and global convergence properties and 3) A materials derivative block that evaluates the material derivatives of the shape function (in the case of total Lagrangian formulation). While these building blocks are generally applicable for any type of finite element shape functions, specific cases in 1D and 2D are presented to illustrate the application. We further show through 2D and 3D examples that convergence can be enhanced with a physics-based loss function that employs either potential energy or out-of-balance force.

The rest of the paper is organized as follows. In Section 2, the basic formulation of HiDeNN-FEM is reviewed. In Section 3, we outline the three building blocks and the process to construct the HiDeNN-FEM shape functions. This is followed by a discussion on the general solution processes for nonlinear problems in section 4. Section 5 presents results and discussions on several benchmark problems in both 2D and 3D. Finally, conclusions are drawn in Section 6.

2 The Basic Formulation of HiDeNN-FEM

We first introduce the basic notations that are commonly used for representing DNN. The basic unit of the DNN is an artificial neuron that is also known as perceptron as shown in Figure 1. Perceptron takes multiple inputs given as $x_1, x_2, \ldots, x_n$. Each input is multiplied by a weight, i.e., w_i with $i = 1, \ldots, n$. The weighted inputs are then summed and a bias $\mathbf{b}$ is added to give $\mathbf{wx} + \mathbf{b}$, which serves as an input to the activation function $\mathscr{A}$. The result $\mathscr{A}(\mathbf{wx} + \mathbf{b})$ is assigned as the output of the artificial neuron. Many choices of the activation function $\mathscr{A}$ have been proposed based on the applications. For details we refer to the introduction in [32]. A DNN consists of an arrangement of the perceptrons in multiple layers including at least one internal (hidden) layer (Figure 1). To describe the information passing within the

DNN, we introduce the notation $w_{j,k}^{l-1,l}$ as the weight from the j-th neuron in the $(l-1)$-th layer to the k-th neuron in the l-th layer and b_j^l as the bias of the j-th neuron in the l-th layer.

In HiDeNN-FEM, a structured DNN is developed to realize the interpolation of the shape functions. This is accomplished by establishing three basic building blocks as shown in Figure 2, i.e., linear, multiplication, and inversion. The linear building block establishes the piecewise linear function, defined as

$$L(x; x_A, x_B, y_A, y_B) = \begin{cases} y_A, & x < x_A, \\ \dfrac{y_B - y_A}{x_B - x_A}(x - x_A) + y_A, & x_A \le x \le x_B, \\ y_B, & x > x_B, \end{cases}$$

The multiplication building block M performs the multiplication of two functions F_1, F_2 that are represented in DNN, i.e., $M(F_1, F_2) = F_1 \cdot F_2$. The inversion building block V provides the quotient of two DNN-represented functions F_1, F_2, i.e., $V(F_1, F_2) = F_2/F_1$. For the detailed implementation of using DNN to establish these building blocks, we refer to [28].

3 Nonlinear HiDeNN-FEM

3.1 A brief introduction to nonlinear FEM

The method outlined below applies to both total Lagrangian (TL) and updated Lagrangian formulation (UL). In this work we adopted the total Lagrangian (TL) formulation [33] to illustrate the application of HiDeNN. We first introduce **X** as the material coordinate and **x** as the spatial coordinate. The weak form of the momentum equation is given as

$$\int_{\Omega_0} \rho_0 \ddot{\mathbf{u}} \cdot \delta\mathbf{u} d\Omega + \int_{\Omega_0} \mathbf{P} : \delta\mathbf{F}^{\mathrm{T}} d\Omega - \int_{\Omega_0} \mathbf{b} \cdot \delta\mathbf{u} d\Omega - \int_{\Gamma_{t0}} \mathbf{T} \cdot \delta\mathbf{u} d\Gamma = 0 \tag{1}$$

in which $\mathbf{u} = \mathbf{x} - \mathbf{X}$ is the displacement and the superimposed dot denotes the time derivative. The symbol δ represents variational operator and $\delta\mathbf{u}$ is the virtual displacement, ρ_0 is the mass density defined in the initial configuration Ω_0, **P** is the 1st Piola-Kirchhoff (nominal) stress and **F** is deformation gradient, **b** is the body force and **T** is traction applied on boundary Γ_{t0}.

To solve Eq.(1) using FEM, we introduce Lagrangian mesh and three configurations (Figure 3):

1. The parent element domain □ on which the shape function approximation is built. The element coordinates are given as $\boldsymbol{\xi}^e$ with e the element index;
2. The initial (reference) configuration Ω_0^e with material coordinate **X**;
3. The current configuration Ω^e with spatial coordinate **x**;

To describe the motion, the mapping from the initial to the current configuration is introduced as $\mathbf{x} = \phi(\mathbf{X}, t)$. Additionally, the initial and current configurations are mapped

from the parent domain, given as $\mathbf{X} = \mathbf{X}(\boldsymbol{\xi})$ and $\mathbf{x} = \mathbf{x}(\boldsymbol{\xi}, t)$, respectively. With the shape functions constructed in the parent domain, these last two mappings are approximated as $\mathbf{X}(\boldsymbol{\xi}) = \mathbf{X}_I N_I(\boldsymbol{\xi})$ and $\mathbf{x}(\boldsymbol{\xi}) = \mathbf{x}_I N_I(\boldsymbol{\xi})$ with $N_I(\boldsymbol{\xi})$ the shape function defined at the node I and evaluated at coordinate ξ. Repeated nodal index of I indicates the summation within the element. Substitution of the shape function approximation into the weak form of the TL formulation in Eq.(1) gives the discretized form of the momentum equation, given as

$$\mathbf{M}\ddot{\mathbf{u}} = \mathbf{f}^{ext} - \mathbf{f}^{int} \tag{2}$$

with

$$\mathbf{f}^{int} = f_{iI}^{int} = \int_{\Omega_0} \frac{\partial N_I}{\partial X_j} P_{ji} d\Omega_0 = \int_{\Omega_0} \left(B_{Ij}^0\right)^T P_{ji} d\Omega_0 \tag{3}$$

$$\mathbf{f}^{ext} = f_{iI}^{ext} = \int_{\Omega_0} N_I \rho_0 b_i d\Omega_0 + \int_{\Gamma_{t_i}^0} N_I \bar{t}_i^0 d\Gamma_0 \tag{4}$$

$$\mathbf{M} = M_{\mathrm{ijIJ}} = \delta_{ij} \int_{\Omega_0} \rho_0 N_I N_J d\Omega_0 \tag{5}$$

in which ρ_0 is the mass density in the initial configuration, b_i is the body force, $\bar{t}_i^0$ is the traction applied over the natural boundary $\Gamma_{t_i}^0$ and the shape function derivative matrix is defined through $\mathscr{B}_{jI}^0 = \frac{\partial N_I}{\partial X_j}$ in Eq.(3). In this paper, we will focus on application of HiDeNN-FEM to nonlinear static problems so that the inertia terms in Eq.(2) are neglected. Extension to nonlinear dynamic problems will be described in a separate publication.

3.2 HiDeNN-FEM building blocks for nonlinear FEM

Computing the internal force term in Eq.(3) requires evaluation of the shape function derivative. This is carried through the chain rule,

$$\mathscr{B}_{jI}^0 = \frac{\partial N_I}{\partial X_j} = \frac{\partial N_I}{\partial \mathbf{X}} = \frac{\partial N_I}{\partial \boldsymbol{\xi}} \left(\frac{\partial \mathbf{X}}{\partial \boldsymbol{\xi}}\right)^{-1} \tag{6}$$

The matrix notation of Eq.(6) is shown here for the case of a general 2D element in which the element coordinates are given as ξ, η and material coordinates are X, Y. For a given nodal index I, we have

$$\frac{\partial N_I}{\partial \mathbf{X}} = N_{I,\mathbf{X}}^T = \begin{bmatrix} N_{I,X} & N_{I,Y} \end{bmatrix} = \begin{bmatrix} N_{I,\xi} & N_{I,\eta} \end{bmatrix} \begin{bmatrix} X_{,\xi} & X_{,\eta} \\ Y_{,\xi} & Y_{,\xi} \end{bmatrix}^{-1} \tag{7}$$

Or equivalently

$$N_{I,\mathbf{X}} = \begin{Bmatrix} N_{I,X} \\ N_{I,Y} \end{Bmatrix} = \begin{bmatrix} X_{,\xi} & X_{,\eta} \\ Y_{,\xi} & Y_{,\eta} \end{bmatrix}^{-T} \begin{Bmatrix} N_{I,\xi} \\ N_{I,\eta} \end{Bmatrix} = \begin{bmatrix} X_{,\xi} & Y_{,\xi} \\ X_{,\eta} & Y_{,\eta} \end{bmatrix}^{-1} \begin{Bmatrix} N_{I,\xi} \\ N_{I,\eta} \end{Bmatrix} \tag{8}$$

Once the shape function derivatives are evaluated from Eq.(8) for each nodal index *I*, the matrix notation of Eq.(6) is given as

$$\mathscr{B}^0_{jI} = \begin{bmatrix} N_{1,\mathbf{X}} & N_{2,\mathbf{X}} & \dots & \end{bmatrix} \tag{9}$$

For implementation in HiDeNN-FEM, the operations outlined above to evaluate $\mathscr{B}^0_{jI}$ can be realized with three building blocks as shown in Figure 4. Using the element coordinate as inputs, the block of partial derivative operator evaluates the matrix $\mathbf{D}_N$ that contains the shape function derivatives with respect to the element coordinates. The r-adaptive block computes the Jacobian of the mapping $\mathbf{X} = \mathbf{X}(\boldsymbol{\xi})$. Subsequently, the material derivatives are evaluated in the third block, which gives the matrix that contains the shape function derivative $\mathscr{B}^0_{jI}$.

3.2.1 Construction of shape functions and derivatives in 1D elements—

We first illustrate the construction of the shape functions and derivatives using the three building blocks for the case of 1D linear element. Configurations of the element in the physical and parent domain are shown in Figure 5a. The shape functions at nodes 1 and 2 are $N_1 = \frac{1}{2}(1-\xi)$, $N_2 = \frac{1}{2}(1+\xi)$ with $-1 \le \xi \le 1$. Figure 5b illustrates the DNN representation of the linear shape function in which two hidden layers are introduced. The element coordinate ξ is employed as the input and the activation function $\mathscr{A}_1(x) = x$ is introduced. The output of the DNN are the values of the shape functions defined at the two nodes and evaluated at the element coordinate ξ. According to Figure 5a, the DNN representations of the shape functions are given as

$$\begin{aligned} N_1(\xi; \mathbf{W}, \mathbf{b}, \mathscr{A}) &= W^{23}_{11}\mathscr{A}_1(W^{12}_{11}\xi + b)) \\ N_2(\xi; \mathbf{W}, \mathbf{b}, \mathscr{A}) &= W^{23}_{22}\mathscr{A}_1(W^{12}_{12}\xi + b)) \end{aligned} \tag{10}$$

These representations can be combined with additional layers of NN to provide interpolations of the displacement. For details we refer to [28].

Figure 5c shows the three building blocks for evaluating the 1D shape function derivatives. To explain the function of each building block, we recall that the 1D version of Eq.(6) is $\mathscr{B}^0_I = \frac{\partial N_I}{\partial X} = \frac{\partial N_I}{\partial \xi}\left(\frac{\partial X}{\partial \xi}\right)^{-1}$. For the 2-node element, we introduce

$$\mathscr{B} = \begin{bmatrix} \mathscr{B}^0_1 & \mathscr{B}^0_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial N_1}{\partial X} & \frac{\partial N_2}{\partial X} \end{bmatrix} = \left(\frac{\partial X}{\partial \xi}\right)^{-1} \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi} \end{bmatrix} = \mathbf{J}^{-1}\mathbf{D}_N \tag{11}$$

where $\mathbf{J} = \frac{\partial X}{\partial \xi}$ is the Jacobian for the coordinate transformation, $\mathrm{D}_N = \begin{bmatrix} \frac{\partial N_1}{\partial X} & \frac{\partial N_2}{\partial X} \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} \end{bmatrix}$. The Jacobian is evaluated through

$$\mathbf{J} = \frac{\partial X}{\partial \xi} = \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi} \end{bmatrix} \begin{Bmatrix} X_1 \\ X_2 \end{Bmatrix} = \mathbf{D}_N \mathbf{X}^* \tag{12}$$

where $\mathbf{X}^* = \{X_1 \ X_2\}^T$ provides the physical coordinates of the nodes.

Based on Eqs. (11) and (12), the first building block in Figure 5c employs a single layer of NN to evaluate $\mathbf{D}_N$. Since $\mathbf{D}_N$ is a constant matrix for the 1D linear element, the input to NN is a constant as well. It is called a partial derivative operator block because the block performs differentiation (in general partial differentiation) of the shape functions with respect to the element coordinates. Subsequently, another two layers are added to construct the second building block to compute **J**. This block is called an r-adaptivity block since the weights connecting these two layers are the physical coordinates of the nodes and can be trained by optimizing the loss function. This is generally regarded as a learning process through which NN arrives at the optimum nodal positions. As will be shown later, we establish physics-based loss functions based on nonlinear FEM. The last block is called a materials derivative block that evaluates $\mathscr{B}$ based on Eq.(11). This block takes the computed **J** from the r-adaptivity block and inverts it using the inversion building block V as established in [28]. Another layer is then added to compute $\mathscr{B} = \mathbf{J}^{-1}\mathbf{D}_N$. The computed shape function derivatives can be employed to evaluate the deformation gradient given as

$$\begin{aligned} F(\xi; \mathbf{W}, \mathbf{b}, \mathscr{A}) = u_1 \mathscr{A}_1(W_{11}^{56} \\ \cdot (W_{11}^{45}(\mathscr{A}_1(W_{11}^{34}\mathscr{A}_1(W_{11}^{23}\mathscr{A}_1(W_{11}^{12}))) + \mathscr{A}_1(W_{21}^{34}\mathscr{A}_1(W_{22}^{23}\mathscr{A}_1(W_{12}^{12})))))^{-1}) + u_2 \mathscr{A}_1(W_{11}^{56} \\ \cdot (W_{11}^{45}(\mathscr{A}_1(W_{11}^{34}\mathscr{A}_1(W_{11}^{23}\mathscr{A}_1(W_{11}^{12}))) + \mathscr{A}_1(W_{21}^{34}\mathscr{A}_1(W_{22}^{23}\mathscr{A}_1(W_{12}^{12})))))^{-1}) + \mathbf{I} \end{aligned} \tag{13}$$

Figure 6 illustrates the construction of the shape functions and their derivatives for the case of 1D quadratic element. Configurations of the element in the physical and parent domain are shown in Figure 6a. The shape functions at nodes 1, 2 and 3 are derived from the well-known 2nd order Lagrange polynomials, given as $N_1 = \frac{1}{2}\xi(1-\xi)$, $N_2 = 1-\xi^2$ and $N_3 = \frac{1}{2}\xi(1+\xi)$. Two internal layers are introduced in building the shape function using the given element coordinate as an input. The multiplication building block is introduced in the second layer to realize the product form of the shape functions. The DNN representations of the shape functions are given as

$$\begin{aligned} N_1(\xi; \mathbf{W}, \mathbf{b}, \mathscr{A}) &= W_{11}^{34}\left\{ M\left[(W^{23}\mathscr{A}_1(W_{11}^{12}\xi + b_1^2)), (W^{23}\mathscr{A}_1(W_{12}^{12}\xi + b_2^2))\right]\right\} \\ N_2(\xi; \mathbf{W}, \mathbf{b}, \mathscr{A}) &= W_{22}^{34}\left\{ M\left[(W^{23}\mathscr{A}_1(W_{13}^{12}\xi + b_3^2)), (W^{23}\mathscr{A}_1(W_{14}^{12}\xi + b_4^2))\right]\right\} \\ N_3(\xi; \mathbf{W}, \mathbf{b}, \mathscr{A}) &= W_{33}^{34}\left\{ M\left[(W^{23}\mathscr{A}_1(W_{12}^{12}\xi + b_2^2)), (W^{23}\mathscr{A}_1(W_{14}^{12}\xi + b_4^2))\right]\right\} \end{aligned} \tag{14}$$

with the specific weights and biases shown in Figure 6b.

The shape function derivatives can be constructed using the same three building blocks as in the case of 1D linear element and detailed NN structures are shown in Figure 6c. Unlike the linear element case, neither the Jacobian nor the shape function derivative is constant. To evaluate the shape function derivative, the element coordinate is used as the input. Based on

Figure 6c, the shape function derivatives are evaluated according to $B = \mathbf{J}^{-1}\mathbf{D}_N$ in which the building blocks for evaluating $\mathbf{D}_N$, $\mathbf{J}$ and B are shown. The material coordinates are used as weights in constructing $\mathbf{J}$ and can be optimized through training. The deformation gradient is given as

$$\begin{aligned} F(\xi;\mathbf{W},\mathbf{b},\mathscr{A}) &= u_1 M\left[W_{11}^{67}\mathbf{J}^{-1}, W_{11}^{37}\mathscr{A}_1(W_{11}^{23}\mathscr{A}_1(W_{11}^{12}\xi + b_1^2))\right] \\ &+ u_2 M\left[W_{12}^{67}\mathbf{J}^{-1}, W_{22}^{37}\mathscr{A}_1(W_{22}^{23}\mathscr{A}_1(W_{12}^{12}\xi + b_2^2))\right] \\ &+ u_3 M\left[W_{13}^{67}\mathbf{J}^{-1}, W_{33}^{37}\mathscr{A}_1(W_{33}^{23}\mathscr{A}_1(W_{13}^{12}\xi + b_3^2))\right] \end{aligned} \tag{15}$$

with

$$\mathbf{J}^{-1} = W_{11}^{56}\mathscr{A}_1\begin{Bmatrix} W_{11}^{45}\mathscr{A}_1(W_{11}^{34}\mathscr{A}_1(W_{11}^{23}\mathscr{A}_1(W_{11}^{12}\xi + b_1^2))) + \\ W_{21}^{45}\mathscr{A}_1(W_{22}^{34}\mathscr{A}_1(W_{22}^{23}\mathscr{A}_1(W_{12}^{12}\xi + b_2^2))) + \\ W_{31}^{45}\mathscr{A}_1(W_{33}^{34}\mathscr{A}_1(W_{33}^{23}\mathscr{A}_1(W_{13}^{12}\xi + b_3^2))) \end{Bmatrix} \tag{16}$$

3.2.2 DNN Construction of the shape functions and derivatives in 2D elements—

We consider the 2D 4-node quadrilateral element. Configurations of the parent and material configurations are shown in Figure 7.

The shape functions are given as

$$\begin{aligned} &\mathbf{N} = [N_1 \;\; N_2 \;\; N_3 \;\; N_4] \\ &N_1 = \frac{1}{4}(1-\xi)(1-\eta), \quad N_2 = \frac{1}{4}(1+\xi)(1-\eta) \\ &N_3 = \frac{1}{4}(1+\xi)(1+\eta), \quad N_4 = \frac{1}{4}(1-\xi)(1+\eta) \end{aligned} \tag{17}$$

Figure 8 shows the construction of the shape functions using DNN and the corresponding representations are expressed as

$$\begin{aligned} N_1(\xi,\eta;\mathbf{W},\mathbf{b},\mathscr{A}) &= W_{11}^{34}\left\{M\left[(W^{23}\mathscr{A}_1(W_{12}^{12}\xi + b_2^2)), (W^{23}\mathscr{A}_1(W_{24}^{12}\xi + b_4^2))\right]\right\} \\ N_2(\xi,\eta;\mathbf{W},\mathbf{b},\mathscr{A}) &= W_{22}^{34}\left\{M\left[(W^{23}\mathscr{A}_1(W_{11}^{12}\xi + b_1^2)), (W^{23}\mathscr{A}_1(W_{24}^{12}\xi + b_4^2))\right]\right\} \\ N_3(\xi,\eta;\mathbf{W},\mathbf{b},\mathscr{A}) &= W_{33}^{34}\left\{M\left[(W^{23}\mathscr{A}_1(W_{11}^{12}\xi + b_1^2)), (W^{23}\mathscr{A}_1(W_{23}^{12}\xi + b_3^2))\right]\right\} \\ N_4(\xi,\eta;\mathbf{W},\mathbf{b},\mathscr{A}) &= W_{44}^{34}\left\{M\left[(W^{23}\mathscr{A}_1(W_{12}^{12}\xi + b_2^2)), (W^{23}\mathscr{A}_1(W_{23}^{12}\xi + b_3^2))\right]\right\} \end{aligned} \tag{18}$$

To establish the DNN for the shape function derivative, we define

$$\mathbf{D}_N = \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi} & \frac{\partial N_3}{\partial \xi} & \frac{\partial N_4}{\partial \xi} \\ \frac{\partial N_1}{\partial \eta} & \frac{\partial N_2}{\partial \eta} & \frac{\partial N_3}{\partial \eta} & \frac{\partial N_4}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \mathbf{D}_N^1 \\ \mathbf{D}_N^2 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} -(1-\eta) & (1-\eta) & (1+\eta) & -(1+\eta) \\ -(1-\xi) & -(1+\xi) & (1+\xi) & (1-\xi) \end{bmatrix} \tag{19}$$

$$\mathbf{X}^* = \begin{bmatrix} X_1 & Y_1 \\ X_2 & Y_2 \\ X_3 & Y_3 \\ X_4 & Y_4 \end{bmatrix} \tag{20}$$

$$\mathbf{J} = \mathbf{D}_N \cdot \mathbf{X}^* = \begin{bmatrix} \mathbf{D}_N^1 \cdot \mathbf{X}^* \\ \mathbf{D}_N^2 \cdot \mathbf{X}^* \end{bmatrix} \tag{21}$$

Figure 9a provides the building blocks for obtaining $\mathbf{D}_N$, $\mathbf{J}$ based on Eqs. (19)) to (21) and eventually shape function derivative matrix B_0 that is used to compute the deformation gradient in 2D. Note that the HiDeNN multiplication and inversion blocks have been used extensively. For instance, computing $\mathbf{J}^{-1}$ requires evaluating the determinant $\det(\mathbf{J})$ and the detailed DNN is shown in Figure 9b with the application of the multiplication block M. To further evaluate $\mathbf{J}^{-1}$ the matrix inversion block V is applied as shown in Figure 9c.

4 Training of DNN in HiDeNN-FEM and Solution Method

With the shape functions established in section 3 using DNN, the general expression for the displacement is given as

$$\mathbf{u}^h = \mathbf{N}(\boldsymbol{\xi}; \mathbf{W}, \mathbf{b}, \mathscr{A})\mathbf{d} \tag{22}$$

in which $\mathbf{N}$ is the shape function matrix and $\mathbf{d}$ is the nodal displacement vector. Unlike the conventional FEM approximation, this DNN-constructed approximation contains additional NN parameters such as the weights $\mathbf{W}$, the biases $\mathbf{b}$ and the activation functions $\mathscr{A}$. In particular, the weights for the r-adaptivity block are functions of the material coordinates of the nodes. Thus, we rewrite Eq.(22) as

$$\mathbf{u}^h = \mathbf{N}(\boldsymbol{\xi}; \mathbf{X}^*, \mathbf{b}, \mathscr{A})\mathbf{d} \tag{23}$$

in which $\mathbf{X}^*$ is the vector that contains all the material coordinates of the nodes as part of the weight in the DNN. For the given choices of the activation functions $\mathscr{A}$, $\mathbf{X}^*$ can be trained to achieve optimum accuracy. This is equivalent to the r-adaptivity in FEM that is realized through a learning process in DNN in HiDeNN-FEM.

The learning process in the general field of ML can be categorized into supervised and unsupervised learning, depending on whether any data that correlates the input (feature) to the output (label) is used for the training process. In this work, an unsupervised learning approach is established in which the NN parameters are optimized by defining a loss function without using training data. For nonlinear FEM the loss function is formulated based on the residual while using Newton's method. Here we focus on static problems so that the inertia terms in Eq.(2) are neglected. In Newton's method, Eq.(2) is linearized and solved through iteration. The iterative step starts with the residual vector $\mathbf{r}$ for the υ-th iteration that is expressed as

$$\mathbf{r}_{\upsilon} = \mathbf{r}\left(\mathbf{d}_{\upsilon}, \mathbf{X}_{\upsilon}^{*}\right) = \mathbf{f}_{\upsilon}^{\text{int}} - \mathbf{f}_{\upsilon}^{ext} \tag{24}$$

Setting the conditions that $\mathbf{r}_{\upsilon+1} = \mathbf{0}$ and performing a first-order Taylor expansion of the values at the υ-th iteration gives

$$\mathbf{r}_{\upsilon+1} = \mathbf{r}_{\upsilon} + \frac{\partial \mathbf{r}\left(\mathbf{d}_{\upsilon}, \mathbf{X}_{\upsilon}^{*}\right)}{\partial \mathbf{d}} \Delta\mathbf{d} + \frac{\partial \mathbf{r}\left(\mathbf{d}_{\upsilon}, \mathbf{X}_{\upsilon}^{*}\right)}{\partial \mathbf{X}^{*}} \Delta\mathbf{X} = \mathbf{0} \tag{25}$$

We define

$$\mathbf{A} = \frac{\partial \mathbf{r}}{\partial \mathbf{d}} \quad \text{and} \quad \mathbf{A}^{*} = \frac{\partial \mathbf{r}}{\partial \mathbf{X}^{*}} \tag{26}$$

which represents the tangent stiffness associated with the system itself and the mesh, respectively. Correspondingly, $\mathbf{A}$ and $\mathbf{A}^{*}$ are called the system Jacobian matrix and mesh Jacobian matrix, respectively. It can be shown that the matrix components of $\mathbf{A}$ are given as

$$\begin{aligned} \mathbf{A}_{IJ} &= \frac{\partial \mathbf{r}_I}{\partial \mathbf{d}_J} = \frac{\partial \mathbf{f}_I^{mat}}{\partial \mathbf{d}_J} + \frac{\partial \mathbf{f}_I^{geo}}{\partial \mathbf{d}_J} - \frac{\partial \mathbf{f}_I^{ext}}{\partial \mathbf{d}_J} \\ &= \int_{\Omega_0} \mathbf{B}_{0I}^{T} [\mathbf{C}^{SE}] \mathbf{B}_{0J} d\Omega_0 + \mathbf{I} \int_{\Omega_0} \mathscr{B}_{0I}^{T} \mathbf{S} \mathscr{B}_{0J} d\Omega_0 - \frac{\partial \mathbf{f}_I^{ext}}{\partial \mathbf{d}_J} \end{aligned} \tag{27}$$

in which $\mathbf{f}_I^{mat}$ and $\mathbf{f}_I^{geo}$ represents the contribution to the internal nodal force due to the material and geometric nonlinearity, respectively. $\frac{\partial \mathbf{f}_I^{ext}}{\partial \mathbf{d}_J}$ is denoted as the external load stiffness due to loads that are changing with the configuration of the body. This term is neglected in the current work. $\mathbf{B}_{0I}$ is the Voigt form of the $\mathbf{B}_0$ matrix for the shape function defined at node I. The detailed step to form $\mathbf{B}_{0I}$ can be found in ref [33]. $\mathbf{C}^{SE}$ is the material tangent stiffness tensor that relates the 2nd Piola-Kirchhoff stress $\mathbf{S}$ to Green-Lagrangian strain $\mathbf{E}$ through $\dot{\mathbf{S}} = \mathbf{C}^{SE} : \dot{\mathbf{E}}$. $\mathbf{I}$ is the 2nd order identity tensor.

The mesh Jacobian matrix is given as

$$\mathbf{A}^{*} = \frac{\partial \mathbf{r}}{\partial \mathbf{X}^{*}} = \frac{\partial \mathbf{f}^{\text{int}} - \partial \mathbf{f}^{ext}}{\partial \mathbf{X}^{*}} \tag{28}$$

We assume that $\mathbf{f}^{ext}$ is independent of the nodal coordinates and thus

$$\mathbf{A}^{*} = \frac{\partial \mathbf{f}^{\text{int}}}{\partial \mathbf{X}^{*}} = \int_{\Omega_0} \left(\frac{\partial \mathscr{B}^{0}}{\partial \mathbf{X}^{*}} \right)^{T} \mathbf{P} d\Omega_0 \tag{29}$$

With the system and mesh Jacobian matrix evaluated based on Eqs.(27) and (29), the nodal displacement and coordinates are updated by solving Eq.(25). Since this equation can yield

multiple sets of solutions due to the additional unknowns of the nodal coordinate, the computational implementation seeks to improve the accuracy without sacrificing efficiency by establishing a two-step process: In the first step, the nodal displacements are updated from solving $\Delta\mathbf{d} = \mathbf{A}^{-1}\mathbf{r}$ while the nodal coordinates are being held fixed. In the second step, nodal coordinates are updated by solving $\Delta\mathbf{X} = (\mathbf{A}^{*})^{-1}\mathbf{r}$ and neglecting the contribution from the system Jacobian matrix. The Lagrangian mesh is then updated. This process will continue iteratively until the loss function (residual error) $\mathbf{r}$ is within the given tolerance. Figure 10 provides the detailed algorithm (Algorithm 1) for this implementation.

For conservative systems, setting the residual in Eq.(25) to be zero is equivalent to the application of the stationary potential energy principle. Based on this, we can introduce the potential energy Π and solve the minimization problem given below

$$\text{Find } \mathbf{d}, \mathbf{X}^{*}, s.t. \quad \Pi(\mathbf{d}, \mathbf{X}^{*}) = W^{\text{int}} - W^{\mathbf{ext}} \text{ is minimized} \tag{30}$$

Here W^{int} and $W^{\mathbf{ext}}$ are the internal (strain) energy and external work respectively. The specific expression for W^{int} depends on the material model used, as long as it is conservative. For instance, we have used hyperelastic material in which the strain energy density function w is introduced. The internal energy is then given as $W^{\text{int}} = \int_{\Omega_0} w d\Omega_0$. The specific form of w has been provided in the example problems in section 5. W^{ext} can be derived based on Eq.(4) assuming the external forcing terms are conservative. Using indicial notation, we have

$$W^{ext} = f_{iI}^{ext} u_{iI} = u_{iI}\left(\int_{\Omega_0} N_I \rho_0 b_i d\Omega_0 + \int_{\Gamma_{t_i}^0} N_I \bar{t}_i^0 d\Gamma_0\right) \tag{31}$$

Based on Eq. (30), one can define the loss function to be the potential energy of the system and systematically minimize it to solve for equilibrium and realize the mesh update.

Figure 11 provides the detailed algorithm (Algorithm 2) used for solving the conservative system using DNN. A two-step nested loops optimization scheme is developed. The outer loop resolves the nodal displacement while keeping mesh fixed and the inner loop then further provides the mesh update. In both loops, the variables (the weight of DNN) are updated using the gradient descent approach with associated learning rate parameters α and γ as shown in Figure 11.

5 Nonlinear HiDeNN-FEM Numerical Examples

In this section, we provide several numerical examples to demonstrate the performance of nonlinear HiDeNN-FEM and compare it with regular FEM. An in-house code has been developed for this purpose. It should be noted that the degrees of freedom (DoFs) referred in the following example problems are the nodal DoFs only. In HiDeNN-FEM, there are

additional DoFs associated with the nodal coordinates that are also being optimized through r-adaptivity. Therefore, the total DoFs in HiDeNN-FEM will be twice of that in regular FEM if all the nodal coordinates are being optimized. In addition, the commercial code ABAQUS has been employed to obtain reference solutions for validation. As described in section 4, we have established two classes of algorithms, one for the general nonlinear problems (Algorithm 1) and one for the conservative systems (Algorithm 2). In the first 4 examples to be described below, we focus on conservative systems using Algorithm 2 and the last 2 examples dealing with the general nonlinear plasticity problems are solved using Algorithm 1. All computations were performed on a workstation with Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz and 32 GB memory.

5.1 Plane stress problem of a hyperelastic plate with a hole under tension

We consider a plane stress problem of a square plate of dimension 1 m by 1 m with the center hole of radius of 0.1m. As shown in Figure 12, the plate is modeled as Neo-Hookean material with the strain energy density function w given as

$$w=C_{10}(\bar{I}_1-3)+\frac{1}{D_1}(J\text{-}1)^2 \tag{32}$$

with $\bar{I}_1=J^{-2/3}I_1$, $J=\det(\mathbf{F})$ and I_1 is the first invariant of the right Cauchy-Green deformation tensor. The material parameters are given as $C_{10}=115.385$ kPa, $D_1=4\times10^{-6}\text{Pa}^{-1}$, and the plate is subjected to a traction of 100 kPa on the top surface while the bottom surface is constrained.

Figure 13 and Table 1 show the result for the given traction and boundary condition. The converged result in Figure 13 (dashed line) is obtained from a very fine mesh of ~10M degrees of freedom (DoFs) using the CPS4 (4-node plane stress) element in ABAQUS. HiDeNN-FEM achieves much faster convergence than the standard FEM using ABAQUS. For the same FE model using the standard 4-node quadrilateral element with 13,840 degrees of freedom, the error from HiDeNN-FEM is 0.36% in terms of the maximum Mises stress whereas the corresponding is 3.87% from ABAQUS.

5.2 Compression of a rectangular plate with 2 different aspect ratios

We consider a plane strain case of a rectangular plate with the dimension of 0.02 m by 0.01 m as shown in Figure 14. The plate is modeled as nearly incompressible Neo-Hookean material using the same form of strain energy density function as in the previous cases. The material parameters are given as $C_{10}=100.341$ kPa, $D_1=1.02\times10^{-7}\text{Pa}^{-1}$. These constants provide an equivalent Poisson's ratio of 0.4949. In terms of the loading condition, displacement is applied on the top surface while the bottom surface is constrained.

This case aims to test the ability of HiDeNN-FEM method in capturing large deformation and alleviating the issue of mesh distortion with r-adaptivity. The initial domain was discretized using a standard 4-node quadrilateral element with different mesh densities (40×10, 80×20, 160×40). For the same mesh, both the standard FEM using ABAQUS and HiDeNN-FEM with r-adaptivity were employed to capture the compression response.

Reduced integration with hourglass control was used to avoid mesh locking as well as spurious modes. Figure 15 and Table 2 show the maximum percentage of compression that has been reached for each case. It is observed that the maximum compression ratio reached in the case of HiDeNN-FEM with r-adaptivity is consistently higher than the corresponding FEM with the same initial mesh by ~66% to 75%. Figure 16 shows undeformed and deformed mesh obtained from regular FEM and HiDeNN-FEM with r-adaptivity for the case of mesh with total of 13,202 DoFs. The maximum compression ratio was 33.11% for the regular FEM case and 55.08% for the HiDeNN-FEM. Mesh updates were observed at the four corners in the HiDeNN-FEM case and are responsible for the higher compression ratio achieved using HiDeNN-FEM.

In the next example, as shown in Figure 17, the plate dimensions were changed to a width of 0.1 m and height of 0.01 m, leading to an aspect ratio of 10:1. The plate shown is discretized using the same 4-node quadrilateral element with 160×40 elements. The material model remains the same as in the last example. Figure 18 shows the initial and deformed mesh before the simulation terminates due to mesh distortion for the cases of regular FEM and HiDeNN-FEM. It is observed that the maximum compression ratio nearly doubles from FEM (20.6%) to HiDeNN-FEM (40.3%). A comparison between the HiDeNN-FEM mesh and the normal stress in the Y direction shows a correlation between the area of high stress gradient and the area where mesh has been updated to alleviate the mesh distortion in these areas and capture the high stress gradient.

5.3 Compression for a 3D cubic block

We consider a 3D cubic block with dimensions of 1 m × 1 m × 1 m as shown in Figure 20. The block is modeled as hyperelastic Neo-Hookean material with the same set of material parameters as in Section 5.1. In terms of the boundary condition, the block is subjected to a 0.1 m displacement on the top surface while the bottom surface is constrained.

For this example, we have run 4 different cases depending on whether regular FEM (using ABAQUS) or HiDeNN-FEM with r-adaptivity is used and whether hourglass control is implemented, as indicated in Table 3 below.

All four cases are discretized using 8 nodes hexahedron element. To assess the accuracy of the results, a reference solution is obtained by prescribing a very fine mesh with 10,155,231 degrees of freedom using the C3D8R element (3,310,008 elements) with hourglass control. Figure 21 compares the maximum transversal displacement obtained from cases a and b in Table 3. Each case was run with three different mesh densities with 768, 8670 and 42471 DoFs. It can be observed that the maximum displacement solved from regular FEM does not converge to the reference solution. On the other hand, results from HiDeNN-FEM with r-adaptivity converge to the reference solution as the mesh is refined, indicating that the hourglass mode effect is suppressed. The specific values of the maximum displacements obtained from the mesh configuration and the reference solution are provided in Table 4.

To further verify whether hourglass mode is present, Figure 22 shows the deformed mesh from the case a and b in Table 3 for a total DoF of 42,471. The hourglass mode is visible from the regular FEM results, whereas HiDeNN-FEM does not exhibit any. Figure 23

provides the cross-sectional view from the two cases where the cross-sections are located at 0.3m, 0.5m, and 0.8m from the bottom surface. Again, the hourglass mode is seen in the regular FEM case. Finally, Figure 24 compares von Mises stress distribution along the left edge of the top surface of the cubic block. Hourglass mode leads to stress oscillations in the cases of regular FEM, whereas HiDeNN-FEM converges to the reference solution.

Since no hourglass mode is present for the applied displacement of 0.1m (compression ratio of 10%), we continue to apply compression and observe the onset of hourglass mode at a compression ratio of 30.6% for HiDeNN-FEM (case b) with total DoFs of 42,471, whereas regular FEM (case a) exhibited hourglass mode at a compression ratio of 12% with the same mesh. These results indicate the proposed HiDeNN-FEM can significantly suppress the hourglass mode effect in the case of large deformation.

In cases c and d in Table 3, hourglass control was implemented in both regular FEM and HiDeNN-FEM. As described in section 4, a 2-step optimization scheme is introduced in HiDeNN-FEM. To assess the computational efficiency of this implementation, the time consumption for nodal position optimization was monitored and shown in Table 5 for the different iterative steps. Table 5 shows that the r-adaptivity realized through the nodal position update loop adds a moderately less than 5% overhead to the overall computational time. The detailed mesh updates are shown in Figure 25 with the cross-sectional view of the mesh at the 1st, 3rd, and 5th iterative steps.

Figure 26 compares the deformed shape between regular FEM mesh and HiDeNN-FEM mesh and the cross-section view. Results show that even with hourglass control, spurious mode emerges in regular FEM at 24.50% compression ratio and the computation stopped at 31.62% compression ratio due to mesh distortion. On the other hand, the HiDeNN-FEM mesh result does not show apparent hourglass mode at the same compression ratio (24.50%) and can be further compressed to 35.97% before the simulation terminates.

Figure 27 compares the Z direction stress distribution at the line along the Y direction (X=1, Z=1) between cases c and d in Table 3 and with the reference solution at 24.50% compression ratio. Since the model with C3D8R element in ABAQUS can not reach such a high compression ratio, a different model using 12,288 C3D20RH elements (20 node hybrid quadratic element with reduced integration) was run to obtain the reference solution. These results show that HiDeNN-FEM can better capture stress distribution and matches well with the reference solution.

5.4 3D analysis of spot weld in a hat-stiffened panel

In this case, we consider a hat-stiffened panel with one stiffener. Spot weld was applied to connect the panel skin with the stiffener as shown in Figure 28. A quarter of the part was modeled due to symmetry. The dimension and boundary conditions are also shown in Figure 28. The spot welds are 0.07 inches in radius and 0.0005 inches thick. The thickness values of the panel and the hat stiffener are 0.0625 inches and 0.032 inches respectively. We assume the same Neo-Hookean material model with material constants of $C_{10} = 9.6154 \times 10^8$ psi and $D_1 = 4.8 \times 10^{-10}\text{psi}^{-1}$. For this problem, a refined mesh is needed in regular FEM for

modeling the stress accurately in the local spot weld area due to the small dimension, while HiDeNN-FEM is shown to provide improved accuracy with the same initial mesh density.

The model is meshed by an 8-node hexahedron element with four mesh densities that contain 5177, 8791, 10843, 18919 elements, respectively. The reference solution is obtained by solving the same problem with C3D8R with hourglass control in ABAQUS. A total of 2,871,396 elements and 10,044,327 degrees of freedom were used. The results are assessed in terms of the stress concentration as measured by the maximum effective (von Mises) stress as shown in Table 6 and Figure 29. It is observed that for the case of 18,919 elements HiDeNN-FEM yields Max Mises stress value that is within 0.29% error when compared with the reference solution, whereas the error was 2.53% for regular FEM. This accuracy improvement only adds 7.3% overhead in computing time to implement r-adaptivity. The resolved effective stress distribution from HiDeNN-FEM is shown in Figure 30.

5.5 A 2D elastoplastic block subjected to nonuniform body force

We consider a 2D rectangular block with dimensions of 0.2 m by 2 m as shown in Figure 30. The block is subjected to nonuniform body force while being fixed on the left end. The body force distribution is given as $b_x = b_0\left[-\frac{4\pi^2(10(X-0.25))^2 - 2\pi}{e^{\pi(10(X-0.25))^2}}\right]$ and $b_y = b_0\sin(10Y)$ with $b_0 = 65$ MPa/m. An elastoplastic constitutive model is introduced for the block. Before it reaches plasticity, the material is assumed to be linear elastic with Young's modulus of $E = 200$GPa and Poisson's ratio of 0.3. To describe the plastic response, we introduce the standard von Mises yield surface with linear hardening, given as $f = \bar{\sigma} - \sigma_Y$ in which $\bar{\sigma}$ is the effective stress. Furthermore, $\sigma_Y = \sigma_Y^0 + H\bar{\varepsilon}_p$ with $\sigma_Y^0 = 100$ MPa as the initial yield stress, $H = 100$ MPa is the hardening modulus, $\bar{\varepsilon}_p$ is the effective plastic strain. A standard radial return method is employed to resolve the stress and strain when material enters plasticity.

Since the system is no longer conservative, we have applied Algorithm 1 in HiDeNN-FEM implementation. A reference solution is obtained by solving the same problem in ABAQUS using a refined mesh of 5,017,600 CPS4 (4 node plane stress) elements with ~ 10 million DoFs. Table 7 compares the solution and time consumption for different implementations. Four different meshes were tested in comparing regular FEM with HiDeNN-FEM. As can be seen, regular FEM fails to converge in the case of 100 elements, whereas HiDeNN-FEM converges and yields a maximum von Mises stress within 0.2% of the reference solution. Regular FEM does not yield comparable accuracy until it uses 6,400 elements while HiDeNN-FEM consistently maintains high accuracy as the mesh is refined. In terms of time consumption, it takes HiDeNN 8.86 seconds to arrive at a prediction within 0.1% of the reference solution, whereas the same for regular FEM was 107.92 seconds. The last column in Table 7 shows the overhead time used for performing r-adaptivity and this operation consumes ~10% of the overall computing time. Figure 32 shows the effective stress contour along with the mesh from both the regular FEM and HiDeNN-FEM.

5.6 An elastoplastic plate with a hole subjected to tension

In this example, we consider a 2D plane strain problem of a square plate of dimension 1m by 1m with a hole of radius of 0.1 m located in the center as shown in Figure 33. The

left side of the plate is fixed and the right side is subjected to uniform traction of 37 MPa. The same elastoplasticity model as described in section 5.5 is employed for modeling the plate with the same material constants. Reference solutions are obtained using ABAQUS by discretizing the problem domain with 5,054,756 CPE4 (4-node plane strain) elements and total DoFs of 10,120,464. For comparison between regular FEM and HiDeNN-FEM, the domain is discretized with 4-node quadrilateral elements with two different mesh densities. The number of elements for the two meshes is respectively 436, and 1700 with the corresponding DoFs of 972 and 3600 respectively. For each mesh, simulations using regular FEM and HiDeNN-FEM are performed to compare the predictions as well as the execution time. Similar to the last case, Algorithm 1 was implemented in HiDeNN-FEM to perform r-adaptivity.

Table 8 provides the computed maximum von Mises stress from regular FEM and HiDeNN-FEM for the two meshes and the difference when compared with the reference solution. It is observed that HiDeNN-FEM yields high accuracy even with a relatively coarse mesh, whereas for the same mesh density the regular FEM prediction differs from the reference solution by more than 10%. The total computing time from HiDeNN-FEM is relatively higher due to the time it takes to perform the r-adaptivity. When the mesh is further refined to a total number of elements of 1700, both regular FEM and HiDeNN-FEM converge while HiDeNN-FEM demonstrates better accuracy with r-adaptivity. The computing time of HiDeNN-FEM is also higher than the regular FEM by ~ 17%, much of which is due to the r-adaptivity as shown in the last column. As can be seen from Figure 34, the nodes in the regions of high stress concentration are moved based on Algorithm 1 and are responsible for the accurate prediction without the use of a large number of elements as in regular FEM. A separate simulation (not shown here) using regular FEM shows that ~ 4 times the elements (6400) are needed to accomplish the same order of accuracy in the prediction when compared to the case of HiDeNN-FEM with 1700 elements.

6 Conclusion

In summary, we have presented a general framework of hierarchical deep-learning Neural Network for nonlinear finite element (nonlinear HiDeNN-FEM) by building on the prior work by Zhang et al [28] on linear HiDeNN-FEM and basic building blocks. In nonlinear HiDeNN-FEM, the shape function approximations and material derivatives are constructed through three new basic building blocks: The first building block differentiates the shape functions with respect to the element coordinates. The second building block evaluates the Jacobian of the coordinate transformation and its inverse. It also incorporates the material coordinates as the weights of the DNN, thus enabling r-adaptivity through training. The third building block evaluates the material derivatives of the shape functions, which can then be used to form the shape function derivative matrix as commonly used in nonlinear FE. Since the building blocks are described independently of the specific element formulation, it can be generally applied to any 2D and 3D elements.

Aside from the HiDeNN-FEM approximation, implementations of the nonlinear solution scheme based on Newton’s methods are also presented. A general linearization approach was adopted and it is shown that this leads to an iterative scheme that involves the

optimization of the nodal solutions as well as the nodal coordinates, i.e., r-adaptivity. For practical implementation and ease of integration with existing FE codes, a 2-step iterative scheme is proposed to improve computational efficiency, and this solution scheme is termed Algorithm 1. For conservative systems, the problem can be generalized into minimizing a loss function that represents the potential of the system. A solution scheme featuring the 2-step nested loop is proposed for these types of problems and is termed Algorithm 2. Both Algorithms 1 and 2 have been implemented on multiple problems involving geometric and material nonlinearities. These benchmark problems demonstrate that nonlinear HiDeNN-FEM achieves much better accuracy than regular FEM without adding significant overhead to the computational cost. In addition, it is also shown that r-adaptivity can effectively reduce element distortion and suppress the hourglass mode, which are some of the main issues faced in the application of nonlinear FEM.

The work presented has focused on nonlinear static equilibrium problems and will be extended to incorporate inertia effects and time-dependent properties by introducing a space-time framework [34–36]. It is worth noting that the integration with DNN offers a very interesting perspective on nonlinear HiDeNN-FEM, as both advanced algorithms and computing platforms are being developed for machine learning applications. These integrations are expected to significantly enhance the predictive capabilities of nonlinear HiDeNN-FEM and will be the focus of future research.

Acknowledgements

DQ and YJL would like to acknowledge the support from NIH under the Grant NIH/NIBIB R01 EB025247 (through a subcontract from the UT Southwestern Medical Center). WKL, SM, and YL would like to acknowledge the support of National Science Foundation (NSF, USA) grants CMMI-1762035 and CMMI-1934367. C. Park would like to thank the Division of Orthopedic Surgery and Sports Medicine at Ann and Robert H. Lurie Children's Hospital for their philanthropic grant.

References

1. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Cambridge
2. Zhao Z, Zheng P, Xu S, et al. (2019) Object detection with deep learning: A review. IEEE transactions on neural networks and learning systems, 30(11), 3212–3232 [PubMed: 30703038]
3. Krizhevsky A, Sutskever I, Hinton G (2017) ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84–90
4. He K, Zhang X, Ren S, Sun J (2016) Deep Residual Learning for Image Recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770–778)
5. Sermanet P, Eigen D, Zhang X, et al. (2013) OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. arXiv preprint arXiv:1312.6229
6. Otter D, Medina J, Kalita J, (2020) A survey of the usages of deep learning for natural language processing. IEEE transactions on neural networks and learning systems, 32(2), 604–624
7. Walczak S (2005) Artificial neural network medical decision support tool: Predicting transfusion requirements of ER patients. IEEE Transactions on Information Technology in Biomedicine, 9(3), 468–474 [PubMed: 16167701]
8. Tarca AL, Carey VJ, Chen XW, et al. (2007) Machine learning and its applications to biology. Machine learning and its applications to biology. PLoS computational biology, 3(6), e116 [PubMed: 17604446]
9. García-Cano E, Cosío F, Duong L, et al. (2018) Prediction of spinal curve progression in adolescent idiopathic scoliosis using random forest regression. Computers in biology and medicine, 103, 34–43 [PubMed: 30336363]

10. Halabi S, Prevedello L, et al. (2019) The RSNA pediatric bone age machine learning challenge. Radiology, 290(2), 498 [PubMed: 30480490]
11. Silver D, Huang A, Maddison C, et al. (2016) Mastering the game of Go with deep neural networks and tree search. nature, 529(7587), 484–489 [PubMed: 26819042]
12. Liu Y, Zhao T, Ju W, et al. (2017) Materials discovery and design using machine learning. Journal of Materiomics, 3(3), 159–177
13. Gomes C, Selman B, et al. (2019) Artificial intelligence for materials discovery. MRS Bulletin, 44(7), 538–544
14. Weinan E, Bing Y (2018) The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems. Communications in Mathematics and Statistics 6(1):1–12
15. Lagaris I, Likas A, Fotiadis D (1998) Artificial neural networks for solving ordinary and partial differential equations. IEEE transactions on neural networks, 9(5), 987–1000 [PubMed: 18255782]
16. Asady B, Hakimzadegan F, Nazarlue R (2014) Utilizing artificial neural network approach for solving two-dimensional integral equations. Mathematical Sciences, 8(1), 1–9
17. Piscopo M, Spannowsky M, Waite P (2019) Solving differential equations with neural networks: Applications to the calculation of cosmological phase transitions. Physical Review D, 100(1), 016002
18. Lee H, Kang I (1990) Neural algorithm for solving differential equations. Journal of Computational Physics, 91(1), 110–131
19. Tang S, Yang Y (2021) Why neural networks apply to scientific computing? Theoretical and Applied Mechanics Letters, 11(3), 100242
20. Oishi A, Yagawa G (2017) Computational mechanics enhanced by deep learning. Computer Methods in Applied Mechanics and Engineering, 327, 327–351
21. Yao H, Gao Y, Liu Y (2020) FEA-Net: A physics-guided data-driven model for efficient mechanical response prediction. Computer Methods in Applied Mechanics and Engineering, 363, 112892
22. Wu J, Wang J, Xiao H, Ling J (2017) A Priori Assessment of Prediction Confidence for Data-Driven Turbulence Modeling. Flow, Turbulence and Combustion, 99(1), 25–46
23. Xiao H, Wu J, Wang J, et al. (2016) Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier–Stokes simulations: A data-driven, physics-informed Bayesian approach. Journal of Computational Physics, 324, 115–136
24. Liu D, Nocedal J (1989) On the limited memory BFGS method for large scale optimization. Mathematical programming, 45(1), 503–528
25. Kingma D, Ba J (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980
26. Ruder S (2016) An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747
27. Raissi M, Perdikaris P, Karniadakis G, (2019) Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational physics, 378, 686–707
28. Zhang L, Cheng L, Li H, et al. (2021) Hierarchical deep-learning neural networks: finite elements and beyond. Computational Mechanics, 67(1), 207–230
29. Saha S, Gan Z, Cheng L, et al. (2021) Hierarchical Deep Learning Neural Network (HiDeNN): An artificial intelligence (AI) framework for computational science and engineering. Computer Methods in Applied Mechanics and Engineering, 373, 113452
30. Zhang L, Lu Y, Tang S, et al. (2022) HiDeNN-TD: Reduced-order hierarchical deep learning neural networks. Computer Methods in Applied Mechanics and Engineering, 389, 114414
31. Lu Y, Li H, Saha S, et al. (2021) Reduced Order Machine Learning Finite Element Methods: Concept, Implementation, and Future Applications. Computer Modeling in Engineering & Sciences, 129(1):1351
32. Liu WK, Gan Z, Fleming M (2021) Mechanistic Data Science for STEM Education and Applications. Springer

33. Belytschko T, Liu WK, Moran B, Elkhodary K (2013) Nonlinear finite elements for continua and structures. Wiley, New York

34. Zhang R, Wen L, Xiao J, Qian D (2019) An efficient solution algorithm for space–time finite element method. Computational Mechanics, 63(3), 455–470

35. Zhang R, Naboulsi S, Eason T, Qian D (2019) A high-performance multiscale space-time approach to high cycle fatigue simulation based on hybrid CPU/GPU computing. Finite Elements in Analysis and Design 166:103320

36. Zhang R, Wen L, Naboulsi S, Eason T, Vasudevan V, Qian D (2016) Accelerated multiscale space–time finite element simulation and application to high cycle fatigue life prediction. Computational Mechanics, 58(2), 329–349

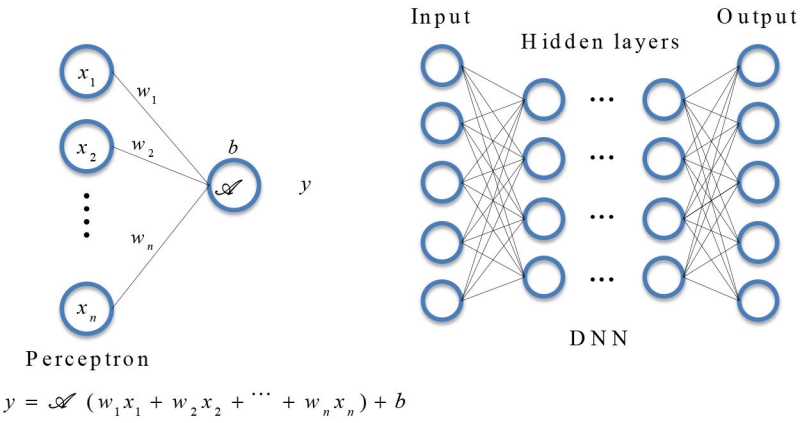


Figure 1.
An illustration of perceptron and Deep Neural Network (DNN)

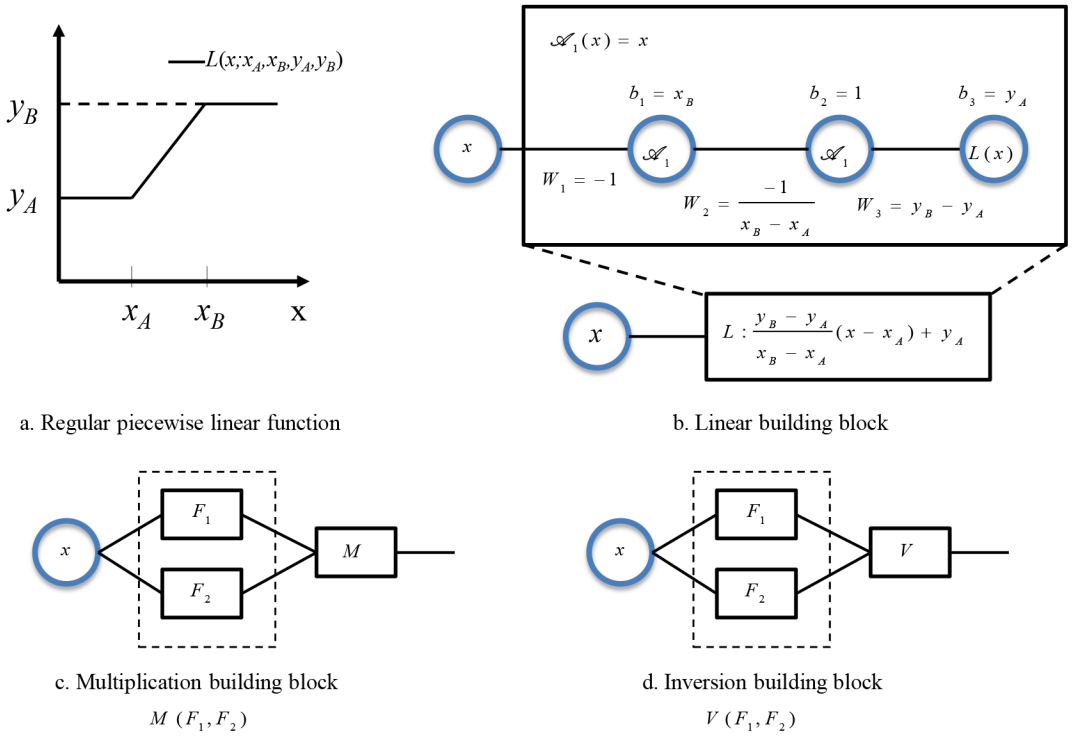


Figure 2.

An illustration of (a) piecewise linear function (b) linear building block (c) multiplication block and (d) inversion block.

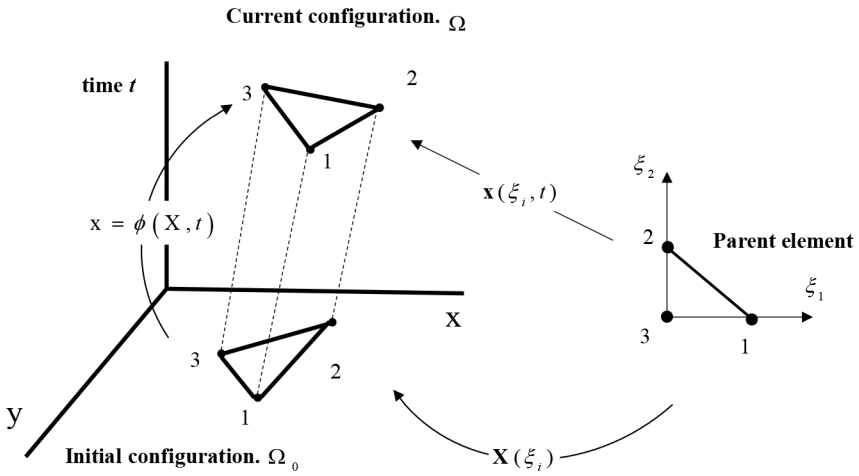


Figure 3.
An illustration of the three configurations in nonlinear FE and mapping relation.

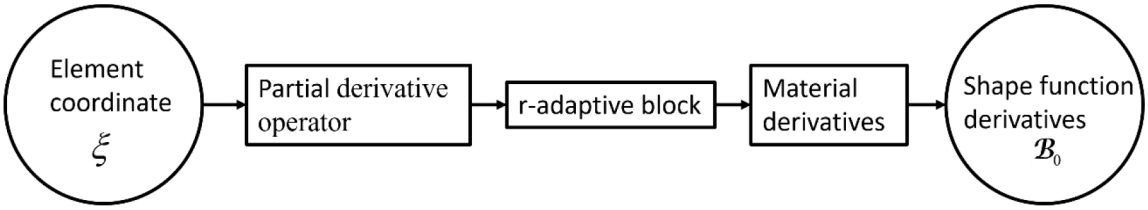


Figure 4.
Flowchart showing the three building blocks for performing shape function derivative.

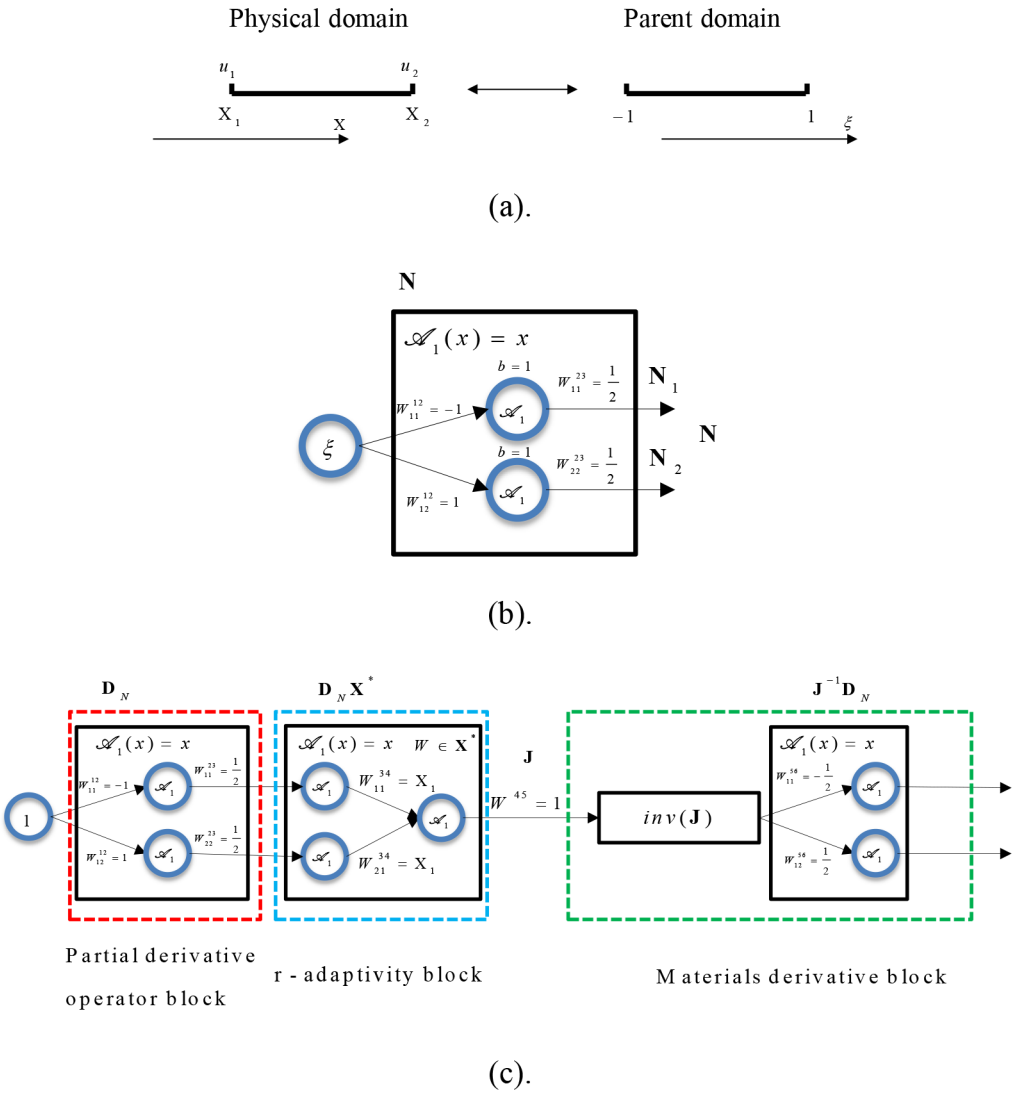
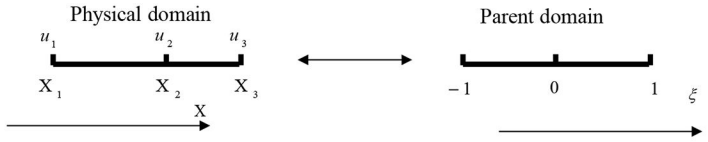
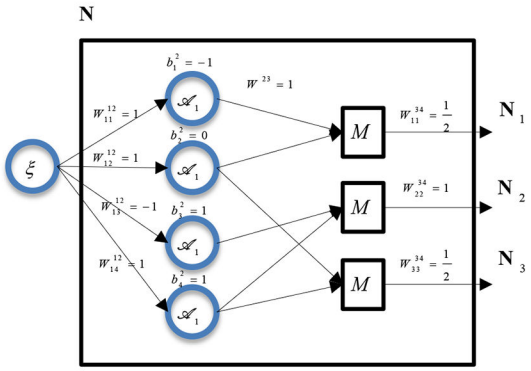


Figure 5.

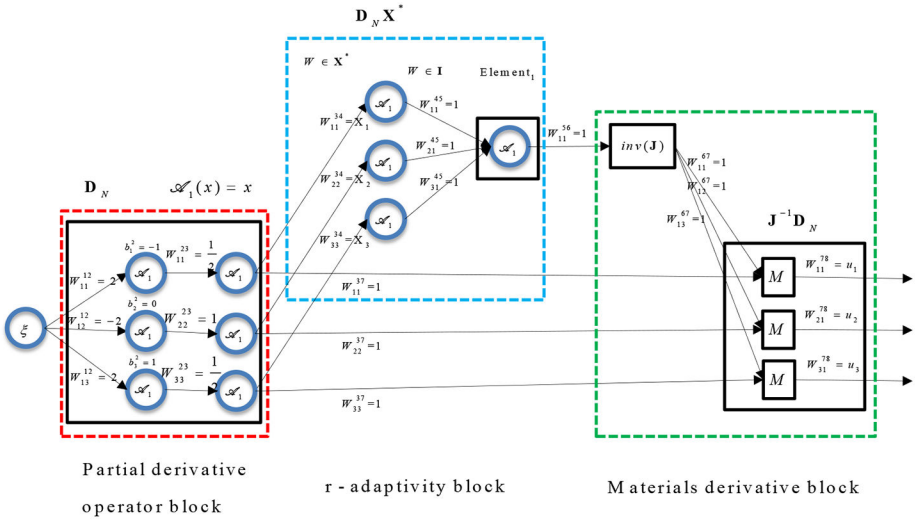
(a) Mapping between physical domain and parent domain for 1D linear element. (b) Construction of the shape function and (c) Construction of the shape function derivative using HiDeNN.



(a).



(b).



(c).

Figure 6.

(a) Mapping between physical domain and parent domain for 1D quadratic element. (b) Construction of the 1D quadratic shape function and (c) Construction of the shape function derivatives using HiDeNN.

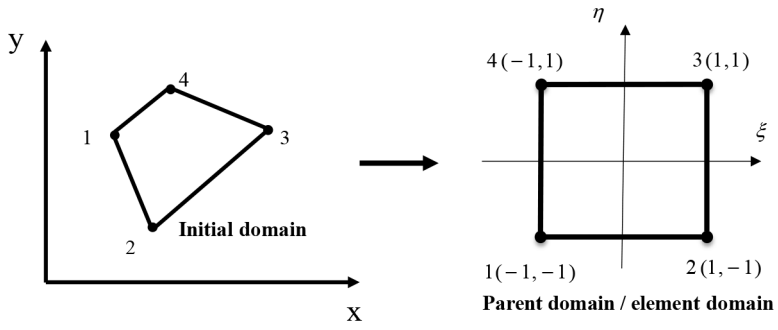


Figure 7.
An illustration of the 2D 4-node element.

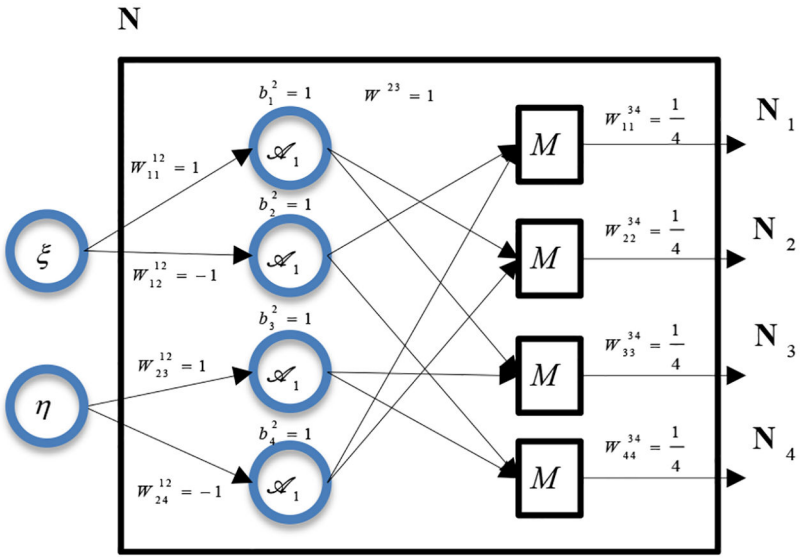
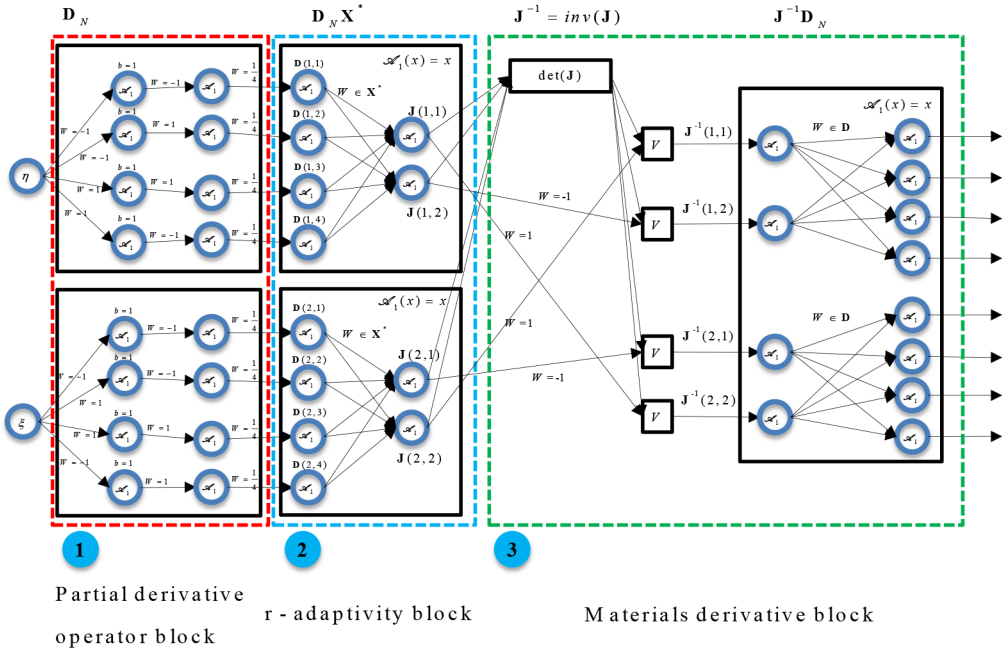
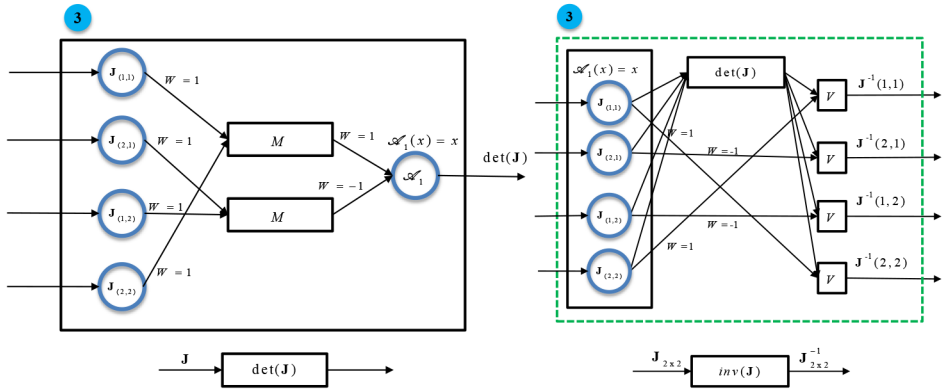


Figure 8.
Construction of the 2D 4-node element shape function.



(a)



(b). Matrix determinant building block. (c). Matrix inversion building block.

Figure 9.

(a) Construction of the shape function derivatives for the 2D 4-node quadrilateral element using HiDeNN. (b) DNN representation of the determinant det(**J**) and (c) the inverse of **J**.

Algorithm 1

- 1. Initial conditions and initialization: set $\mathbf{d}^0=\mathbf{0}$; $\boldsymbol{\sigma}^0$; n=0; t=0; $\mathbf{d}_{new}=\mathbf{d}^0$, $\Delta\mathbf{d}=\mathbf{0}$, $\mathbf{X}_{new}=\mathbf{X}$, $\Delta\mathbf{X}=\mathbf{0}$, flag=1; (flag is the variable to control mesh update)
- 2. Newton iterations for load increment n+1:
  a. compute $\mathbf{f}(\mathbf{X}_{new}, \mathbf{d}_{new}, t^{n+1})$; $\mathbf{r}=\mathbf{f}(\mathbf{d}_{new}, t^{n+1})$;
  b. compute $\mathbf{A}(\mathbf{X}_{new}, \mathbf{d}_{new})$, and $\mathbf{A}^*(\mathbf{X}_{new}, \mathbf{d}_{new})$;
  c. modify $\mathbf{A}(\mathbf{X}_{new}, \mathbf{d}_{new})$, and $\mathbf{A}^*(\mathbf{X}_{new}, \mathbf{d}_{new})$ for essential boundary conditions;
  d. solve linear equations $\Delta\mathbf{d}=-\mathbf{A}^{-1}\mathbf{(r)}$ and $\mathbf{d}_{new} \leftarrow \mathbf{d}_{new}+\Delta\mathbf{d}$;
  e. check flag condition:
    - if (flag equals 1): solve linear equations $\Delta\mathbf{X}=-\mathbf{A}^{*-1}\mathbf{(r)}$ and $\mathbf{X}_{new} \leftarrow \mathbf{X}_{new}+\alpha\Delta\mathbf{X}$, α is learning rate, set flag=0;
    - if (flag equals 0): go to step 2g;
  f. Interpolate field results (stress, strain, etc.) from previous iteration step mesh $\mathbf{X}$ to updated mesh $\mathbf{X}_{new}$.
  g. check error criterion; if not met, go to 2a.
- 3. Update displacements $\mathbf{d}$ and nodal position $\mathbf{X}$, step count and time: $\mathbf{d}^{n+1}=\mathbf{d}_{new}$, $\mathbf{X}^{n+1}=\mathbf{X}_{new}$, $n \leftarrow n+1$, $t \leftarrow t+\Delta t$;
- 4. Output; if simulation not complete, go to step 2 and set flag=1.

Figure 10.
A general algorithm (Algorithm 1) for solving nonlinear FEM with r-adaptivity.

Algorithm 2

- 1. Initial conditions and initialization: set $\mathbf{d}_0=\mathbf{0}$; load step index $n = 0$;
- 2. Outer loop: set iterative index $k = 0$, start iteration for load increment $n+1$, :
  a. $k = k + 1$

  b. computes $\left(\frac{\partial \Pi}{\partial \mathbf{d}}\right)_k$

  c. If $(k > 1)$, check if $\left|\left(\frac{\partial \Pi}{\partial \mathbf{d}}\right)_{k+1} \Big/ \left(\frac{\partial \Pi}{\partial \mathbf{d}}\right)_k\right| < \mathbf{TOL}$

  If yes, then go to step 3, otherwise continue to d

  d. Update displacement according to $\mathbf{d}_{k+1} = \mathbf{d}_k + \alpha \cdot \frac{\partial \Pi}{\partial \mathbf{d}}$ and continue for next iteration
- 3. Inner loop: Update mesh coordinates according to $\mathbf{X}^*_{n+1} = \mathbf{X}^*_n + \gamma \cdot \frac{\partial \Pi}{\partial \mathbf{X}^*}$ while holding displacement solution fixed
- 4. Output; if simulation not complete, go to step 2 and set $n = n + 1$

Figure 11.
Algorithm 2 for solving conservative systems with r-adaptivity.

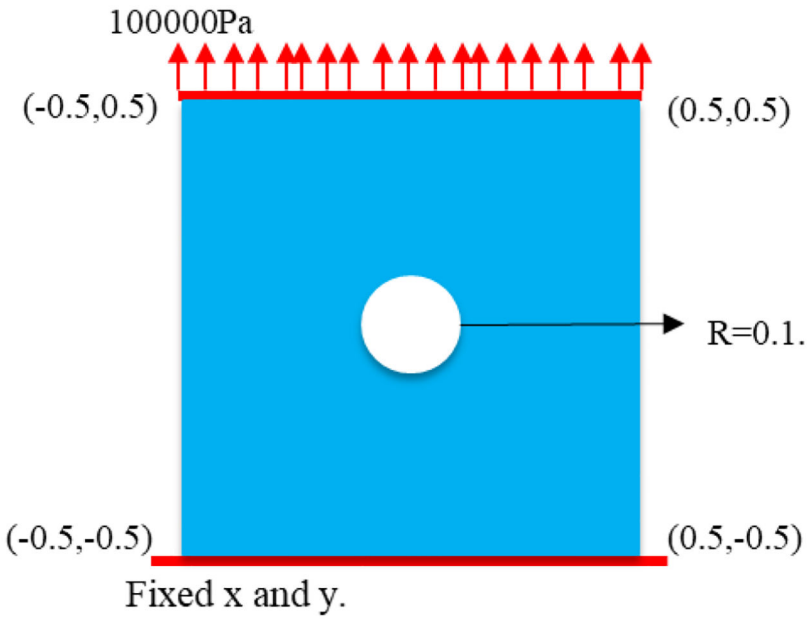


Figure 12.
2D plane stress problem of a square plate with a hole under tension.

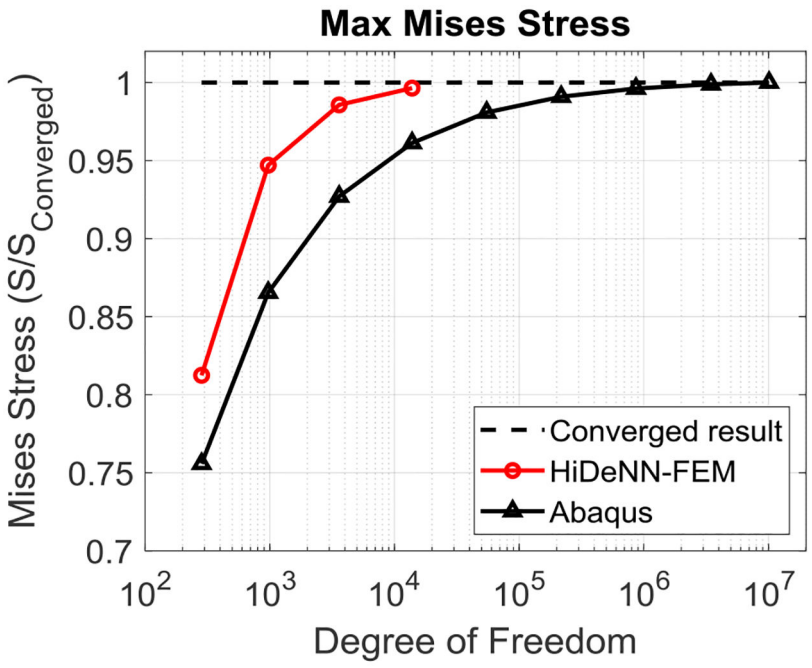


Figure 13.
Normalized maximum von Mises stress as a function of degrees of freedom.

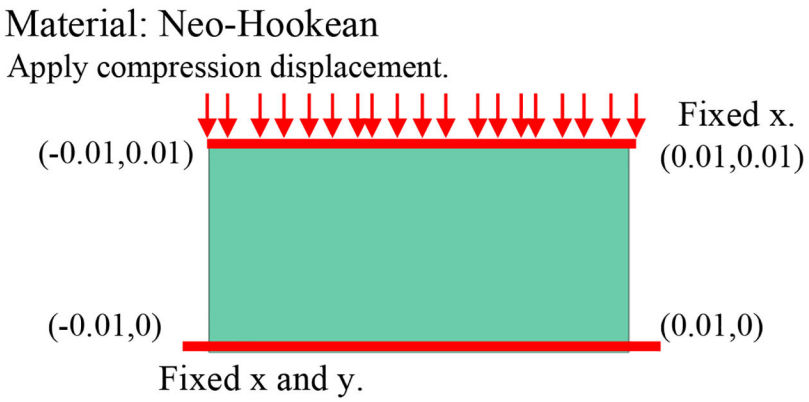


Figure 14.
2D Plane strain model of a rectangular plate under compression.

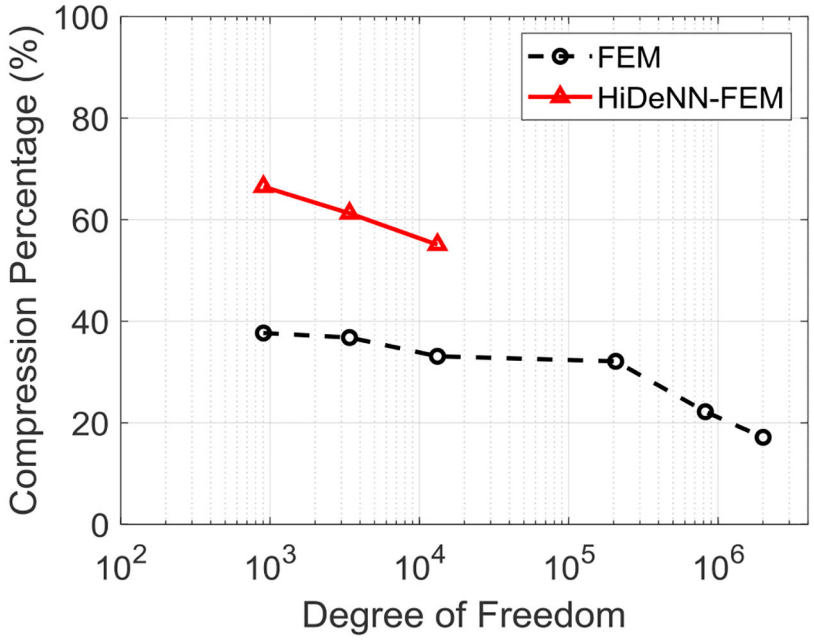


Figure 15.
Compression percentage as a function of the degrees of freedom.

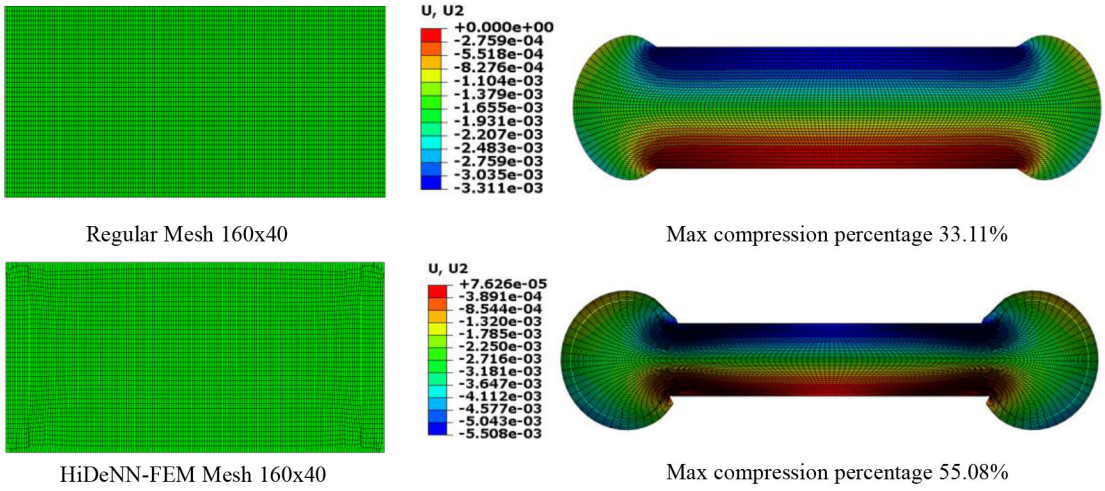


Figure 16.
Mesh of 160 by 40 case: Incompressible material compression percentage.

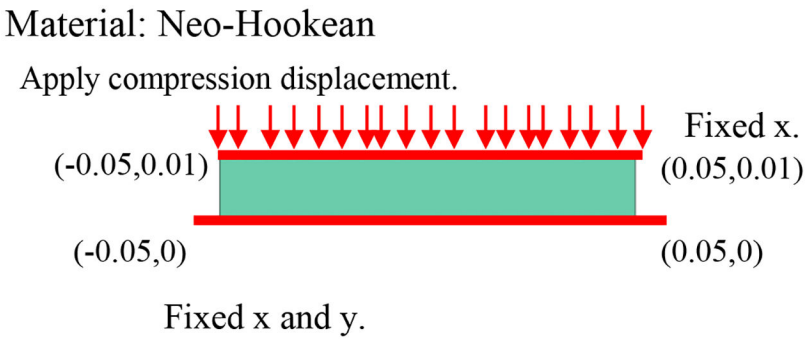


Figure 17.
Plane strain incompressible material compression case problem statement (Aspect ratio 10:1).

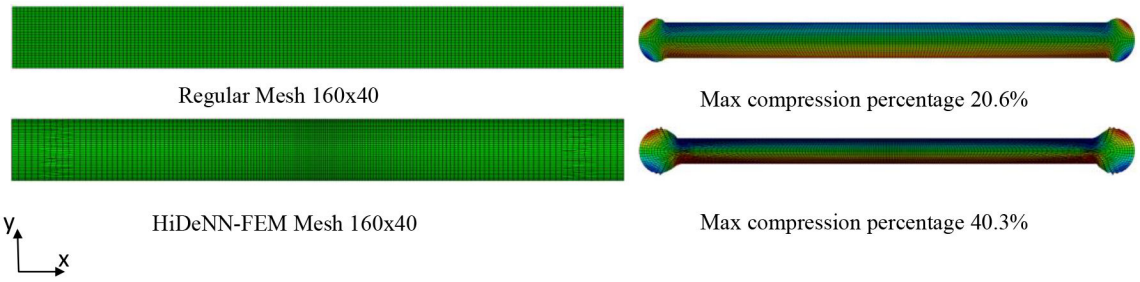


Figure 18.
Aspect ratio 10:1 plate incompressible material case compression result.

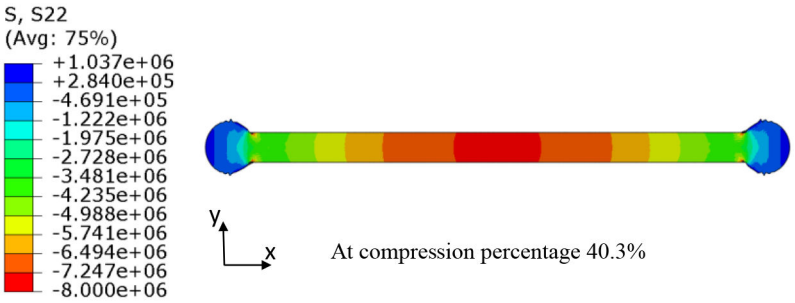


Figure 19.
Aspect ratio 10:1 plate incompressible material case compression stress-Y-Y distribution.

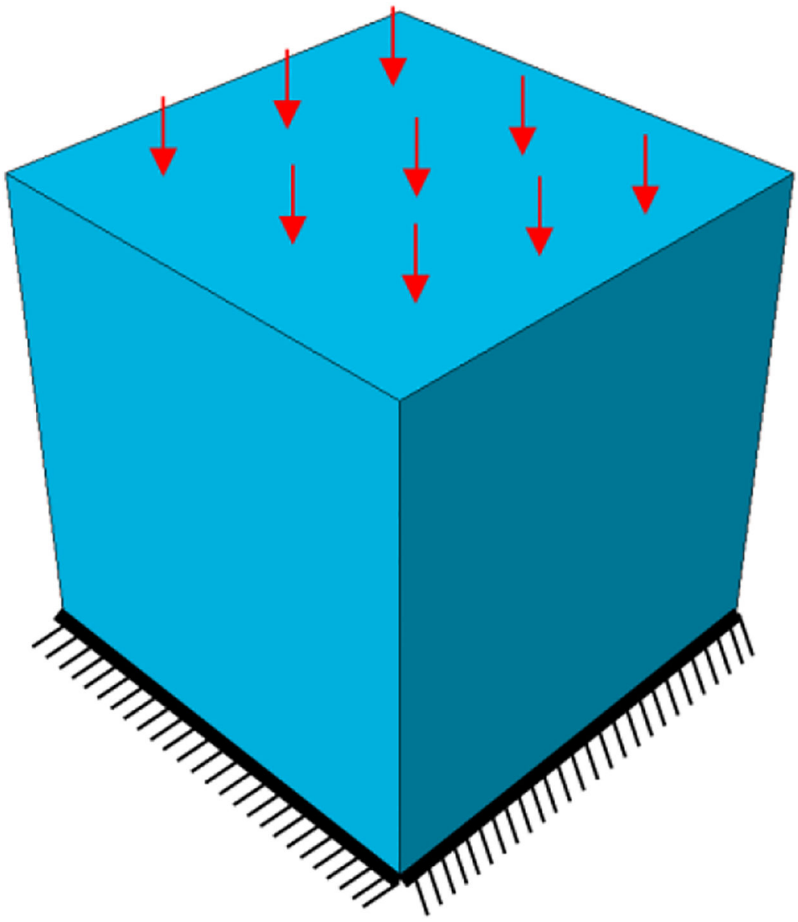


Figure 20.
Configuration of the 3D cubic block under compression

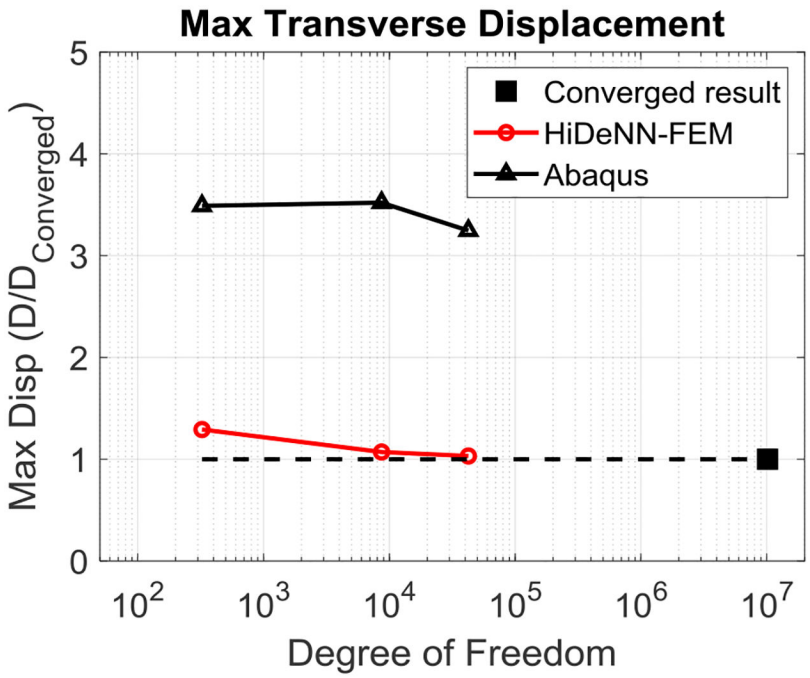


Figure 21.
Compression results between HiDeNN mesh and regular mesh.

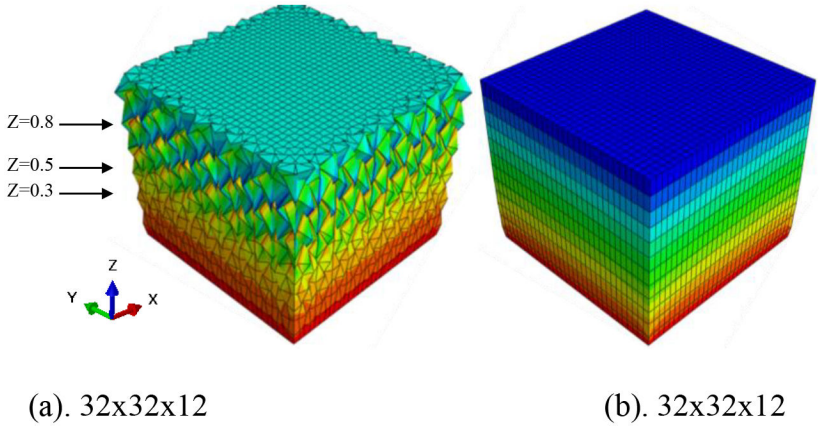


Figure 22.
Results of deformed shape with DoF 42,471 between (a) regular mesh and, (b) HiDeNN-FEM mesh.

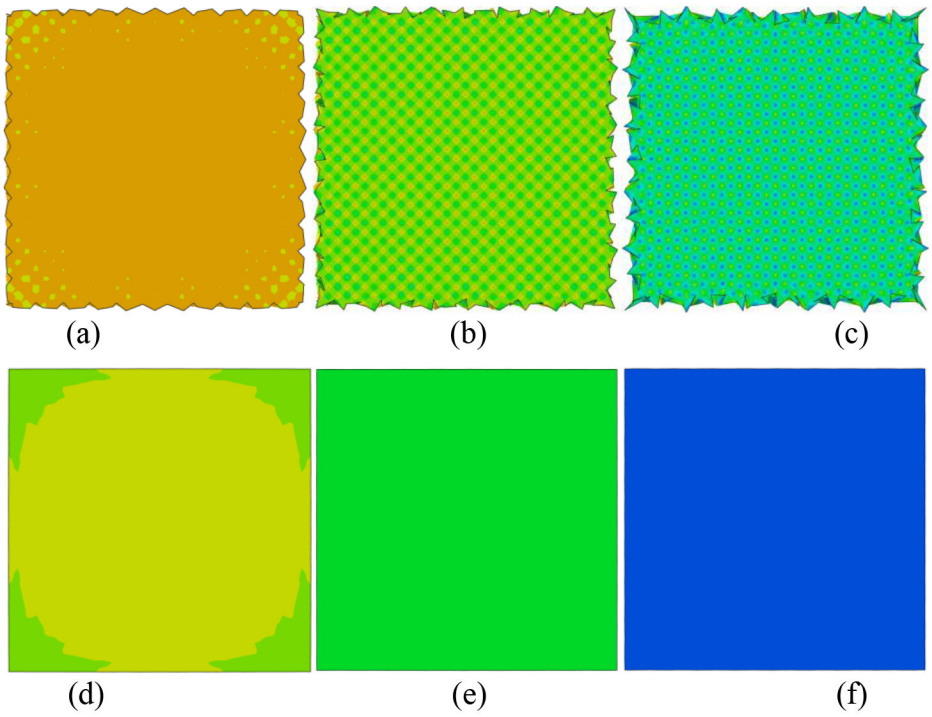


Figure 23.

Cross-sectional view of deformed shape with DoF of 42,471. The top row shows the results from Regular FEM at (a). Z=0.3m, (b). Z=0.5m, (c). Z=0.8m. The bottom row shows the corresponding from HiDeNN-FEM at (d). Z=0.3m, (e). Z=0.5m, (f). Z=0.8m.

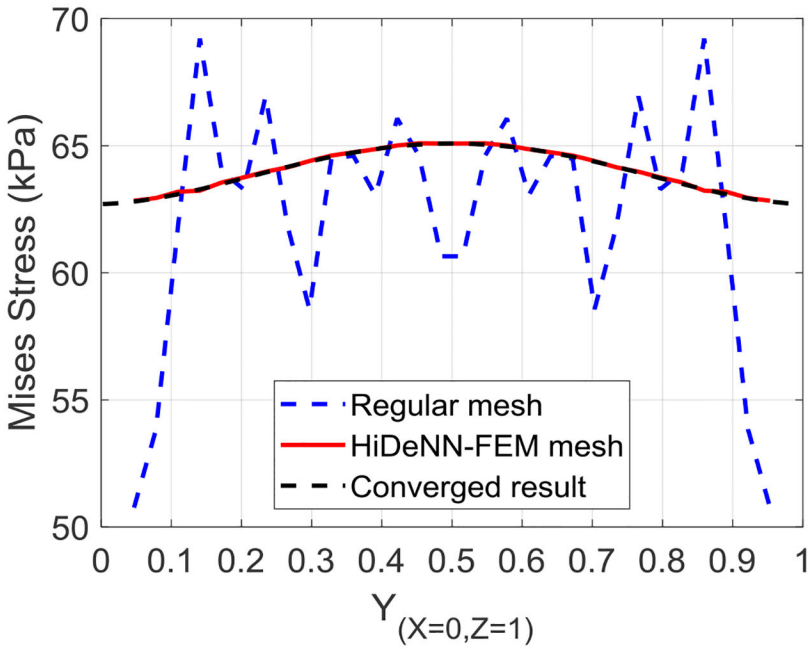


Figure 24.
von Mises stress distribution along the left side of the top surface for the case of 42,471 DoFs.

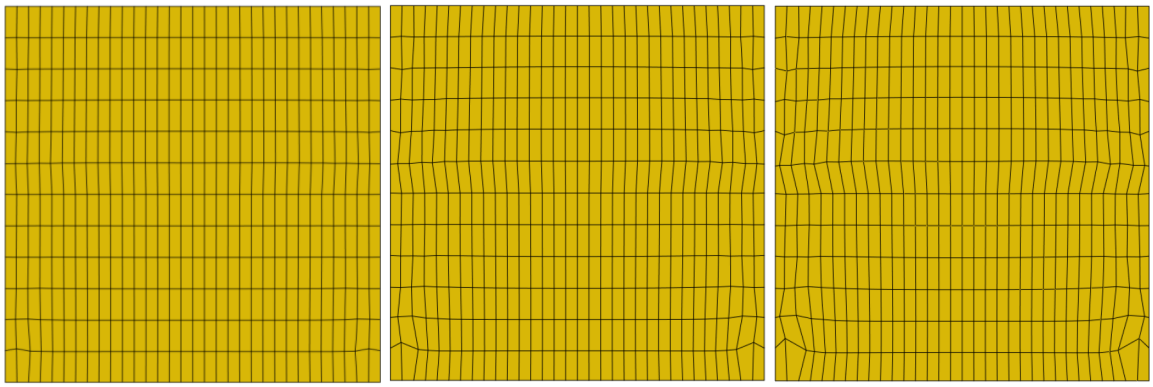
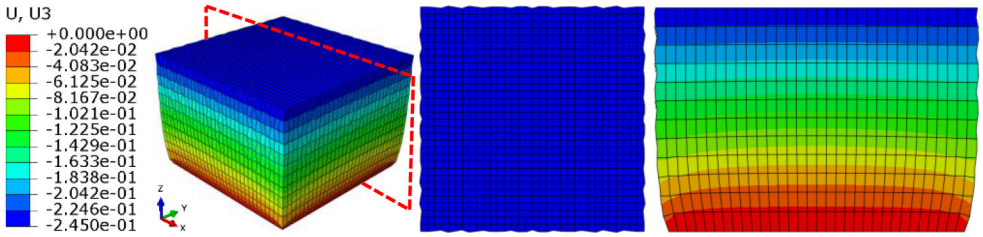
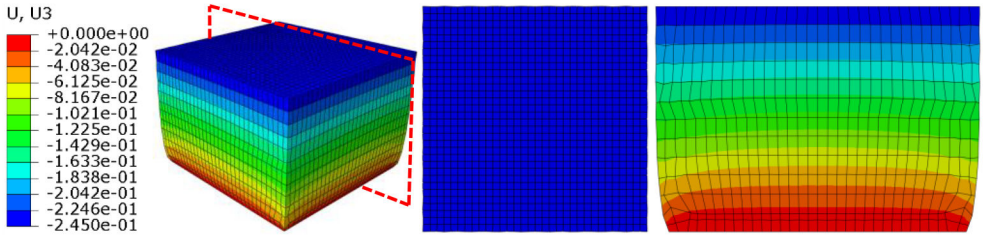


Figure 25.
X-Z Plane cross-section view of mesh update at different iterative steps.



(a). Deformed shape. (b). Top view. (c). Z-X plane cross-section view.

Regular Mesh



(d). Deformed shape. (e). Top view. (f). Z-X plane cross-section view.

HiDeNN-FEM Mesh

Figure 26.

Results from 32×32×12 mesh with hourglass control at 24.50% compression ratio. The deformed shape and cross-section view from regular FEM are shown in a, b, and c and the corresponding from HiDeNN-FEM are shown in d, e, and f.

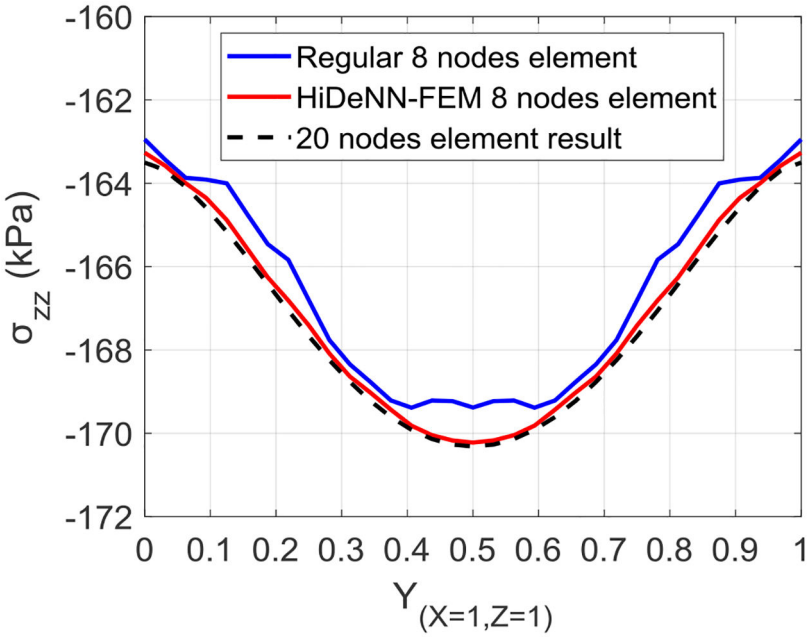


Figure 27.
Stress Z-Z distribution result at the top surface of mesh 32×32×12 case (24.50% compression ratio).

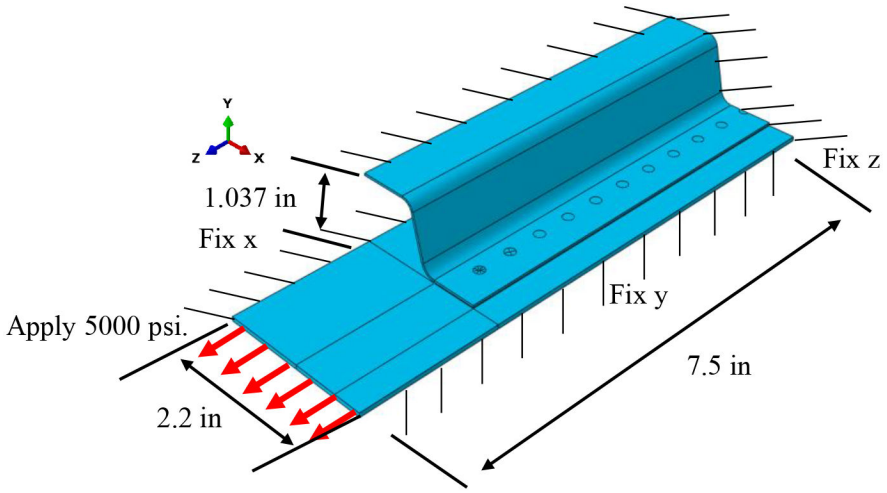


Figure 28.
Geometry and loading/boundary conditions for a hat-stiffened panel with spot weld.

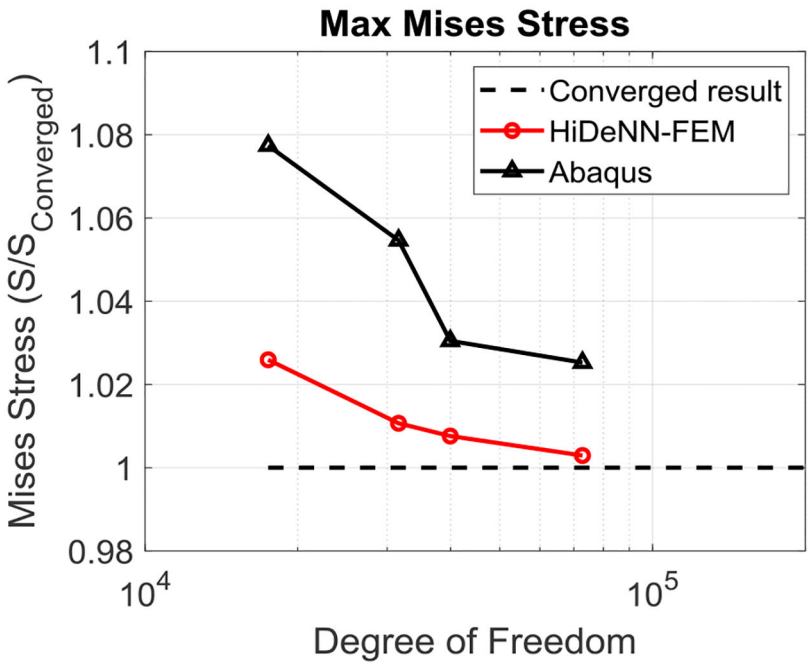


Figure 29.
Comparison of normalized maximum Mises stress as a function of degrees of freedom in spot weld region.

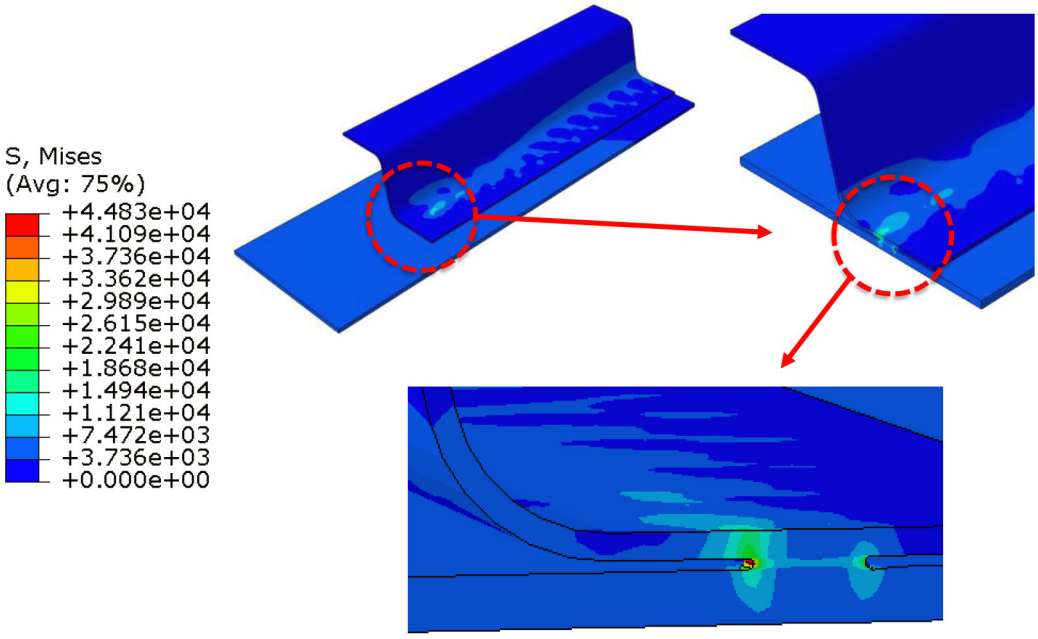


Figure 30.
Distribution of effective stress in the spot weld application.

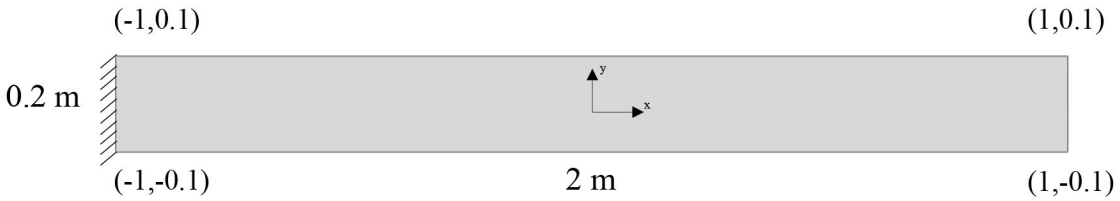


Figure 31.
A rectangular elastoplastic block subjected to non-uniform body force.

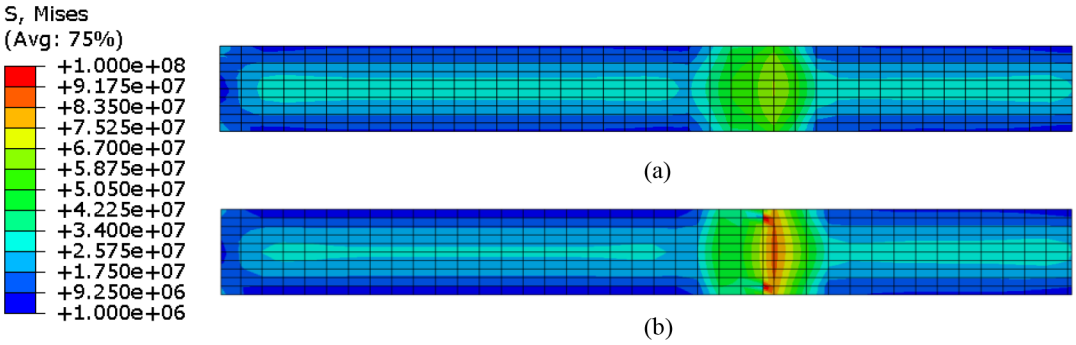


Figure 32.

A comparison between regular FEM mesh (a. 40×10) and HiDeNN-FEM mesh (b. 40×10) on the prediction of effective stress.

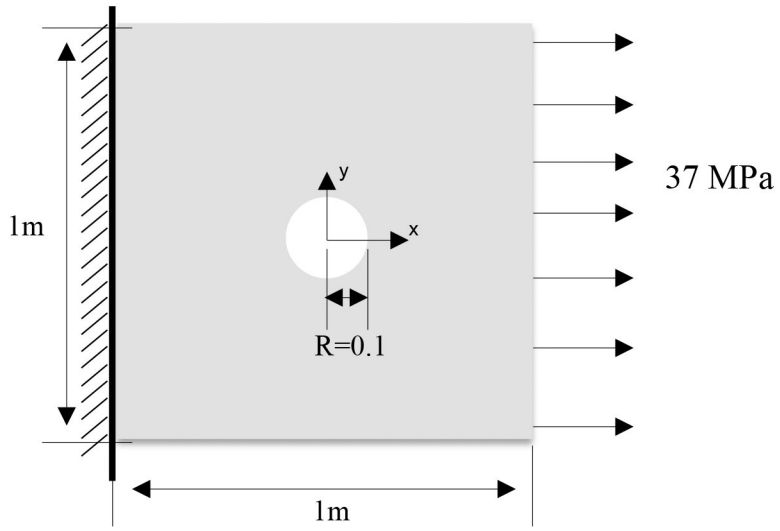


Figure 33.
FE model for a rectangular plate with a hole subjected to tension.

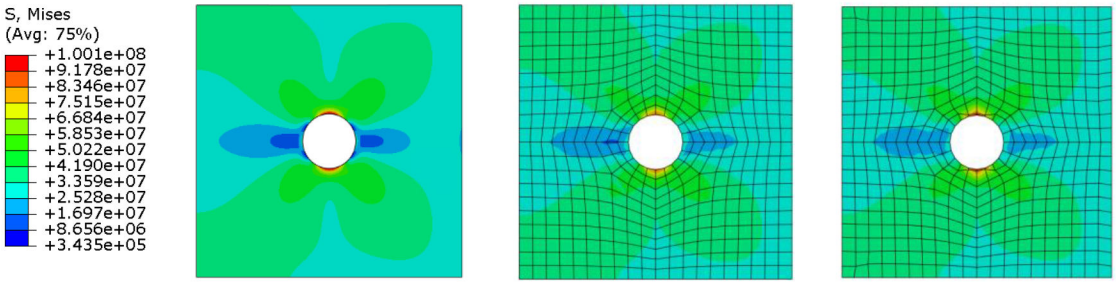


Figure 34.

Predicted von Mises stress from (left) reference solution (middle) regular FEM and (right) HiDeNN-FEM. Meshes from regular FEM (436 elements) and HiDeNN-FEM (436 elements) are shown to demonstrate the differences.

Table 1.

Plane stress tension problem results.

Analysis	Degrees of freedom	σ_{max}^{von} (Pa)	Difference
Converged solution	10,120,464	320,120	-
FEM	284	241,941	24.42%
HiDeNN-FEM		260,080	18.76%
FEM	972	277,043	13.46%
HiDeNN-FEM		303,170	5.29%
FEM	3,600	296,765	7.30%
HiDeNN-FEM		315,540	1.43%
FEM	13,840	307,722	3.87%
HiDeNN-FEM		318,980	0.36%

Table 2.

Compression percentage result of the incompressible material plate.

Analysis	Degrees of freedom	Max compression Percentage (%)	Compression Capacity Improve
FEM	902	37.69	75.21%
HiDeNN-FEM		66.51	
FEM	3,402	36.80	66.47%
HiDeNN-FEM		61.26	
FEM	13,202	33.11	66.14%
HiDeNN-FEM		55.08	

Table 3.

Four cases on the 3D cubic block problem.

Simulation Case	Regular FEM or HiDeNN-FEM	Hourglass control
a	Regular FEM	N
b	HiDeNN-FEM	N
c	Regular FEM	Y
d	HiDeNN-FEM	Y

Table 4.

Compression results between HiDeNN mesh and regular mesh.

Analysis	Degrees of freedom	Max Transverse Displacement	Difference
FEM	10,155,213	0.016693	
HiDeNN-FEM			
FEM	768	0.0582458	248.92%
HiDeNN-FEM		0.0215659	29.19%
FEM	8,670	0.0587635	252.02%
HiDeNN-FEM		0.0178751	7.08%
FEM	42,471	0.0541853	224.60%
HiDeNN-FEM		0.0172213	3.16%

Table 5.

Time consumption of 5 load steps to 10% compression ratio.

Iteration number	1st	2nd	3rd	4th	5th
Displacement solution loop	61.02s	60.69s	64.80s	61.92s	61.72s
Nodal position optimization loop	2.77s (4.54%)	2.73s (4.50%)	2.72s (4.20%)	2.71s (4.38%)	2.73s (4.42%)

Table 6.

Computational results from the hat-stiffened panel with spot weld.

Analysis	Degrees of freedom	σ_{max}^{von} (psi)	Difference	Time Consumption (FEM time plus r-adaptivity time)
Converged solution	10,044,327 (2,871,396 Elements)	44,689	------	5506s
FEM	26,235 (5,177 Elements)	48,148	7.74%	47s
HiDeNN-FEM		45,848	2.59%	47+4.54s
FEM	47,391 (8,791 Elements)	47,130	5.46%	135s
HiDeNN-FEM		45,166	1.07%	135+8.25s
FEM	59,922 (10,843 Elements)	46,051	3.05%	172s
HiDeNN-FEM		45,029	0.76%	172+10.43s
FEM	109,158 (18,919 Elements)	45,818	2.53%	203s
HiDeNN-FEM		44,829	0.29%	203+14.87s

Table 7.

A comparison of the prediction results with the reference solution.

Type of analysis	Number of elements	Degrees of freedom	σ_{max}^{von} (Pa)	Difference	CPU-based computational time (s)	Adaptivity time (s)
ABAQUS (reference solution)	5,017,600	10,046,402	1.000e8	------	16501	------
FEM	100	252	Diverge	100%	1.95	------
HiDeNN-FEM			1.002e8	0.2%	2.41	0.27(13.7%)
FEM	400	902	6.778e7	32.2%	7.47	-----
HiDeNN-FEM			1.001e8	0.1%	8.86	0.83(11.0%)
FEM	1,600	3402	9.305e7	6.9%	28.63	-----
HiDeNN-FEM			1.001e8	0.1%	33.79	3.09(10.8%)
FEM	6,400	13202	1.001e8	0.1%	107.92	-----
HiDeNN-FEM			1.001e8	0.1%	126.64	11.22(10.4%)

Table 8.

A comparison of the prediction results with the reference solution.

Type of analysis	Number of elements	Degrees of freedom	σ_{max}^{von} (Pa)	Difference	CPU-based computational time (s)	Adaptivity time (s)
ABAQUS (reference solution)	5,054,756	10,120,464	1.001e8	------	15894	------
FEM	436	972	8.886e7	11.23%	8.76	------
HiDeNN-FEM			1.014e8	1.3%	10.32	0.94(10.7%)
FEM	1,700	3,600	9.512e7	4.98%	30.81	-----
HiDeNN-FEM			1.007e8	0.6%	35.94	3.22(10.46%)